Л. А. РАСТРИГИН

Адаптация сложных систем

Методы и приложения

РИГА «ЗИНАТНЕ» 1981

6S0.1 32.81
Р245 УДК
62-506

Р а с т р и г и н Л. А. Адаптация сложных систем. — Рига: Зинатне, 1981. — 375 с.

Монография посвящена одному из бурно развивающихся направлений современной кибернетики — методам адаптации сложных объектов. В качестве алгоритмов адаптации описываются различные модификации случайного поиска как наиболее эффективного средства управления сложными объектами. Впервые рассматриваются алгоритмы структурной адаптации (альтернативной и эволюционной), позволяющие эффективно изменять структуру объекта в процессе его функционирования. Приводятся многочисленные примеры применения адаптивного подхода для решения широкого класса научно-технических задач — таких, как адаптация вычислительных систем и процессов обучения иностранному языку; синтез оптимальных решающих правил, многопороговых логических элементов и планов эксперимента; агрегация (разрезание) графов, и других. Табл. 15, ил. 115, библиогр. 304 назв.

Рецензенты:
д-р техн. наук Я. А. Гельфандбейн,
канд. техн. наук Л. П. Богомолов

Печатается по решению Редакционно-издательского совета Академии наук Латвийской ССР от 28 декабря 1979 года.

30501—126
Р М811(11)— 81 24.81.1502000000

Оглавление

ПРЕДИСЛОВИЕ

> В своих орудиях человек обладает властью над внешней природой, тогда как в своих целях он скорее подчинен ей.
>
> *Гегель*

Понятие адаптации как инструмента («орудия») целенаправленного воздействия на объект, столь распространенное в биологии и социологии (известны и широко используются феномены биологической и социальной адаптации), в последние 20 лет стало фигурировать в математической, технической и особенно в кибернетической литературе [4—11]. Трудно сказать определенно, с чем связано пристрастие математиков и инженеров к биологической терминологии. Может быть, это инерция давно начавшегося процесса «биологизации» терминов кибернетики, который идет одновременно с процессом обогащения биологии кибернетической терминологией. Но возможно, что причины здесь более глубокие: огромные возможности биологических систем, выгодно отличающие их от самых совершенных технических устройств и вызывающие пристальное внимание, восхищение и даже зависть инженеров.

Необходимость введения адаптации хорошо чувствует всякий проектировщик, которому приходится создавать систему при значительной априорной неопределенности об условиях ее функционирования. Осреднение по этой неопределенности редко бывает удачным. С другой стороны, всякое осреднение поведения среды позволяет спроектировать систему, оптимально работающую только при среднем состоянии среды. Всякое же отклонение среды от среднего приводит к неоптимальности функционирования системы.

Именно поэтому так важно вводить в систему адаптирующие подсистемы, которые будут ее изменять, с тем чтобы поддерживать ее эффективность в оптимальном состоянии независимо от состояния среды.

Как видно, термин «адаптация» уже прочно вошел в инженерный лексикон и, следовательно, нуждается в более точном определении. Действительно, разъяснение понятия адаптации как приспособления к новым условиям, которое вполне удовлетворяет биологов и социологов, совершенно неудовлетворительно с точки зрения инженера. Для строгого определения этого понятия

следует рассмотреть процессы адаптации в биологии и социологии с инженерных позиций.

В понятие адаптации как активного действия (управления) обычно вкладывают два смысла: приспособление к фиксированной среде (условно назовем пассивной адаптацией) и поиск среды, адекватной данной системе (назовем соответственно активной адаптацией). В первом случае адаптирующаяся система функционирует так, чтобы выполнять свои функции в данной среде наилучшим образом, т. е. максимизирует свой критерий эффективности функционирования в данной среде. Активная адаптация, наоборот, подразумевает либо изменение среды с целью максимизации критерия эффективности, либо активный поиск такой среды, в которой достижим желаемый комфорт.

Очевидно, что в действительности оба вида адаптации встречаются одновременно и взаимодействуют друг с другом. Растения обладают преимущественно пассивной адаптацией, а животные — активной. В социальной жизни и та и другая адаптация проявляются, по-видимому, в равной мере. В обоих случаях для осмысления адаптации как процесса необходимо разобраться по крайней мере в двух обстоятельствах:

1 . Какова цель адаптации, т. е. что называть эффективным функционированием системы?

2. Каков алгоритм адаптации, т. е. каким способом достига ется поставленная цель?

Таким образом, задавая цель (какая бы она ни была) и способ ее достижения, мы тем самым определяем адаптацию как процесс.

Это означает, что адаптация ничем не отличается от управления (в широком смысле). Действительно, адаптация, как и всякое управление, есть организация такого целенаправленного воздействия на объект, при котором достигаются заданные цели.

Отождествляя адаптацию и управление, необходимо определить тип управления, к которому относится адаптация.

Распространено мнение, что адаптацию как управление следует отнести к оптимизации в обстановке помех, в процессе которой параметры объекта изменяются так, чтобы его показатель качества стремился к экстремальному значению независимо от изменения ситуации. Аналогичное определение приводится в книге Я. 3. Цыпкина «Адаптация и обучение в автоматизированных системах» (М., Наука, 1968): «...процесс изменения параметров и структуры системы, а возможно, и управляющих воздействий на основе текущей информации с целью достижения определенного, обычно оптимального состояния системы при начальной неопределенности и изменяющихся условиях работы». Здесь четко определено требование оптимизации по заданному критерию.

Однако сложные системы (особенно биологические и социальные), как правило, не имеют единственного критерия функционирования. Такого рода системы функционируют в обстановке многокритериальности, причем эти критерии могут быть не только экстремальными, но и иметь характер ограничений. Это побуждает формулировать сразу несколько критериев и варьировать их выбор в зависимости от сложившейся ситуации и внутренних потребностей самой системы. Таким образом, уже выбор критериев адаптации является процессом адаптивным и должен учитываться при определении адаптации.

Поэтому, с учетом особенностей сложных систем, адаптацию в широком смысле можно определить как процесс целенаправленного изменения параметров и структуры системы, который состоит в определении критериев ее функционирования и выполнении этих критериев.

Это определение включает приведенное выше, но, кроме того, позволяет изменять критерии функционирования системы, по которым оценивается эффективность ее работы при организации адаптации. Введение процедуры выбора критерия оптимальности в процессе адаптации расширяет понятие адаптации и сближает его с биологическим и социологическим толкованием. Последнее обстоятельство является очень важным.

Дело в том, что в технике пока очень мало эффективно работающих адаптивных систем — во всяком случае, значительно меньше, чем хотелось бы. Такая ситуация сложилась в результате того, что реальные объекты не терпят поисковых воздействий, необходимых для организации поиска экстремума критерия оптимальности. Интерес, проявляемый в последнее время к так называемым беспоисковым системам оптимизации, вызван именно этим обстоятельством. Однако далеко не все процессы адаптации могут быть выполнены беспоисковым способом, требующим информации о структуре объекта. Поэтому проблема адаптации в технике сводится к снижению той высокой платы, которую приходится платить за процесс адаптации.

Это можно сделать по крайней мере двумя путями — выбором удачного алгоритма адаптации при фиксированном критерии и удачным варьированием критериев при фиксированном алгоритме адаптации. Третий путь довольно естественно образуется варьированием алгоритмов и критериев. В технике применяется только первый путь. Биологические и социальные системы широко используют еще и второй путь адаптации — изменение целей, в чем, по-видимому, и заключается причина удивительно гибкой адаптивности этих систем, которой пока практически лишены технические системы адаптации.

Основная цель изучения процессов адаптации должна состоять именно в отыскании причин и механизмов гибкости

процессов адаптации в биологических и социальных системах с целью их перенесения в технические системы. Именно это соображение заставляет анализировать процессы адаптации на разных уровнях: от самого низкого — технического, до самого высокого— социального.

К сожалению, пока далеко не все идеи биологической и социальной адаптации могут быть формализованы и использованы в технических системах (например, известные феномены мгновенной адаптации и преадаптации). Однако то немногое, что уже доступно, дает великолепные результаты (например, алгоритмы адаптации, моделирующие биологическую эволюцию и поведение живых существ). Именно это позволяет надеяться, что следующий «прорыв» в области адаптивных систем будет именно в направлении моделирования биологической и социальной адаптации.

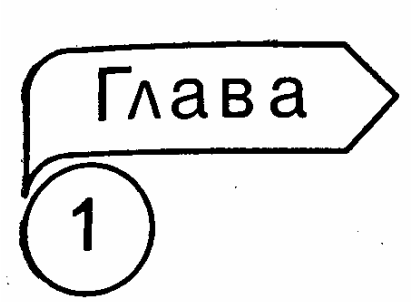
Однако в данной книге затронутые проблемы не обсуждаются: здесь пока слишком мало математически содержательных результатов. Эта книга появилась как обобщение работ автора в области адаптации и оптимизации сложных систем различного вида. Спектр рассматриваемых систем достаточно широк — от вычислительной сети до системы обучения иностранному языку. И каждая из этих систем адаптируется с использованием алгоритмов случайного поиска.

Применение случайного поиска для целей адаптации и оптимизации рассматриваемых систем имеет два основания. Во-первых, случайный поиск является наиболее эффективным средством адаптации сложных систем: он прост в реализации, легко модифицируется и программируется, имеет высокое быстродействие. Это объективные показатели, способствовавшие выбору случайного поиска в качестве основного инструмента. Есть и субъективные причины, связанные с предыдущими интересами автора, который последние 20 лет занимался разработкой и пропагандой алгоритмов случайного поиска. Естественно, что при решении прикладных задач был выбран именно случайный поиск.

Кроме теоретических вопросов в книге рассмотрено более десяти различных прикладных адаптационных и оптимизационных задач, в решении которых принимал участие автор (отсюда несколько субъективный отбор этих задач). Полученные результаты имеют достаточно общий характер и легко распространяются на решение других задач.

С книгой в рукописи ознакомились Я. З. Цыпкин, С. З. Кузьмин, Е. В. Маркова и А. Н. Скляревич, которые сделали ряд полезных замечаний, учтенных в окончательной редакции. Всем этим лицам автор выражает свою глубокую признательность.

Пос. Межциемс (Рижский р-н)
23 июля 1979 года *Автор*



Проблема адаптации

Каждый успех наших знаний ставит больше проблем, чем решает.

Де Бройль

В этой главе формулируется проблема адаптации как способа управления объектом в обстановке неопределенности среды и самого объекта. Последняя неопределенность связана прежде всего со сложностью объекта, препятствующей получению адекватной его модели. Адаптация выступает в качестве средства управления объектом при отсутствии его точной модели. Далее показано, что адаптация эквивалентна поисковой оптимизации в обстановке помех, связанных с неопределенностью среды и объекта. Адаптация противопоставляется компенсации, для реализации которой необходимо иметь адекватную модель объекта. Как всякое управление, адаптацию удобно классифицировать по способам изменения объекта. Если изменяются его параметры, то это параметрическая адаптация, а при изменении структуры — структурная.

§ 1.1. Адаптация в процессах управления сложным объектом

В процессах управления сложными объектами адаптация занимает почетное место. (Определение сложного объекта будет дано во второй главе, здесь же достаточно интуитивного представления о сложном объекте.) Дело в том, что без адаптации совершенно невозможно эффективно управлять сложным объектом (простым — можно), т. е. цели управления без адаптации не достигались бы. Не будь адаптации, пришлось бы ограничить управление самыми простыми объектами, типа объектов автоматического регулирования [265, 266].

Для того чтобы показать это, рассмотрим специфику управления сложным объектом. Прежде всего о термине «управление» [172].

1.1.1. Управление и его атрибуты

Под *управлением* будем понимать процесс организации такого целенаправленного воздействия на объект, в результате которого этот объект переводится в требуемое (целевое) состояние.

Объектом управления будем называть ту часть окружающего нас мира, состояние которой нас интересует и на которую можно целенаправленно воздействовать — управлять. Состояние объекта изменяется под воздействием среды, в которой он находится. Пусть X — состояние среды, взаимодействующей с объектом, а Y — состояние объекта (рис. 1.1.1). Тогда объект можно представить как преобразователь F^0 состояния среды в состояние объекта:

$$Y=F^0(X), \qquad (1.1.1)$$

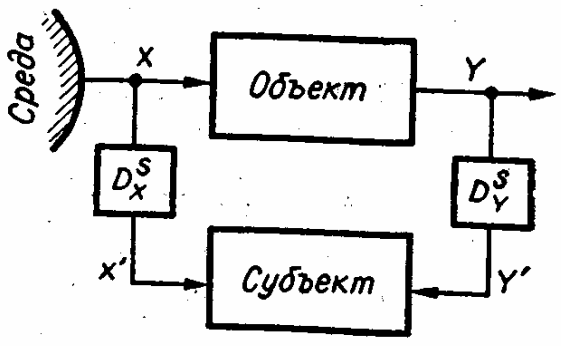
где F^0 — пока неизвестный оператор связи входа X и выхода Y объекта, характеризующий специфику его работы. (Здесь X уже выступает как вход, а Y — как выход объекта.)

Говоря об управлении как о целенаправленном процессе, нельзя обойти того, чьи цели реализуются в процессе управления. Для этого необходимо ввести понятие «субъекта», который является источником целей, реализуемых управлением. Эти цели возникают у субъекта под давлением потребностей, связанных с его жизнедеятельностью и взаимодействием со средой и объектом управления. Заметим, что под субъектом совершенно необязательно подразумевать какую-то личность; это может быть группа людей, объединенная по какому-то признаку, и даже все человечество, если рассматривается управление глобальными объектами — такими, как окружающая среда, космос и т. д.

Если состояние Y объекта удовлетворяет потребностям субъекта, взаимодействующего с этим объектом и эксплуатиру-

Рис. 1.1.1. Блок-схема объекта как преобразователя причины X в следствие

Рис. 1.1.2. Блок-схема взаимодействия субъекта и объекта управления.



ющего его, то никакого управления не нужно. Если же состояние объекта почему-либо не удовлетворяет потребностей субъекта, то последний должен организовать такое воздействие на объект, которое перевело бы объект в новое состояние, удовлетворяющее субъекта. Это и есть управление.

Так как фигура субъекта важна для управления, «выделим» его из среды (рис. 1.1.2). Здесь D_x и D_Y — система «датчиков» (его рецепторы, сенсоры), с помощью которых он воспринимает среду и объект. Информация $\langle X', Y' \rangle$ образует сенсорную среду субъекта, т. е. ту часть среды $\langle X, Y \rangle$, которую он способен воспринять своими сенсорами.

Удобно (хотя это и не соответствует действительности) считать, что субъект всегда по поводу любого объекта формулирует свою цель (цели) Z^*, реализация которой в объекте приведет, по мнению субъекта, к удовлетворению его потребностей. Эта цель представляет собой набор требований, предъявляемых субъектом к состоянию У объекта. Будем обозначать выполнение целевых требований Z* в объекте равенством

$$Y = Z^*, \qquad (1.1.2)$$

а невыполнение — неравенством

$$Y \neq Z^*. \qquad (1.1.3)$$

В последнем случае (при отсутствии управления) цели субъекта не реализуются. В результате субъекту приходится решать дилемму:

1) либо смириться с существующим положением, выражен ным неравенством (1.1.3), и тем самым терпеть определенный ущерб, связанный с неудовлетворением своих потребностей;

2) либо создать систему управления, которая реализовала бы его цели Z^* в объекте, т. е. добиваться выполнения равенства (1.1.2), но при этом затратить неизбежные средства и усилия на ее создание и эксплуатацию.

Если цели Z^* важные (т. е. потребности, их породившие, насущны), то субъект идет на создание системы управления. Для этого ему прежде всего необходимо определить, каким образом можно воздействовать на объект, т. е. назначить каналы управления. Такими каналами могут быть отдельные входы объекта X, поддающиеся целенаправленному изменению. Но этого обычно бывает мало и приходится создавать новые, до сих пор не существовавшие каналы воздействия на объект (например, изменять те параметры объекта, которые были постоянными, менять его структуру и т. д.).

Во всяком случае, для реализации управления необходима создать канал управления U, с помощью которого можно влиять на состояние объекта управления:

$$Y=F^0(X, U), \qquad (1.1.4)$$

где F^0 — по-прежнему оператор работы объекта, но учитывающий наличие фактора управления U. Теперь можно создавать *систему управления,* под которой будем понимать совокупность алгоритмов обработки информации и средств их реализации, объединенных для достижения заданных целей управления в объекте. (Заметим, что система управления далеко не всегда воплощается «в металле». Она может представлять собой систему правил, договоров и обязательств, которые реализуются в процессе управления.)

Блок-схема системы управления показана на рис. 1.1.3. Здесь D_X и D_Y — датчики, измеряющие состояние среды и объекта соответственно. Результаты измерений

$$X' = D_X(X); \qquad (1.1.5)$$
$$Y' = D_Y(Y)$$

поступают на управляющее устройство (УУ), которое вырабатывает команды управления U. Эти команды должны быть обработаны исполнительными механизмами (ИМ), с тем чтобы изменить состояние управляемого входа U' объекта.

Для функционирования управляющего устройства ему нужно сообщить цель Z^* управления (к чему следует стремиться в процессе управления), а также алгоритм управления φ — указание, как добиваться поставленной цели, располагая информацией о состояниях среды, объекта и цели:

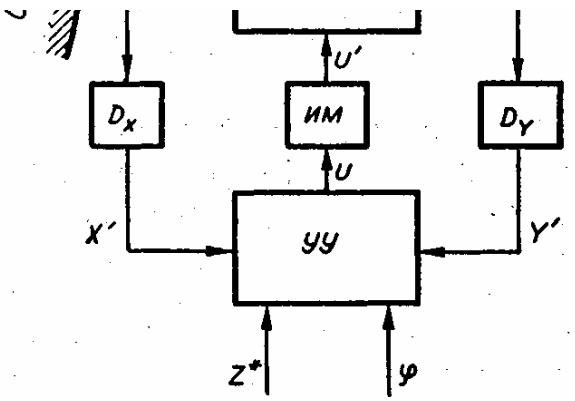


Рис. 1.1.3. Блок-схема системы управления объектом.

$$U = \varphi(X'_1 Y', Z^*). \qquad (1.1.6)$$

Как видно, управление связано прежде всего с целями $\{Z^*\}$, которые поступают извне в систему управления. Эти цели образует (генерирует) субъект, который и является потребителем будущей системы управления объектом. Субъект выступает в качестве заказчика и потребителя создаваемой системы управления.

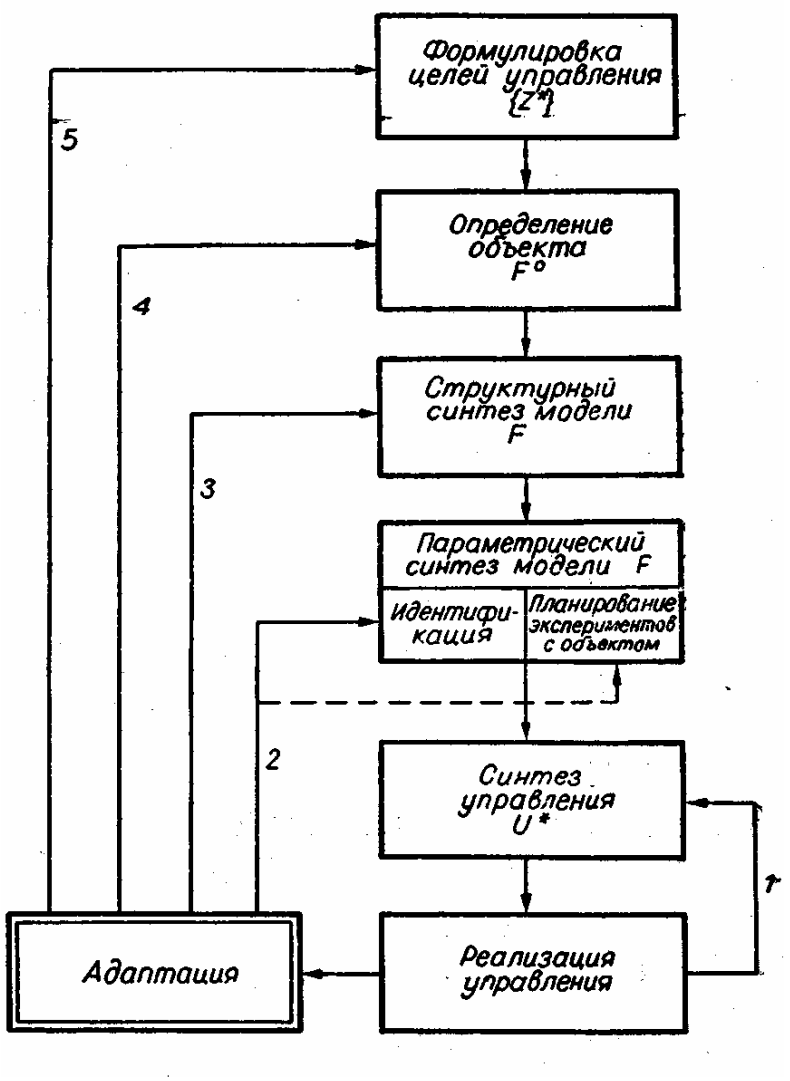


Рис. 1.1.4. Последовательность этапов управления сложным объектом.

Прежде чем принимать решение о создании системы управления, необходимо рассмотреть все этапы управления, независимо от того, с помощью каких материальных средств будут реализованы эти этапы. Такой алгоритмический анализ управления является основой для принятия решения о создании системы управления, форме ее реализации и степени автоматизации.

1.1.2. Этапы управления сложным объектом

Управление сложным объектом можно подразделить на следующие этапы (рис. 1.1.4).

I. Формулировка целей управления. На этом этапе определяются цели (множество целей), которые должны быть реализованы в процессе управления. Слово «цель» - здесь используется в смысле модели потребного будущего субъекта, т. е. некоторого определенного состояния среды, которое желательно потребителю и которое в определенном смысле «неестественно», т. е. не-

реализуется естественным образом без вмешательства извне (без управления).

Субъект в процессе общения с окружающей средой фиксирует свое внимание на тех ее параметрах, которые, с одной стороны, определяют состояние его потребностей, а с другой — могут быть им изменены, т. е. субъект располагает средствами для такого воздействия на среду, при котором эти параметры изменяются в нужную ему сторону. Будем считать, что субъект, образуя цели, реагирует только на эти параметры. Параметры среды, которые определяют его потребности, но не могут быть изменены субъектом, вообще говоря, косвенно влияют на его поведение при целеобразовании. Здесь, по-видимому, вступает в действие механизм эмоций, что не может не оказать влияния на процесс образования цели. Однако этот механизм мы рассматривать не будем.

Таким образом, субъект воспринимает окружающую среду ;как конечный или бесконечный набор ее параметров

$$S = (S_1,..., S_e), \qquad (1.1.7)$$

каждый из которых интересует субъекта и может быть им изменен. Иначе говоря, воспринимаемая субъектом ситуация всегда управляема:

$$S(U) = (s_1(U), ..., s_e(U)), \qquad (1.1.8)$$

где U — управление субъекта.

Введем понятие пространства ситуаций $\{S\}$, которое образуется указанными параметрами s_i (i=1,...,e). Каждая точка этого пространства определяет какую-то конкретную ситуацию, сложившуюся вокруг субъекта (рис. 1.1.5). Через это ситуационное пространство $\{S\}$ субъект и воспринимает окружающие его среду и различные объекты.

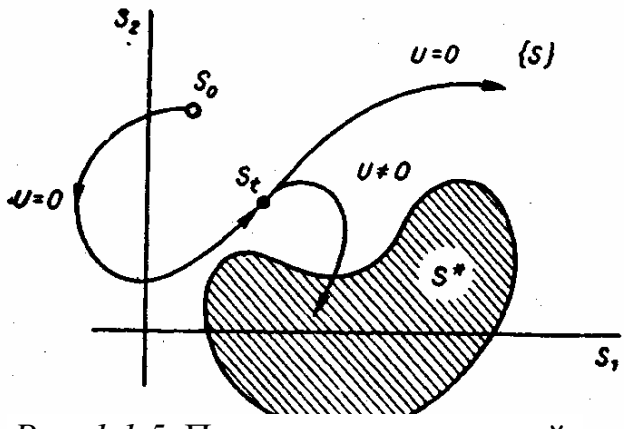


Рис. 1.1.5. Пространство ситуаций.

Естественный дрейф ситуации, вызванный изменением (эволюцией) среды, связанным, возможно, с действиями других субъектов, приводит к смещению точки S вдоль какой-то траектории (см. сплошную линию на рис. 1.1.5).

Однако свои цели субъект формулирует не в терминах среды S: субъекту удобнее оперировать иными, свойственными ему понятиями (на-

зовем их целевыми). Пусть эти целевые понятия описываются вектором

$$Z = (z_1,...,z_k), \qquad (1.1.9)$$

где каждый целевой параметр z_i однозначно определяется ситуацией S, т. е.

$$z_i=\psi_i(S) \quad (i = 1, ..., k), \qquad (1.1.10)$$

а функции ψ_i (•) определяют связь состояния среды I и целевого параметра z_i. В векторной форме эта связь выражается в виде

$$Z=\psi(S),$$

(1.1.11)

где

$$\psi(S) = (\psi_1(S), ..., \psi_k(S)) \qquad (1.1.12)$$

— некоторая определенная вектор-функция.

Преобразование Ψ информации в форму Z необходимо еще и потому, что субъект обычно формулирует свои цели в терминах к понятиях, связанных с измеряемыми, но не тождественных им. В частном случае может оказаться, что $Z = Y$ ($k = m$), но это бывает редко. Например, при управлении температурным режимом объекта достаточно измерять температуру, т. е. $z = y = t^0$, так как цель сформулирована в терминах измерений. Однако при создании оператору комфортных условий необходимо измерять температуру и влажность, в то время как цель формулируется в виде определенного ограничения на определенную их комбинацию.

Рассмотрим k-мерное пространство целей $\{Z\}$, которое образуется точками (1.1.9). Это пространство удобно субъекту тем, что по поводу каждой координаты он может высказать свое требование (цель), выполнение которого, по мнению субъекта, приведет к удовлетворению какой-то одной или нескольких его потребностей. Свою цель субъект формулирует в виде вектор-цели

$$Z^* = (z^*_1, ..., z^*_k), \qquad (1.1.13)$$

где z^*_i — i-е требование к состоянию среды S, выраженное с помощью функции $\psi_i(S)$. Эти цели-требования могут иметь различный характер, но форма их должна быть унифицирована.

Рассмотрим унифицированный вид целей Z^*. Они могут выражаться различным образом. Однако для формализации их необходимо свести к одной из следующих форм:

1) z^*_i («приравнять»): $z_i=a_i$ это требование означает, что i-я целевая переменная $z_i=\psi_i(S)$ должна быть равна заданной величине a_i;

2) z^*_j («ограничить»): $z_j \geq b_j$ это требование, накладывае мое на j-ю целевую переменную, — она не должна быть меньше заданного порога b_j;

3) z^*_v («минимизировать») : $z_v \to \min$, т. е. *v-я* целевая пере менная должна быть минимальной.

Если цели субъекта не могут быть сведены к этим формам, то нельзя говорить о создании формальной системы управления для достижения этих целей.

Однако практика показывает, что к такого рода требованиям («приравнять», «ограничить», «минимизировать») можно свести почти все ситуации, которые встречаются субъекту, особенно в научно-технической сфере.

Таким образом, процесс формулировки целей Z^* субъекта связан, во-первых, с определением вектор-функции $\psi(S)$ (1.1.12) и, во-вторых, с выработкой требований, накладываемых на каждую составляющую этого вектора.

Рассмотрим, как отражается цель Z^* в пространство ситуаций $\{S\}$. Для этого достаточно рассмотреть область, определяемую системой целевых требований:

$$S^*: \begin{cases} \psi_i(S) = a_i & (i=1, \ldots, s); \\ \psi_j(S) \geqslant b_j & (j=s+1, \ldots, s+p); \\ \psi_v(S) \to \min & (v=s+p+1, \ldots, s+p+l), \end{cases} \tag{1.1.14}$$

где $s + p + l = k$.

Точка или область S^*, удовлетворяющая этим требованиям, и является тем состоянием среды, которого добивается субъект. Удастся ли субъекту достичь такого состояния среды, зависит ют его возможностей воздействовать на среду, т. е. от вида зависимости

$$S = S(U) \tag{1.1.15}$$

и от ресурсов R, выделяемых на управление:

$$U \in R. \tag{1.1.16}$$

Эти ресурсы определяют энергетические, материальные, временные и другие возможности управления U.

Теперь рассмотрим взаимодействие целевой зоны S^* и траектории изменения среды $S(t)$ под воздействием внешних факторов (см. рис. 1.1.5), т. е. дрейф ситуации. Если траектория дрейфа $S(t)$ проходит через зону, никакого управления субъекту не нужно. Ему остается ждать, когда внешние обстоятельства приведут к тому, что $S(t)$ принадлежит S^*. Однако рассчитывать на подобное маловероятное совпадение неразумно, хотя в принципе оно воз-

можно. Именно поэтому субъект предпочитает управлять ситуацией, т. е. целенаправленно воздействует на среду:

$$S_t = S(U, t) \qquad (1.1.17)$$

и изменяет ее так, чтобы

$$S(U, t) \in S^*, \qquad (1.1.18)$$

т. е. чтобы достигались цели управления. В этих выражениях время t обозначает дрейф свойств среды.

Таким образом, управление U необходимо субъекту для того, чтобы:

1) добиться поставленной цели управления Z^*, т. е. реали зовать условие (1.1.18);

2) компенсировать дрейф ситуации, который, как правило, нарушает целевое условие (1.1.18).

Именно поэтому всякое управление следует рассматривать с двух точек зрения: во-первых, как средство добиться поставленных целей и, во-вторых, как средство компенсации неблагоприятных изменений в среде, нарушающих выполнение этих целей.

Заметим, что под параметрами среды S здесь и в дальнейшем подразумеваются измеряемые параметры собственно среды X и параметры объекта Y, взаимодействующего со средой. Таким образом,

$$S = \langle X, Y \rangle. \qquad (1.1.19)$$

Однако дифференциация 5 возникает лишь после выделения объекта из среды. Процедура такого выделения представляет собой следующий этап управления сложным объектом.

II. Определение объекта управления связано с выделением той части среды, состояние которой интересует потребителя в связи с реализацией сформулированных им целей.

Цели и ресурсы управления позволяют выделить ту часть пространства, состояние которой необходимо контролировать и на которую следует воздействовать, для того чтобы выполнить заданные цели управления.

Иногда, когда границы объекта очевидны, такой проблемы не возникает. Это бывает в случаях, когда объект достаточно автономен (самолет, корабль, любой прибор, автомашина и т. д.). Однако в других случаях связи «объекта» со «средой» настолько сильны и разнообразны, что порой очень трудно понять, где кончается объект и начинается среда. Именно это обстоятельство и заставляет выделять процесс определения объекта в самостоятельный этап управления.

Задача заключается в том, чтобы для заданного множества целей $\{Z^*\}$ и ресурсов R определить такой вариант объекта, который по критерию достижимости этих целей окажется лучше всех.

Если бы мы располагали формальным описанием среды, то процесс выделения объекта из этой среды в принципе не представлял бы трудностей. Действительно, «высекая» различные «куски» среды и называя их объектом, мы всегда могли бы проверить на модели, достигаются ли цели управления в данном объекте или нет. Если нет, то можно было бы оценить численно, какова степень неуправляемости этого варианта объекта. Повторив эту процедуру для других вариантов высечения, мы остановились бы на том, который позволяет получить максимальную (желательно 100%-ную) управляемость.

Однако формального описания среды нет и этот путь закрыт. Тем не менее направление поиска довольно очевидно. Всегда, когда нет (или пока нет) формального аппарата решения проблемы, за решением (хотя и очень приближенным) обращаются к экспертам. На стадии определения объекта это единственно возможный подход. Для этого следует экспертно синтезировать несколько вариантов объекта, а затем также с помощью экспертов оценить их по критерию и выбрать наилучший.

III. Структурный синтез модели [178]. Под структурой будем понимать вид, характер зависимости F состояния объекта Y от его входов — неуправляемого X и управляемого U:

$$Y = F(X, U). \qquad (1.1.20)$$

В общем случае зависимость F определяется некоторым алгоритмом (правилом, инструкцией), который указывает, как, располагая информацией о входах X и U, определить выход У. Вид этого алгоритма с точностью до его параметров и определяет структуру F. Условно можно считать, что модель F состоит из структуры и параметров:

$$F = \langle St, C \rangle, \qquad (1.1.21)$$

где St - структура модели F, $C = (c_1, \ldots, c_k)$ — ее параметры. Таким образом, целью третьего этапа является определение структуры St объекта управления. Например, категории линейности, статичности, детерминированности, дискретности являются структурными категориями. Так, линейная статичная непрерывная детерминированная структура однозначно определяет следующий вид для F: $y = c_1x_1+\ldots+c_nx_n$, причем на стадии структурного, синтеза конкретные значения параметров $c_i, \ldots, c_n$ пока неважны. Важен лишь вид зависимости F от этих параметров и входов объекта.

Сам по себе структурный синтез модели является сложным и многоэтапным процессом и подразумевает следующие подэтапы:

1. Определение входов и выходов объекта, т. е. синтез мо дели на уровне «черного ящика».
2. Экспертное ранжирование входов и выходов объекта.
3. Декомпозиция модели.
4. Выбор структурных элементов модели.

Экспертный метод решения перечисленных задач является основным.

IV. Параметрический синтез модели связан с определением параметров $C = (c_1, ..., c_k)$ модели

$$Y = F(X, U, C), \qquad (1.1.22)$$

где выбранная на предыдущем этапе структура *St* отражена в модельном операторе *F*.

Для определения параметров *C* модели, очевидно, необходимо иметь информацию о поведении входов *X, U* и выхода У объекта. В зависимости от того, как получена эта информация, различают два подхода — идентификацию и планирование экспериментов с объектом.

IV.1. Идентификация [178] параметров модели *F* объекта связана с оценкой численных значений искомых параметров в режиме нормального функционирования объекта, т. е. без организации специальных управляющих воздействий на него. Исходной информацией для идентификации являются структура *St* и наблюдения за поведением входа $X(t)$ и выхода $Y(t)$ объекта при его взаимодействии со средой.

Таким образом, пара

$$J(t) = \langle X(t), Y(t) \rangle. \qquad (1.1.23)$$

полученная в режиме нормального функционирования объекта, является основным источником информации при идентификации. Однако не все входы объекта *(X* и *U)* изменяются в процессе его нормальной эксплуатации. Так, наверняка не изменяются те параметры из *U,* на которые не влияет состояние среды. Для выяснения зависимости выхода объекта У от параметров такого рода необходимо преднамеренно их варьировать, т. е. необходим эксперимент с объектом. Однако всякого рода эксперименты нарушают режим нормального функционирования объекта, что всегда нежелательно. Поэтому эксперимент, которого нельзя избежать, следует проводить, минимально возмущая объект, но так, чтобы получить максимальную информацию *о* влиянии варьируемых параметров на выход объекта. Здесь приходят на помощь методы планирования эксперимента.

IV.2. В процессе **планирования эксперимента** синтезируется специальный план эксперимента, позволяющего в заданных ограничениях с максимальной эффективностью определить параметры C модели объекта управления. Например, для статического объекта этот план представляет собой набор состояний управляемого входа объекта $U_1, ..., U_N,$ принадлежащих заданной допустимой области варьирования, в которых определяется его выход $Y_1,...,Y_N$, т. е. $Yi = F^0 (U_i)$ (i=1,...,N). Полученные N пар $\langle U_i, Y_i \rangle$ (i=1,...,N) являются исходной информацией для определения необходимых параметров модели.

Поскольку в процессе проведения экспериментов на объекте получается новая информация, то могут измениться представления о структуре модели (например, первоначальная гипотеза о линейности модели сменится на нелинейную). Это обстоятельство заставляет снова обращаться к структурному синтезу, точнее, вводить коррекцию структуры модели. Сказанное несколько «размывает» понятие этапа планирования эксперимента, распространяя его и на процессы выбора и коррекции структуры.

После того как выяснено влияние на выход объекта Y неуправляемого X и управляемого U входов, задачу синтеза модели, которой были посвящены два последних этапа (третий и четвертый), можно считать выполненной. Полученная модель является исходной для процесса синтеза управления.

V. Синтез управления связан с принятием решения о том, каково должно быть управление U, чтобы в сложившейся ситуации S достигнуть заданной цели управления Z^* в объекте. Это решение опирается на имеющуюся модель объекта $F,$ заданную цель Z^*, полученную информацию о состоянии среды X и объекта $Y,$ а также на выделенные ресурсы R управления, которые представляют собой ограничения, накладываемые на управление U в связи со спецификой объекта и возможностями системы управления. Синтез управления сводится к решению вариационной задачи, которая получается из (1.1.14) путем соответствующих преобразований и свертки, необходимых для преодоления многокритериальности.

Полученное управление должно быть оптимально с точки зрения целей управления и представляет собой, вообще говоря, программу изменения управляемых параметров во времени, т. е.

$$U^* = U^* (t). \qquad (1.1.24)$$

VI. Реализация управления связана с процессом отработки объектом программы, полученной на предыдущем этапе. Такой процесс для неактивных объектов решается методами теории следящих систем, отрабатывающих заданную программу. Значительно более сложна реализация управления активной системой.

Однако эти трудности должны быть преодолены на стадии синтеза модели объекта управления, учитывающей его активность. Тогда отработка программы будет такой же, как и при управлении пассивным объектом.

Если управление реализовано, а его цель не достигнута (напомним, что рассматривается управление сложным объектом), приходится возвращаться к одному из предыдущих этапов. Даже в самом лучшем случае, когда поставленная цель достигнута, необходимость обращения к предыдущему этапу вызывается изменением состояния среды X или сменой цели управления Z^*. Таким образом, при самом благоприятном стечении обстоятельств следует обращаться к этапу синтеза управления (стрелка *1* на рис. 1.1.4), на котором определяется новое управление, отражающее новую, сложившуюся в среде ситуацию. Так функционирует стандартный контур управления, которым пользуются при управлении простыми объектами.

VII. Специфика сложного объекта управления требует расширения описанного цикла за счет введения этапа **адаптации**, т. е. коррекции всей системы управления или, точнее, — всех этапов управления. Адаптация здесь выступает в роли более глубокой обратной связи, улучшающей процесс управления сложнюй системой. Рассмотрим ее подробнее.

1.1.3. Адаптация системы управления

Адаптация как процесс приспособления системы управления к специфическим свойствам объекта и окружающей среды имеет несколько иерархических уровней, соответствующих различным этапам управления сложным объектом. Начнем с нижнего уровня.

1. Параметрическая адаптация связана с коррекцией, подстройкой параметров C модели. Необходимость в такого рода адаптации возникает ввиду дрейфа характеристик управляемого объекта. Адаптация позволяет подстраивать модель на каждом шаге управления, причем исходной информацией для нее является рассогласование откликов объекта и модели, устранение которого и реализует процесс адаптации (стрелка *2* на рис. 1.1.4).

Такого рода адаптивное управление часто называют управлением с адаптивной (или адаптирующейся) моделью объекта. Преимущества его очевидны. Методы и аппарат, которые при этом используются, присущи этапу идентификации (см. выше, этап IV.1). Именно поэтому адаптация параметров связана стрелкой *2* с их идентификацией, т. е. с определением парамет-

ров в режиме нормального функционирования управления объектом.

Однако процесс управления объектом часто не предоставляет достаточной информации для коррекции модели, так как управление недостаточно разнообразно, чтобы дать информацию о специфических свойствах объекта, которые необходимы для синтеза управления и которые следует отразить в модели объекта. Это обстоятельство заставляет искусственно вводить в управление дополнительное разнообразие в виде тестовых сигналов, накладывающихся на собственно управление. Организация этих сигналов и образует следующий контур адаптации, показанный пунктиром на рис. 1.1.4. При этом, строго говоря, снижается эффективность управления. Однако полученная информация позволяет адаптировать модель, что гарантирует успех управления на последующих шагах.

Адаптивное управление, в процессе которого не только достигаются цели, но и уточняется модель, называют дуальным, т. е. двойственным. Здесь путем специальной организации управления сразу достигаются две цели — управления и адаптации модели. Методически введение тестовых сигналов соответствует решению задачи планирования эксперимента. Действительно, при дуальном управлении следует таким образом воздействовать тестовыми сигналами на объект, чтобы, минимально нарушая нормальное функционирование процесса управления, получить максимальную информацию о специфике объекта в целях использования этой информации для коррекции модели. Как видно, это типичная задача планирования эксперимента (см. также положение пунктирной стрелки на рис. 1.1.4).

2. Структурная адаптация. Далеко не всегда адаптация модели путем коррекции ее параметров позволяет получить адекватную модель объекта. Неадекватность возникает при несовпадении структур модели и объекта. Если в процессе эволюции объекта его структура изменяется, то такая ситуация складывается постоянно. Указанное обстоятельство заставляет обращаться к адаптации структуры модели, что реализуется методами структурной адаптации. Например, здесь можно воспользоваться процедурой перехода от одной альтернативной модели к другой. При этом альтернативы могут различаться числом и характером входов-выходов модели, вариантами декомпозиции и структурой элементов модели.

Альтернативные модели нуждаются в идентификации параметров, что осуществляется отмеченными выше методами параметрической адаптации.

Методически структурная адаптация модели использует алгоритмы структурного синтеза. Поэтому ее реализация в алгоритме осуществляется путем, показанным на рис. 1.1.4 стрелкой *3*.

3. Адаптация объекта. Если и структурная адаптация модели не позволяет повысить эффективность функционирования (на пример, какие-то цели управления не реализуются в объекте), то следует адаптировать объект управления (стрелка *4* на рис. 1.1.4). Эта адаптация связана с изменением объекта, т. е. пере смотром границы, разделяющей объект и среду. При этом сле дует учитывать, что расширение объекта приводит, как правило, к повышению его управляемости, но требует дополнительных ресурсов для реализации управления (т. е. последующего струк турного и параметрического синтеза). Разные варианты расши рения объекта квалифицируются различным образом по управ ляемости и требуемому ресурсу управления. Выбор наилучшего варианта объекта в процессе управления им и составляет основу адаптации объекта.

4. Адаптация целей управления. Наконец, если и эта мера неэффективна, следует обратиться к адаптации целей управления (стрелка 5 на рис. 1.1.4). В этом случае определяется новое мно жество целей $\{Z^*\}'$, достижение которых обеспечивается создан ной системой управления. Ввиду того что объект эволюционирует (вместе со средой), изменяется и множество достигаемых им целей. Важно знать, какие именно цели могут быть поставлены перед системой управления. Такую информацию можно получить путем адаптации целей. В результате этого процесса фактически адаптируется субъект, который изменяет свои потребности так, чтобы они удовлетворялись путем реализации нового множества целей, достигаемых системой управления в данный период вре мени. Поэтому адаптацию целей следует считать *адаптацией потребностей* субъекта, пользующегося услугами созданной си стемы управления и поставленного перед необходимостью такой адаптации.

Как видно, все указанные выше четыре уровня адаптации системы управления решают одну и ту же задачу — обеспечение достижения системой поставленных целей.

Каждый последующий уровень адаптации имеет постоянную времени на несколько порядков выше, чем предыдущий, т. е. работает значительно медленнее. Это обстоятельство следует учи-тывать при создании системы адаптации: верхние уровни адаптации должны включаться лишь в том случае, если нижние не могут эффективно отследить изменения, произошедшие в объекте.

В заключение отметим, что перечисленные уровни адаптации используют алгоритмы, которые будут рассмотрены ниже, если, разумеется, эти уровни формализуемы. Неформализуемые уровни адаптации реализуются человеком, который при этом может при-менять приемы и подходы, близкие к рассматриваемым.

§ 1.2. Постановка задачи адаптации

Рассмотрим задачу адаптации объекта F^0 (рис. 1.2.1) как задачу управления. Объект погружен в среду, состояние которой X влияет на его состояние Y, Кроме того, состояние Y объекта может изменяться с помощью его адаптируемых факторов U:

$$U = (u_1, ..., u_q). \qquad (1.2.1)$$

Цель Z^* адаптации определяет требования к критериям, заданным на состоянии Y объекта. Эти требования могут иметь троякую структуру аналогично (1.1.14).

1 . Критерии-неравенства:

$$H(U) = (h_1(U),..., h_p(U)) \geq 0. \qquad (1.2.2)$$

2. Критерии-равенства:

$$G(U) = (g_1(U),..., g_s(U)) = 0. \qquad (1.2.3)$$

3. Минимизируемые критерии:

$$Q(U) = (q_1(U), ..., q_l(U)) \rightarrow \min, \qquad (1.2.4)$$

где

$$h_i(U) = M_x h'_i(Y) = M_x h'_i(F^0(X, U)) \quad (i = 1, \ldots, p); \qquad (1.2.5)$$

$$g_j(U) = M_x g'_j(Y) = M_x g'_j(F^0(X, U)) \quad (j = 1, \ldots, s); \qquad (1.2.6)$$

$$g_k(U) = M_x q'_k(Y) = M_x q'_k(F^0(X, U)) \quad (k = 1, \ldots, l); \qquad (1.2.7)$$

M_x — оператор осреднения по X:

$$M_x(\cdot) = \int (\cdot)\, p(X)\, dX, \qquad (1.2.8)$$

где $p(X)$ — плотность распределения состояний X среды.

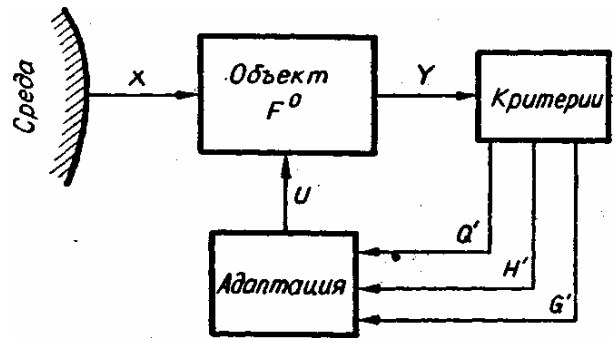


Рис. 1.2.1. Блок-схема адаптации объекта.

Очевидно, что критерии адаптации представляют собой функционалы, определенные на состояниях У объекта как средние значения функций $h'_i(\cdot)$, $g'_j(\cdot)$ и $q'_k(\cdot)$, которые должны быть заданы.

Цель адаптации заключается в решении задачи

$$Q(U) \to \min_{U \in S} \Rightarrow U^*, \tag{1.2.9}$$

где

$$S: \begin{cases} H(U) \geqslant 0; \\ G(U) = 0. \end{cases} \tag{1.2.10}$$

Задачи такого рода при заданных модели F объекта F^0 и распределении $p(X)$ называют задачами стохастического программирования [248], которые отличаются тем, что минимизируемые функционалы и функционалы ограничений являются стохастическими, т. е. математическими ожиданиями. Трудность решения этих задач очевидна.

Однако задачи адаптации осложняются еще и тем, что нет модели объекта F и отсутствует информация о распределении $p(X)$ состояний среды X. Более того, эти факторы изменяются во времени непредсказуемым образом.

Следовательно, даже располагая алгоритмом φ решения задачи (1.2.9.) с ограничениями (1.1.10) в виде

$$U^* = \varphi\,(F,\, p\,(X)\,). \tag{1.2.11}$$

нельзя считать задачу адаптации решенной. Необходимо еще идентифицировать изменяющийся объект F^0 и статистические свойства среды, чтобы получить

$$W = V\,(Ft,\, Pt\,(X)\,), \tag{1.2.12}$$

где F_t — модель объекта, а $p_t\,(X)$ — плотность распределения X. Эти обстоятельства заставляют отказаться от попыток искать алгоритм φ решения поставленной задачи адаптации, так как для сложного объекта адаптации зависимости F_t и $p_t(X)$ весьма сложны, что исключает даже определение функционалов. (1.2.2) — (1.2.4), образующих решаемую задачу (1.2.9).

Именно поэтому для решения задачи (1.2.9) приходится обращаться к алгоритмам адаптации, использующим лишь значения функций $h'_i(\cdot)$, $g'_j(\cdot)$ и $q'_k(\cdot)$ (i=1,...,s; j=1,...,s; k= =1,...,l) в определенные моменты времени, т. е. в общем случае

$$U_{N+1} = \psi\,(\vec{U}_N,\, \vec{H'}_N,\, \vec{G'}_N,\, \vec{Q'}_N), \tag{1.2.13}$$

где ψ — алгоритм рекуррентной адаптации;

(1.2.14)

$$\vec{U}_N = (U_N, U_{N-1}, \ldots, U_{N-W});$$

$$\vec{H'}_N = (H'_N, H'_{N-1}, \ldots, H'_{N-W});$$

$$\vec{G'}_N = (G'_N, G'_{N-1}, \ldots, G'_{N-W});$$

$$\vec{Q'} = (Q'_N, Q'_{N-1}, \ldots, Q'_{N-W}),$$

а U_N, H'_N, G'_N, Q'_N — значения адаптируемых параметров и измеренные значения функций:

$$H' = (h'_1, \ldots, h'_p);$$

$$G' = (g'_1, \ldots, g'_s);$$

$$Q' = (q'_1, \ldots, q'_l)$$

(1.2.15)

в N-й момент времени; W — глубина памяти алгоритма адаптации ψ.

Синтез конкретных алгоритмов адаптации ψ для решения различных задач будет рассмотрен в главах 4—6; Здесь же введем некоторые упрощения общей постановки задачи адаптации (1.2.9).

Прежде всего будем предполагать, что задача однокритери-.альна, т. е. имеется свертка критериев (1.2.4), что позволяет считать $l = 1$, а $Q'(\cdot)$ — скалярной функцией.

Далее, будем рассматривать лишь дискретные моменты времени $N = 0,1,...$, когда измеряются критерии и изменяются адаптируемые параметры объекта.

И наконец, будем предполагать, что состояния X среды образуют во времени случайный процесс типа белого шума. Это приводит к тому, что функционалы (1.2.5) — (1.2.7) связаны со значениями функций (1.2.15) следующими довольно очевидными соотношениями:

$$h'_i(F^0(X, U)) = h_i(U) + \varepsilon_i;$$

$$g'_j(F^0(X, U)) = g_j(U) + \varepsilon_j;$$

$$Q'(F^0(X, U)) = Q(U) + \varepsilon_k,$$

где ε_i, ε_j и ε_k — реализации нормальных некоррелированных во времени случайных величин с нулевыми математическими ожиданиями и некоторыми дисперсиями:

$$\varepsilon_i = N(0, \sigma_{h_i}^2);$$

$$\varepsilon_j = N(0, \sigma_{g_j}^2); \quad (1.2.16)$$

$$\varepsilon_k = N(0, \sigma_{Q_k}^2).$$

Теперь рассмотрим наиболее интересные частные случаи задачи адаптации (1.2.9).

1. $S \in R^q$, т. е. задача не имеет ограничений ($p = s = 0$):

$$Q(U) \to \min_{U \in R^q}. \tag{1.2.17}$$

Это эквивалентно решению задачи оптимизации в обстановке помех при случайно блуждающей точке экстремума по наблюдениям $Q'(U) = Q(U) + \varepsilon$, где ε — случайная помеха вида (1.2.16).

2. $l = s = 0$, т. е. задача решения системы неравенств в обстановке помех:

$$H = (U) \geq 0. \tag{1.2.18}$$

Эта задача сводится к предыдущей (1.2.17) следующим простым преобразованием:

$$Q'(U) = -\sum_{i=1}^{p} a_i \min [h'_i(U), 0], \tag{1.2.19}$$

где a_i — значимость i-го ограничения.

3. $p = l = 0$, $s = q$, т. е. задача решения системы уравнений

$$G(U) = 0 \tag{1.2.20}$$

в обстановке помех. Эта задача сводится к (1.2.17) следующим образом:

$$Q'(U) = \sum_{j=1}^{s} b_j f(g'_j(U)), \tag{1.2.21}$$

где $f(\bullet)$ — неотрицательная четная унимодальная функция, например $(\bullet)^2$; b_j — вес j-го ограничения.

Аналогичным образом можно свести к (1.2.17) и другие задачи, которые получаются из (1.2.9). Так, сама задача (1.2.9) сводится к (1.2.17) следующим образом:

$$\tilde{Q}'(U) = Q'(U) - \sum_{i=1}^{p} a_i \min [h'_i(U), 0] + \sum_{j=1}^{s} b_j f(g'_j(U)), \tag{1.2.22}$$

т. е. путем введения соответствующих штрафов.

Таким образом, задача адаптации всегда с помощью свертки (1.2.22) сводится к задаче (1.2.17) оптимизации в обстановке

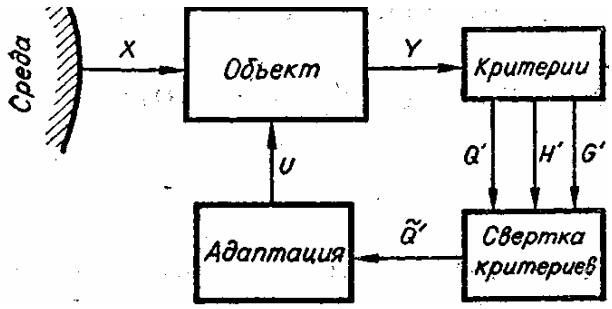


Рис. 1.2.2. Блок-схема адаптации объекта с использованием свертки критериев.

случайных помех и/или при блуждающем экстремуме. На рис. 1.2.2 изображена блок-схема решения задачи адаптации с использованием свертки критериев, что эквивалентно применению метода штрафных функций.

§ 1.3. Адаптация и оптимизация

Как показано выше, задача адаптации возникает в том случае, если отсутствует информация, необходимая для оптимизации объекта. Однако задачи адаптации и оптимизации тесно взаимосвязаны. Покажем это.

Пусть необходимо оптимизировать функционал Q, характеризующий эффективность функционирования объекта в среде X, статистические свойства которой $p(X)$ известны. Тогда функционал Q определяется однозначно:

$$Q(U) = \int q(X, Y)\, p(X)\, dX, \qquad (1.3.1)$$

где $q(X, Y)$ — мгновенная оценка эффективности объекта при данных состояниях X среды и Y объекта. Располагая моделью F объекта F^0, связывающей его выход Y с состоянием X среды и оптимизируемыми факторами U объекта F^0 и подставляя $Y = F(X, U)$ в (1.3.1), получаем

$$Q(U) = \int q(X, F(X,U))\, p(X)\, dX \qquad (1.3.2)$$

— выражение для оптимизируемого функционала. На стадии оптимального проектирования, располагая моделью F объекта и моделью статистических свойств $p(X)$ среды, можно решить задачу оптимизации (1.2.9). Однако в реальных условиях при функционировании сложных объектов обычно отсутствует информация о моделях среды и объекта и необходимо решать задачу (1.2.9) путем соответствующей подстройки параметров U объекта, располагая лишь наблюдениями X и Y.

Пусть для простоты объект и наблюдение непрерывны. Оценим в этом случае функционал (1.3.2) на интервале времени [0; T] в процессе функционирования объекта:

$$\hat{Q}(U)=\frac{1}{T}\int_0^T q(X_t, Y_t)\,dt. \qquad (1.3.3)$$

При $T\to\infty$ и неизменных свойствах объекта и среды получаем $\hat{Q}(U)=Q(U)$ и для решения задачи адаптации можем воспользоваться методами оптимизации. Однако временные затраты будут неприемлемы. Эти затраты уменьшаются при уменьшении базы наблюдений Г, но одновременно снижается и эффективность каждого шага, так как оценка (1.3.3) становится очень грубой. При T = 0 получаем адаптацию.

Как видно, адаптация является оптимизацией в обстановке значительных помех, связанных с грубостью оценки функционала (1.3.2). Снижение эффективности адаптации компенсируется ее оперативностью.

На рис. 1.3.1 показаны обе схемы, причем пунктиром обозначена схема оптимизации. Как видно, схемы адаптации и оптимизации отличаются незначительно — базой оценки минимизируемого функционала объекта $Q(U)$.

Таким образом, и на стадии оптимального проектирования, и при адаптации решается одна и та же задача (1.2.9), но с разной исходной информацией. При оптимальном проектировании следует вычислить $Q(U)$, а в процессах адаптации можно ограничиться оценкой $\hat{Q}(U)=q(X_t, Y_t)$. Это обстоятельство позволяет довольно просто строить модели адаптации, которые по сути дела являются моделями оптимизации в обстановке помех. Рассмотрим наиболее характерные модели, используемые ниже.

Пусть $Q(U, U^*)$ — скалярная функция векторного аргумента $U = (u_1,...,u_q)$, экстремум которой расположен в точке U^*. Тогда моделью функции, минимизируемой при адаптации, является следующая зависимость:

$$Q'(U)=a(t)\,Q(U, U^*(t))+\varepsilon(\sigma_t), \qquad (1.3.4)$$

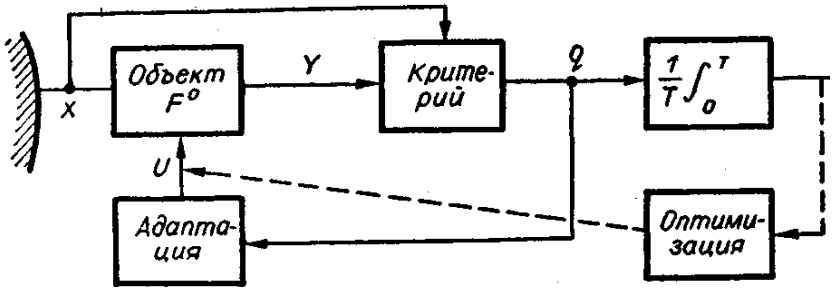


Рис. 1.3.1. Блок-схема адаптации и оптимизации.

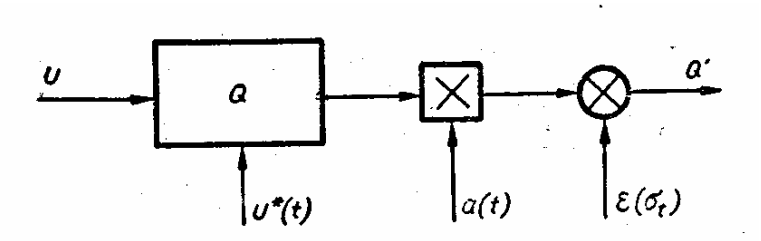


Рис. 1.3.2. Блок-схема модели объекта адаптации.

где $U^*(t)$ — случайный дрейф экстремума U^* в пространстве адаптируемых параметров $\{U\}$; $a(t)$ — случайный дрейф градиента минимизируемой функции; $\varepsilon(\sigma_t)$ — случайная функция, моделирующая помеху с нулевым средним и дисперсией σ_t, зависящей, вообще говоря, от времени. Блок-схема модели (1.3.4) показана на рис. 1.3.2.

Таким образом, модель объекта адаптации задается четверкой

$$\langle a(t), Q(\cdot,\cdot), U^*(t), \sigma_t \rangle. \qquad (1.3.5)$$

Рассмотрим некоторые простейшие модели.

1. Стационарная модель:

$$Q'(U) = f(|U-U^*|) + \varepsilon(\sigma), \qquad (1.3.6)$$

где f — унимодальная функция с минимумом в нуле, например $f(\cdot) = (\cdot)^2$, где U^* и σ неизменны во времени. Это модель со стационарной помехой и без дрейфа экстремума. По мере приближения к экстремуму U^* при $f(\cdot) = (\cdot)^2$ отношение сигнал/шум уменьшается и при $U = U^*$ равно нулю.

Это означает, что трудности продвижения к оптимуму U^* возрастают по мере приближения к нему.

2. Нестационарная модель без помех (дрейф экстремума):

$$Q'(U) = f(|U - U^*(t)|), \qquad (1.3.7)$$

где $U^*(t)$ — модель дрейфа, которая может быть линейной и диффузионной. Линейный дрейф описывается естественным соотношением

$$U^*(t) = U^*_0 + At, \qquad (1.3.8)$$

где U^*_0 — исходное положение экстремума, A — вектор скорости (направления и интенсивности) дрейфа (A принадл. $\{U\}$). Диффузионная модель дрейфа представляется рекуррентным выражением

$$U^*(t+1) = U^*(t) + a\xi, \qquad (1.3.9>$$

которое описывает случайный процесс с независимыми случайными приращениями $a\xi$, где ξ — единичный случайный вектор, равномерно распределенный в пространстве $\{U\}$, a — интенсивность дрейфа.

3. Нестационарная модель (случайный дрейф градиента):

$$Q'(U) = a(t)\, f(\,|U-U^*|\,), \qquad (1.3.10)$$

где $a(t)>0$ — случайная функция, моделирующая дрейф градиента минимизируемого критерия, которая задается своей автокорреляционной функцией.

Аналогично могут быть построены модели адаптации при наличии ограничений.

Следует заметить, что здесь управление U варьируется в пределах заданного множества возможностей S, которое определяется типом задачи. Это множество может быть непрерывным $(S \in R^q)$ или дискретным, конечным $(S \in D)$, что и определяет тип адаптации (см. § 1.5).

§ 1.4. Адаптация и компенсация

Задача управления сложным объектом, рассмотренная в первом параграфе этой главы, редко решается во всей полноте своей сложности. Чаще рассматриваются два ее варианта, которые удобно назвать компенсацией и адаптацией. При этом блок-схема системы управления как бы «разрезается» вертикально на две половины. Левая представляет собой блок-схему компенсации (рис. 1.4.1, *а*), а правая — адаптации (рис. 1.4.1, *б*).

Формальное отличие этих схем заключается в различных источниках информации для управляющего устройства. В случае компенсации исходной информацией для синтеза управления U является только состояние X среды. Для адаптации же управление U синтезируется на основе информации лишь о состоянии Y объекта. Это отличие определяет ряд существенных черт и особенностей процессов компенсации и адаптации, которые целесообразно рассмотреть специально.

Компенсация как управление связана с достижением целей $\{Z^*\}$ при известном состоянии среды. Совершенно очевидно, что

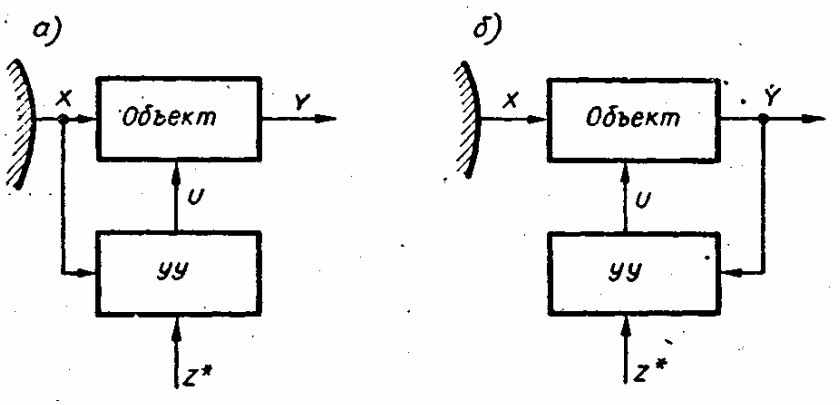


Рис. 1.4.1. Схемы компенсации (а) и адаптации (б).

при отсутствии обратной связи компенсация возможна, если известна модель F объекта, связывающая входы X и U с выходом Y:

$$Y = F(X, U). \tag{1.4.1}$$

При этом для синтеза управления необходимо решить задачу оптимизации, которая получается из формулы (1.1.14) подстановкой выражений (1.1.19) и (1.4.1):

$$\psi_v(X, F(X, U)) \to \min_{U \in S}$$

$$(v = s+p+1, \ldots, s+p+l),$$

1.4.2) где

$$S: \begin{cases} \psi_i(X, F(X, U)) = a_i \quad (i = 1, \ldots, s); \\ \psi_j(X, F(X, U)) \geqslant b_j \quad (j = s+1, \ldots, s+p). \end{cases} \tag{1.4.3}$$

Здесь ψ_v, ψ_i, ψ_j — заданные функционалы, значение которых регламентируется целью Z^* (1.1.13). Как видно, это обычная задача многокритериальной оптимизации. Осуществив свертку критериев, получаем после очевидных преобразований стандартную задачу оптимизации:

$$Q(X, U) \to \min_{U \in S} \Rightarrow U^*_X, \tag{1.4.4}$$

где

$$S: \begin{cases} G(X, U) = 0; \\ H(X, U) \geqslant 0. \end{cases} \tag{1.4.5}$$

Здесь $Q(X, U)$ — скалярная свертка экстремальных критериев,

$$G(X, U) = (g_1(X, U), \ldots, g_s(X, U));$$
$$H(X, U) = (h_1(X, U), \ldots, h_p(X, U)), \tag{1.4.6}$$

причем

$$g_i(X, U) = \psi_i(X, F(X, U)) - a_i \quad (i = 1, \ldots, s);$$
$$h_{j-s}(X, U) = \psi_j(X, F(X, U)) - b_j \quad (j = s+1, \ldots, s+p). \tag{1.4.7}$$

Полученная задача (1.4.4) с заданными функциями Q, G и H является задачей математического программирования. Ее решение U^*_X зависит естественным образом от состояния X среды, которое контролируется при компенсации.

Таким образом, для управления методом компенсации необходимо:

1) знать состояние X среды,
2) иметь модель $F(X, U)$ объекта,
3) уметь решать задачу математического программирования (1.4.4).

Понятие «компенсации» здесь используется потому, что такой тип управления как бы компенсирует влияние среды и позволяет добиваться поставленной цели Z^* независимо от состояния среды и без наблюдения за состоянием Y объекта.

Такое управление иногда называют программным. В этом случае программой является цель Z^*, которая реализуется в процессе компенсации с учетом состояния X среды.

Теперь с этих позиций рассмотрим *адаптацию* (см. рис. 1.4.1, *б*). Здесь задача синтеза управления (1.1.14) записывается в виде (1.4.2), а после преобразования — в форме (1.4.4). Но она не является задачей математического программирования, так как ни состояние X среды, ни модель F объекта неизвестны. Доступны для наблюдения лишь оценки значений функционалов Q, G и H (1.2.15), на которые, вообще говоря, накладываются случайные помехи вида (1.2.16). Располагая этой скудной информацией, необходимо решить задачу (1.4.4).

Такова ситуация, складывающаяся в процессе решения задачи адаптации. Эта ситуация может усложниться тем, что цель — а с нею и структура функционалов Q, G и H — иногда изменяется в соответствии с изменением потребностей субъекта.

Как видно, основное отличие адаптации от компенсации состоит в том, что задача оптимизации (1.4.4) здесь решается по «зашумленным» реализациям функционалов при неизвестной структуре этих функционалов. Этим адаптация обобщает компенсацию, так как позволяет решать и ее задачи, в то время как методами компенсации (т. е. методами математического программирования) нельзя решать задачу адаптации.

Адаптация как специальный процесс управления обладает рядом специфических черт. Прежде всего это стабильный характер целей адаптации. Это означает, что экстремизируемый критерий Q и ограничения S либо не изменяются, либо изменяются, но крайне медленно или редко.

Другой особенностью процесса адаптации является его эффективность лишь для незначительных, эволюционных изменений объекта. Адаптация является мерой, компенсирующей малые изменения объекта.

И наконец, адаптация является средством, с помощью которого удается исправить недостатки проектирования объекта. Действительно, всякое проектирование страдает неточностью в силу недостаточных априорных знаний свойств среды, в которой

предстоит функционировать проектируемому объекту. Уже модель объекта, положенная в основу принятия решения, отличается приближенностью, что заставляет прибегнуть к адаптации. Более того, изменение среды функционирования объекта и целей, реализуемых в процессе функционирования, также вынуждает обращаться к адаптации как способу преодоления этих изменений, с тем чтобы поддерживать значение нового функционала на экстремальном уровне. Наличие адаптации позволяет ослабить требования к процессу проектирования объекта и тем самым упростить и удешевить этот трудоемкий и дорогой процесс.

§ 1.5. Типы адаптации

Как уже говорилось, адаптация является частным случаем управления и заключается в изменении управляемого фактора U таким образом, чтобы поддерживать некие заданные функционалы объекта в требуемом состоянии независимо от действия всякого рода внешних и внутренних воздействий. При этом специфика объекта накладывает на управление требование

$$U \in S, \tag{1.5.1}$$

где S — множество допустимых управлений. Структура этого множества определяется двумя факторами — целевыми ограничениями Z^* и самим объектом. В данном случае нас будет интересовать специфика объекта, так как именно она через структуру множества определяет тип адаптации. Исходя из этого можно классифицировать различные типы адаптации [170].

1.5.1. Классификация типов адаптации

Если объект таков, что его изменение в процессе адаптации удобно осуществлять с помощью параметров $u_1,..., u_n$, т. е.

$$U = (u_1,..., u_n), \tag{1.5.2}$$

то такую адаптацию естественно назвать *параметрической.* Тогда каждый параметр u_i может принимать бесконечное или конечное число значений. В первом случае

$$U \in S \subset R^q, \tag{1.5.3}$$

т. е. множество S является континуумом, во втором

$$U \in S = D, \tag{1.5.4}$$

где D — дискретное множество значений управления U.

Однако очень часто адаптацию объекта удобно осуществлять не путем изменения его параметров, а модификацией его структуры, т. е. вводя *структурную* адаптацию. Для этого представим фактор управления U в виде пары

$$U = \langle W, C \rangle, \qquad (1.5.5)$$

где W — структурные факторы, с помощью которых можно изменять структуру объекта адаптации, а $C = (c_1, ..., c_k)$ — адаптируемые параметры объекта (это параметры (1.5.2), с помощью которых реализуется параметрическая адаптация).

Целенаправленная вариация структурных факторов дает возможность адаптировать структуру объекта.

Такая декомпозиция управления U на структурные W и параметрические C факторы позволяет более эффективно решать задачи адаптации сложных объектов, для которых параметрическая адаптация малоэффективна.

Теперь задачу адаптации (1.2.9) следует записать в виде

$$Q(W, C) \to \min_{W \in E_W} \min_{C \in E_{CW}} \Rightarrow W^*, C^*_{W^*},$$

где E_W — множество допустимых структур W; E_{CW} — множество допустимых параметров С, соответствующих структуре, определяемой W; W^* — оптимальная структура; $C^*_{W^*}$ — оптимальные параметры этой структуры. (Так как W однозначно определяет структуру объекта, то можно W условно называть структурой.) Очевидно, что

$$S = E_W \times E_{CW}, \qquad (1.5.7)$$

т. е. множество S допустимых управлений при адаптации (1.2.10) образуется как произведение множеств допустимых структур E_W и параметров E_{CW} этих структур.

Блок-схема решения задачи (1.5.6) показана на рис. 1.5.1, из которого хорошо виден иерархический характер адаптации. На верхнем уровне производится адаптация структуры W, а на нижнем — параметров С этой структуры. Очевидно, что эти Два контура адаптации Работают в разных временных режимах: темп па-

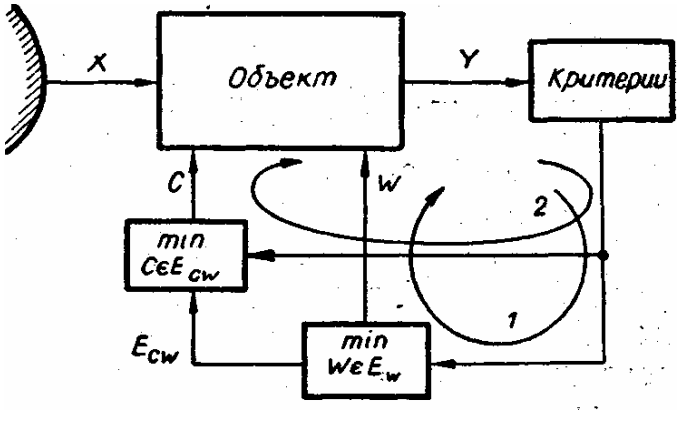


Рис. 1.5.1. Двухконтурная схема структур: ной адаптации.

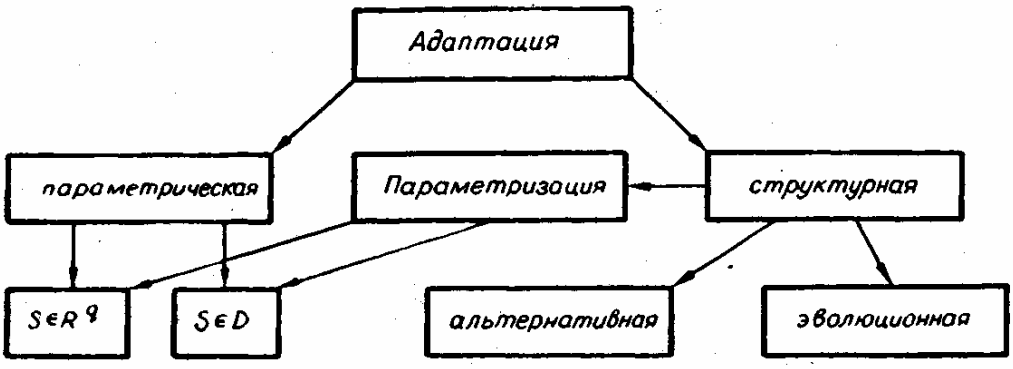


Рис. 1.5.2. Классификация типов адаптации.

раметрической адаптации (контур *2* на рис. 1.5.1) значительно выше темпа структурной (контур *1).* Действительно, на каждый шаг структурных изменений объекта должен приходиться весь цикл параметрической адаптации, иначе не выявится полностью эффективность реализованной структуры.

Очевидно, что методы решения задач структурной и параметрической адаптации различны, что и заставляет обращаться к такой дифференциации. Ее можно продолжить. На рис. 1.5.2 показана схема классификации различных типов адаптации. Прежде всего структурную адаптацию удобно подразделить на альтернативную и эволюционную [113].

Альтернативная адаптация отличается тем, что множество допустимых структур E_W невелико и содержит две — пять альтернативных структур.

Эволюционная адаптация, по сути дела, моделирует процесс биологической эволюции (см. § 3.7 настоящей книги) . Этот алгоритм отличается введением незначительных вариаций структуры δW, моделирующих случайные мутации, которые также незначительно изменяют эффективность Q адаптируемого объекта. Иначе говоря, имеет место соотношение типа неравенства Липшица:

$$| Q(W+\delta W) - Q(W) | \le \mu ||\delta W||, \qquad (1.5.8)$$

где μ = const, а под нормой вариации структуры $||\delta W||$ следует понимать число, характеризующее степень изменения структуры этой вариацией δW. Например, при графическом описании структуры W нормой такой вариации может быть число новых введенных или «выброшенных» вершин или ребер графа, описывающего структуру адаптируемого объекта.

«Мутации» структуры δW и правило отбора, позволяющее выявлять ее благоприятные вариации, и образуют механизм эво-

люции, с помощью которого строится последовательность улучшающихся структур

$$W_0 \rightarrow W_1 \rightarrow \ldots \rightarrow W_N \rightarrow W_{N+1} \rightarrow \ldots, \qquad (1.5.9)$$

обладающих свойством

$$W_N \succ W_{N-1} \quad (N = 1, \ldots), \qquad (1.5.10)$$

где знак предпочтения $\succ$ имеет очевидный смысл:

$$Q(W_N) < Q(W_{N-1}). \qquad (1.5.11)$$

Заметим, что иногда допустимы нарушения соотношения (1.5.11) — например, в случае, когда вариацией структуры W_i не удается получить лучшую, чем W_i, по критерию (1.5.11). Тогда выбирают лучшую из «мутированных» структур, что нарушает (1.5.11), но обеспечивает процедуре эволюции глобальные свойства, очень ценные в задачах структурной адаптации.

Очевидно, что такой новый тип адаптации, как структурная, требует разработки новых подходов к решению задач. Однако не следует забывать, что существует мощный аппарат параметрической адаптации, который можно использовать и для решения задач структурной. Для этого достаточно параметризовать структуру адаптируемого объекта (см. рис. 1.5.2).

1.5.2. Параметризация структуры объекта

Простейшим способом параметризации структуры является введение двоичных переменных

$$u_i \in \{0; 1\}, \qquad (1.5.12)$$

однозначно характеризующих структуру адаптируемого объекта. Это всегда можно сделать, и тогда полученный двоичный вектор (1.5.2) является кодом структуры объекта. Например, в случае альтернативной адаптации, когда множество допустимых структур мало:

$$W \in \{W_1, \ldots, W_q\} = E_W, \qquad (1.5.13)$$

задача структурной адаптации сводится к следующей оптимизационной задаче с двоичными переменными:

$$Q(W) = Q\left(\sum_{i=1}^{q} W_i u_i\right) \rightarrow \min_{U \in S}, \qquad (1.5.14)$$

где

$$S:\begin{cases}\sum_{i=1}^{q} u_i = 1; \\ u_i \in \{0;\ 1\} \quad (i=1, \ldots, q).\end{cases} \quad (1.5.15)$$

Аналогично параметризуются задачу эволюционной адаптации. Например, структура, представленная графом с фиксированным числом (n) вершин, кодируется двоичным вектором

$$U = (u_{11}, \ldots_r u_{1n}, u_{2n}, \ldots, u_{nn}) \quad (1.5.16)$$

компоненты которого определяют наличие (1) или отсутствие (0) соответствующего ребра графа:

$$u_{ij} \in \{0;\ 1\} \quad (i, j=1, \ldots, n), \quad (1.5.17)$$

и задача адаптации сводится к следующей задаче бинарного программирования:

$$Q(U) \to \min_{U \in S}, \quad (1.5.18)$$

где

$$S:\begin{cases}H(U) \geqslant 0; \\ G(U) = 0; \\ u_{ij} \in \{0;\ 1\};\end{cases} \quad (1.5.19)$$

а функционалы H и G определены выше (см. § 1,2) на графе U.

Как видно, полученные задачи являются параметрическими с дискретным множеством допустимых параметров (1.5.4). Подходи к решению задач такого рода рассмотрены в § 3.4.

Эти задачи можно свести к непрерывным $(S \in R^q)$ путем введения штрафной функции:

$$\tilde{Q}(U) = Q(U) + \nu \sum_{i=1}^{q} u_i (1 - a_i) \to \min_{U \in S'}, \quad (1.5.20)$$

где $\upsilon>0$ — коэффициент штрафа, а множество S' уже непрерывно и получается из (1.5.19) путем исключения требования двоичности (1.5.17) вектора U.

Другим способом сведения дискретной задачи к непрерывной является **рандомизация**, в соответствии с которой вводится непрерывный вектор вероятностей

$$P = (p_1, \ldots, p_q), \quad (1.5.21)$$

в котором

$$p_i = \text{Вер}\ (u_i=1). \tag{1.5.22}$$

Для задачи альтернативной адаптации следует добавить условие

$$\sum_{i=1}^{q} p_i = 1, \tag{1.5.23}$$

реализующее лишь одну альтернативу.

Критерий $Q(U)$ в этом случае заменяется на его среднее значение при заданном P:

$$\bar{Q}(P) = \sum_{1}^{2^q} Q(u_1, \ldots, u_q) \prod_{j=1}^{q} p_j^{u_j}, \tag{1.5.24}$$

где суммирование проводится по всем вариантам двоичного вектора U и введены обозначения

$$p_j^0 = 1 - p_j; \quad p_j^1 = p_j. \tag{1.5.25}$$

Функционал (1.5.24) называют рандомизированным или сглаженным [96] функционалом $Q(U)$. Действительно, если $Q(U)$ определен лишь в вершинах q-мерного гиперкуба, то $Q(P)$ — во всем этом гиперкубе, причем в вершинах оба функционала совпадают.

Следовательно, естественно считать $\bar{Q}(P)$ сглаживанием $Q(U)$ путем введения рандомизации, характеризующейся непрерывным вектором P.

Решение непрерывной задачи оптимизации

$$\bar{Q}(P) \to \min_{P} \Rightarrow P^* \tag{1.5.26}$$

дает возможность определить P^*. Это решение, как легко заметить, состоит из нулей и единиц и совпадает с искомым U^*, т. е.

$$P^* = U^*. \tag{1.5.27}$$

Процесс поиска P^* производится с помощью оценок значений минимизируемого функционала (1.5.24) методом Монте-Карло (см. § 3.2):

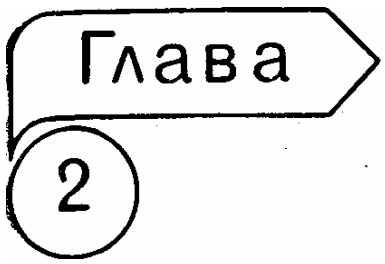
$$\hat{\bar{Q}}(P) = \frac{1}{N} \sum_{i=1}^{N} Q(U_i), \tag{1.5.28}$$

где U_i — i-я случайная реализация двоичного вектора (1.5.2), имеющего вероятностные свойства P.

Такой подход позволяет эффективно решать многие задачи с двоичными переменными [52, 74, 96, 98, 99, 240]. Однако получаемые при этом задачи имеют, как правило, многоэкстремальный характер, что, безусловно, значительно затрудняет их решение. Это обстоятельство и заставляет обращаться к изысканию новых путей решения задач структурной адаптации, минуя этап параметризации. Такие новые подходы рассмотрены в главах 5 и 6 настоящей книги.

Резюмируя содержание первой главы, автор должен признать, что постановку задачи адаптации как задачи минимизации функционала по наблюдениям его приближенных оценок нельзя считать удачной с инженерной точки зрения. Она очень плодотворна для математиков, занимающихся проблемой оптимизации, так как дает возможность эффективно использовать мощный математический аппарат случайных процессов и досконально изучить вопросы сходимости различных процедур адаптации. Однако инженеру чужда асимптотика итерационных процессов адаптации, и для него даже не очень важно, что в среднем он поступает оптимально. Инженеру нужна гарантия, что адаптация сделает данный объект лучше или во всяком случае не хуже того, чем он был без адаптации. А именно такой гарантии описанная выше постановка задачи адаптации не дает.

Здесь нужно искать новые пути постановки задач и их решения. По-видимому, наиболее перспективным следует считать бионический путь. Биологические системы великолепно адаптируются, не пользуясь понятием экстремума, которое сразу привязывает проблему адаптации к вариационным проблемам вычислительной математики, что дает в руки инженера аппарат исследования, но не управления объектом адаптации.



Объекты адаптации

Если бы господь бог сделал мне честь спросить мое мнение при сотворении мира, я бы ему посоветовал сотворить его получше, а главное — попроще.

Альфонс X Кастильский (XIII в.)

Залогом эффективного управления является хорошая модель объекта. В этой главе рассматриваются черты сложных объектов адаптации, которые иллюстрируются на нескольких конкретных примерах. Одним из наиболее важных и интересных для нас объектов является вычислительная система (ВС) с многочисленными задачами адаптации на разных уровнях ее организации. В качестве других примеров рассмотрены процессы обучения, сегментация графа и выбор дисциплины обслуживания в системах массового обслуживания.

§ 2.1. Сложная система как объект адаптации

Понятия «сложная система» и «сложный объект управления» до настоящего времени не имеют строгого определения. Возможно, что такое определение никогда и не появится, поскольку в нем, по сути дела, нет острой необходимости. Интуитивное представление о сложной системе довольно точно соответствует тому, которое используется в теории управления. Это представление носит неформальный характер. Однако некоторые черты (но не формальные признаки) сложной системы можно привести. Рассмотрим основные из них.

1 . Отсутствие необходимого математического описания явля ется обязательной чертой сложного объекта управления. Под математическим описанием подразумевается наличие алгоритма F вычисления состояния объекта Y по наблюдениям его вхо дов — управляемого U и неуправляемого X, т. е. $Y = F(X, U)$.

2. «Зашумленность» сложных систем — также очень важная черта, характеризующая трудность процессов анализа и управ ления ими. Зашумленность обусловлена не столько наличием каких-то специальных генераторов случайных помех, сколько сложностью объекта и вытекающим из нее неизбежным обилием

всякого рода второстепенных (с точки зрения целей управления, разумеется) процессов. Поэтому поведение объекта зачастую оказывается неожиданным для исследователя, причем эту неожиданность удобнее рассматривать как случайный фактор, как зашумленность, чем разбираться в механизме второстепенных процессов, протекающих в сложной системе и порождающих неожиданность ее поведения. Любой сложный объект имеет много такого рода «неожиданностей», которые являются свидетельством его сложности. Таковы биологические, социальные, технологические системы и многие другие.

3. «Нетерпимость» к управлению является, пожалуй, самой досадной чертой сложной системы. Дело в том, что она сущест вует, грубо говоря, вовсе не для того, чтобы ею управляли. Она «не любит» управления по причине «независимости» своего су ществования от целей субъекта, желающего управлять ею. Трудно рассчитывать на то, что «собственные» цели сложной системы совпадут с целями управления. Скорее они будут про тиворечить друг другу. Это и вызывает «негативную» реакцию сложной системы на управление, цель которого с ней «не согла сована».

Заметим, что если сложная система обладает активностью, например, содержит в себе людей или их коллективы, то кавычки в предыдущем абзаце можно снять.

4. Нестационарность сложной системы естественно следует из ее сложности. Нестационарность проявляется в дрейфе харак теристик системы, в изменении ее параметров, в эволюции слож ной системы во времени. Чем сложнее система, тем более рель ефно проявляется эта ее черта, что создает серьезные трудности при создании модели сложной системы и управлении ею.

5. Невоспроизводимость экспериментов со сложной системой также является ее важной чертой. Она связана прежде всего с зашумленностью и нестационарностью сложной системы. Прояв ляется эта черта в различной реакции системы на одну и ту же ситуацию или управление в различные моменты времени. Слож ная система как бы все время перестает быть сама собой. Эта черта накладывает специальные требования на процессы син теза и коррекции модели системы. Перечень особенностей слож ной системы можно было бы продолжить. Однако в любом случае следует помнить, что мы называем лишь черты, свойственные сложной системе, но ни в коей мере не формальные признаки. Отсутствие одной или даже нескольких из указанных черт вовсе необязательно делает систему простой. Эта неформальная харак теристика, отличаясь размытостью и приближенностью, позво ляет тем не менее в какой-то мере описать сложную систему как: объект управления вообще и адаптации в частности.

Теперь рассмотрим несколько конкретных примеров сложных объектов, процедуры адаптации которых описаны в последующих главах.

§ 2.2. Вычислительная система как объект адаптации

Как известно, задача адаптации возникает при наличии в объекте или среде неопределенности, изменяющей его функционирование. ВС является именно таким объектом, внешняя и внутренняя среды которого интенсивно изменяются, что не может не отражаться на эффективности его функционирования. Дестабилизирующими факторами ВС являются потоки заявок на обработку информации (их интенсивность и требуемые ресурсы), помехи в каналах связи ЭВМ, ненадежность отдельных элементов ВС и др.

Если бы мы располагали информацией о состоянии этих факторов и, кроме того, имели бы модель ВС, то, решив соответствующую оптимизационную задачу, можно было бы определить, какие меры следует принять, чтобы вернуть ВС в оптимальное по заданным критериям состояние. Совокупность таких мер и есть управление ВС, направленное на компенсацию дестабилизирующих факторов.

Однако состояние дестабилизирующих факторов, как правило, неизвестно, что практически исключает применение методов оптимального управления для объектов типа ВС и неизбежно, приводит к адаптации [292].

2.2.1. Специфика вычислительных систем

Рассмотрим специфические черты ВС как объекта адаптации. Можно выделить четыре такие черты.

1. Существенная зависимость ВС от среды. ВС как объект адаптации связана со средой, которую можно представить в виде потока решаемых задач. Факторы дрейфа характеристик системы, ее эволюция, а также поток неисправностей также могут быть отнесены к средовым факторам. Поток решаемых задач определяет режим работы ВС, так как именно для его обслуживания эта система и создана. В указанном смысле ВС является обычной системой массового обслуживания, удовлетворяющей (или отклоняющей) поступающие заявки. Очевидно, что специфика заявок должна быть отражена в структуре ВС, иначе она будет работать неоптимально. Более того, поскольку единственной целью ВС является наилучшее удовлетворение заявок, или

эффективное решение поступающих задач, то ВС всегда должна находиться в строгом соответствии с требованиями среды.

Это обстоятельство отличает ВС (как, впрочем, и другие системы массового обслуживания) от других адаптируемых систем, где влияние среды хотя и играет какую-то роль, но не определяющую, являясь лишь досадной помехой, которую должна компенсировать адаптация.

Таким образом, зависимость ВС от средовых воздействий решаемых задач значительно больше, чем обычных, традиционных адаптируемых систем, и поэтому она больше других нуждается в адаптации.

2. Значительная априорная неопределенность среды ВС. Информация о специфике потока задач, как правило, очень трудно предвидится и выявляется не заранее, а во время «обслуживания» — особенно в ВС коллективного пользования, потребителями которой являются научные работники.

Спектр потребностей пользователей настолько широк, а неопределенность временных затрат машинного времени так велика, что построение статистических моделей такого рода пользователей практически не представляется возможным.

Действительно, даже при наличии информации об используемых ресурсах ВС (системных, сервисных, программных и аппаратных) потребляемое для решения задач время всегда неопределенно, так как пока не существует возможности достаточно точно оценить по программе, какое машинное время понадобится для решения той или иной задачи. Благодаря этому создается ситуация, в которой априорная неопределенность среды ВС всегда оказывается очень большой.

3. Нестационарность среды ВС является существенным фактором, который требует введения адаптации. Эта нестационарность вызывается по крайней мере тремя обстоятельствами:

1) изменением вероятностных свойств потока решаемых задач, неизбежным в связи с эволюцией потребностей среды (изменение старых и появление новых задач и т. д.);

2) эволюцией самой ВС в связи с ее ростом и модернизацией;

3) амортизацией ВС и нестационарностью потока неисправностей в ВС как сложной системе.

Эти обстоятельства приводят к тому, что поведение среды ВС — внешней и внутренней — имеет ярко выраженный нестационарный характер, что делает адаптацию совершенно необходимой мерой по поддержанию ВС в оптимальном соответствии со средой.

4. Наконец, существенная черта ВС как объекта адаптации — **необходимость вариации ее структуры**. Это, разумеется, не исключает адаптации параметров ВС, например, при вариации ее дисциплин обслуживания. Однако структурные вариации при

адаптации ВС обладают значительно большим эффектом, чем параметрические. Дело здесь в том, что в ВС «гораздо больше структуры, чем параметров», в силу чего структурная адаптация почти всегда более эффективна, чем параметрическая. Это и обусловливает развитие структурной адаптации как нового эффективного способа адаптации ВС.

Таким образом, ВС как предмет адаптации представляется чрезвычайно сложным, важным и интересным объектом. Здесь введение адаптации связано с жестокой необходимостью, а не с модой. Не будет преувеличением сказать, что большие ВС попросту не смогут существовать без мощной поддержки алгоритмов адаптации, поддерживающих ВС в оптимальном состоянии независимо от изменений среды и самой ВС.

2.2.2. Уровни адаптации вычислительной системы

ВС как объекты управления вообще и адаптации в частности рассматривались мало. Причинами этого, по-видимому, являются новизна, большая сложность такого рода систем и — как следствие — отсутствие хороших моделей. Управление здесь в настоящее время сводится к выявлению критических ситуаций к выработке правил принятия оптимальных решений в этих ситуациях.

Такой подход требует сбора значительной информации о сложившейся ситуации, что далеко не всегда удается сделать, и наличия модели объекта, позволяющей построить оптимальное-решающее правило при заданном критерии функционирования. Именно поэтому пока реализуются лишь простейшие схемы управления ВС — типа программного управления.

В настоящее время назрела необходимость в организации более тонких подходов к управлению многомашинной ВС. Таким подходом и является адаптация.

Как всякое управление, адаптацию следует рассматривать с позиций функционирования объекта адаптации. Рассмотрим задачи адаптации ВС с этой точки зрения.

Едва ли целесообразно сейчас рассматривать задачу глобальной адаптации всей ВС по какой-то даже весьма полной системе критериев эффективности ее функционирования. По-видимому, правильнее будет сначала рассмотреть локальные задачи адаптации, охватывающие лишь отдельные аспекты функционирования ВС (например, по разным критериям) и лишь потом-сформировать из этих подсистем некую — скорее всего, иерархическую — систему адаптации, позволяющую оперативно приспосабливать ВС к новым состояниям среды. Такие подсистемы адаптации ВС могут соответствовать следующим уровням:

1 . Аппаратурный уровень, предполагающий адаптацию уста вок, параметров и структуры аппаратурных средств ВС (напри мер, параметров фильтров в каналах связи машин и т. д.).

2. Алгоритмический уровень адаптации ВС связан с адапта цией алгоритмов обработки информации с целью учета специ фики решаемых в системе задач (например, при адаптации алгоритмов оптимизации и т. д.).

3. Программный уровень адаптации должен осуществлять процесс адаптации при выборе программ работы ВС из альтер нативного множества (например, при выборе программы сорти ровки для обработки данного массива или программы опти мизации).

4. Системный уровень адаптации призван улучшать функцио нирование системного обеспечения ВС (например, адаптация дисциплин обслуживания, распределения памяти и т. д.).

5. Сетевой уровень адаптации связан с адаптацией процессов передачи информации по ВС (например, при определении марш рута сообщения).

Конечно, указанные уровни не исчерпывают всех «этажей» адаптации, которые встречаются или целесообразны в ВС. Возможна иерархия и по другим признакам, например по этапам управления ВС как сложным объектам (см. §1.1).

Ниже будут рассмотрены второй, третий, четвертый и пятый уровни адаптации ВС, что связано с областью интересов и дея-тельности автора настоящей книги.

О сетевом уровне адаптации см. также работы [202; 203, с. 221—225; 299; 300].

Рассмотрим несколько конкретных задач адаптации, возни-кающих на различных уровнях ВС.

2.2.3. Алгоритмический уровень адаптации вычислительной системы

Здесь речь идет об адаптации алгоритмов обработки инфор-мации в ВС. Цель такой адаптации — приспособить алгоритмы к специфике решаемых задач.

Критериями эффективности обычно являются среднее время решения задачи, точность этого решения, объем занимаемой памяти в ВС, вероятность ошибки и т. д.

Специфика решаемых задач проявляется в виде статистически устойчивой в течение некоторого времени особенности этих задач, учет которой позволяет повысить эффективность функционирования ВС по выбранному критерию. Указанная особенность не идентифицируется, т. е. не определяется в явной форме, а учиты-вается в процессе адаптации.

Построим модель этого явления. Пусть

$$P_i = P(\Omega_i, \omega_i) \qquad (2.2.1)$$

— решаемые ВС задачи, отличающиеся структурой (Ω_i) и параметрами (ω_i). Рассмотрим задачи одной структуры Ω:

$$P_i = P(\Omega_i, \omega_i). \qquad (2.2.2)$$

Пусть

$$q_i = q\,(P_i, C) \qquad (2.2.3)$$

— значение критерия эффективности решения задачи P_i алгоритмом, имеющим параметры $C = (c_1,..., c_k)$.

Для выбора оптимального значения этих параметров необходимо осреднить критерии (2.2.3) по всем решаемым задачам со структурой. Ω (другая структура задачи требует иного алгоритма, ее решения):

$$Q(C) = \int q(P, C)\, p(P)\, dP, \qquad (2.2.4)$$

где $p(P)$ — плотность распределения решаемых задач. Очевидно, что такой информации практически никогда не бывает в ВС. Это обстоятельство заставляет обращаться к адаптации параметров алгоритма по наблюдениям отдельных значений критерия (2.2.3), появляющихся при решении различных задач.

Адаптация параметров алгоритма решения задачи возможна не только от задачи к задаче, но и в процессе решения одной задачи, что применимо для рекуррентных методов, когда возможна оценка эффективности одного цикла — например, в процессах поисковой оптимизации. Покажем это.

Алгоритм оптимизации, т. е. алгоритм решения задачи

$$Q(X) \to \min_{x \in S} \Rightarrow X^*,$$

представляет собой оператор F перехода от одного состояния X_N оптимизируемого объекта к другому X_{N+1}, более предпочтительному по заданному критерию Q:

$$X_{N+1} = X_N + \Delta X_{N+1},$$

где $\Delta X_N = F\,(X_N, C)$, а C — параметры оператора оптимизации. Очевидно, что параметры C должны изменяться так, чтобы процесс оптимизации протекал наиболее успешно. Иначе говоря, необходима адаптация этих параметров:

$$C_{N+1} = C_N + \Delta C_{N+1},$$

где $\Delta C_{N+1} = \varphi(C_N, X_N)$ — алгоритм адаптации. Такого рода параметрическая адаптация алгоритмов случайного поиска описана в следующей главе (§ 3.5), а подробнее — в работе [182].

Однако адаптация алгоритма может иметь и структурный характер. Это означает, что нужно уметь переходить от одного альтернативного алгоритма поиска к другому, с тем чтобы все время поддерживать в процессе оптимизации тот алгоритм, который наилучшим образом осуществляет решение данной задачи.

Такие алгоритмы альтернативной адаптации лежат в основе адаптивного пакета программ оптимизации, рассмотренного в пятой главе (§ 5.3).

2.2.4. Программная адаптация вычислительной системы

Программная адаптация связана с манипулированием отдельными программами и выбором той, которая работает наилучшим образом в смысле заданного критерия. Алгоритмически программная адаптация реализуется в виде альтернативного варианта структурной адаптации. Формализуем эту задачу.

Пусть имеются альтернативные программы

$$A_1, \ldots, A_m, \tag{2.2.5}$$

решающие определенный класс Ω задач (2.2.2) в ВС. Как и выше, определен критерий эффективности решения конкретной задачи P_j *с* помощью *i*-й программы:

$$q_{ij} = q\,(P_j, A_i). \tag{2.2.6}$$

Это означает, что, решив задачу, всегда можно оценить критерий апостериорно; априорных соображений обычно не имеется.

Задача состоит в том, чтобы для имеющегося потока задач P выбрать программу, минимизирующую интегральный критерий

$$Q(A) = \int q(P, A)\, p(P)\, dP, \tag{2.2.7}$$

т. е. решить задачу

$$Q(A) \to \min_{A_i (i=1,\ldots,m)} \Rightarrow A^*, \tag{2.2.8}$$

используя только наблюдения значений критерия (2.2.6).

Эту задачу следует решать методами альтернативной адаптации, т. е. так переходить от одной программы (2.2.5) к другой, чтобы процесс привел к программе A^*, оптимальной по выбранному критерию (2.2.7) в сложившейся ситуации.

Рассмотрим в качестве примеров две конкретные задачи программной адаптации.

2.2.4.1. Адаптивный выбор программы сортировки массивов

Задача сортировки поступающего массива, т. е. упорядочения его элементов, является одной из наиболее распространенных задач в ВС. Существует несколько методов сортировки, реализованных в виде программ (2.2.5), которыми располагает ВС. Критерием эффективности (2.2.6) сортировки массива P_i с помощью алгоритма A_i является, например, время сортировки. Если поток поступающих на сортировку массивов имеет некоторую статистическую устойчивость в виде плотности распределения $p(P)$ и для этого распределения имеется наилучший метод A^* сортировки, то естественно найти такой метод, используя альтернативную адаптацию.

Более того, если плотность распределения $p(P)$ массивов изменилась, что повлекло за собой изменение оптимальной программы A^*, то необходимо в процессе адаптации найти новую программу и перейти к ней. Решение этой задачи рассмотрено в пятой главе (§5.5).

2.2.4.2. Адаптация способов кодирования информации в канале передачи данных вычислительной системы

Известно, что выбор оптимального метода (программы) кодирования по заданному критерию зависит от статистических свойств состояния канала. Пусть P — это состояние. Располагая информацией о $p(P)$, т. е. о статистических свойствах состояния канала связи, можно по заданному критерию (например, среднее время передачи одного бита информации, вероятность ошибки и т. д.) выбрать код A^*, наилучший из альтернативных $A_1, \ldots, A_m$. Так и поступают в теории оптимального кодирования.

Однако свойства $p(P)$ реального канала связи изменяются во времени непредвиденным образом и в широких пределах. Поэтому необходимо эксплуатировать в канале ту программу кодирования, которая экстремизирует выбранный критерий эффективности для имеющегося состояния P канала, и переходить к другой, оптимальной при изменении состояния канала связи. Эта задача решается методами альтернативной адаптации и рассмотрена в пятой главе (§ 5.4).

Рассмотрим теперь системный уровень адаптации в ВС.

2.2.5. Системная адаптация вычислительной системы

Эта адаптация затрачивает системное обеспечение ВС, для того чтобы повысить эффективность функционирования ВС при различных условиях эксплуатации.

Пусть P — условия эксплуатации ВС, U — управляемый фактор ВС, который может изменяться целенаправленно в процессе ее работы, а

$$q = q\,(P,\ U) \qquad (2.2.9)$$

— критерий функционирования ВС, характеризующий эффективность ее работы в условиях P при управлении U. Выражение (2.2.9) задается алгоритмически, т. е. в виде правила оценки эффективности q реальной (а не модельной) ВС при различных P и U. Это обстоятельство исключает возможность вычисления интегрального критерия

$$Q(U) = \int q(P, U)\, p(P)\, dP, \qquad (2.2.10)$$

необходимого для синтеза оптимального управления U^*, решающего задачу

$$Q(U) \to \min_{U \in S} \Rightarrow U^*. \qquad (2.2.11)$$

Здесь $p(P)$ — статистические свойства условий эксплуатации ВС (плотность распределения этих условий), которые обычно неизвестны.

Поэтому задачу поддержания ВС в оптимальном состоянии следует решать методами адаптации, т. е. изменять управляемый фактор U, руководствуясь только наблюдениями значений критерия (2.2.9).

Рассмотрим несколько конкретных задач адаптации ВС на системном уровне.

2.2.5.1. Адаптивная сегментация памяти системного математического обеспечения вычислительной системы

Для нормального функционирования ВС следует вводить в ее оперативную память необходимую рабочую информацию, т. е. программы, данные и т. д. Эта информация расположена во внешней памяти в виде блоков, объединенных в сегменты определенного объема. Вызов информации в оперативную память производится сегментами. Процесс обращения к внешней памяти занимает значительное время, в течение которого процессор не работает (рассматривается однопрограммный режим работы ВС). Отсюда вытекает, что сегментация, т. е. объединение имеющихся блоков информации в сегменты, должна быть такой, чтобы частота обращения к внешней памяти была минимальной.

Оптимальная сегментация внешней памяти дает возможность повысить КПД ВС за счет увеличения процессорного времени. Однако обращаемость (частота обращения) к внешней памяти зависит от специфики решаемых задач, точнее, от последовательности используемых блоков информации. Поэтому для различных задач оптимальная сегментация внешней памяти будет различной. Естественно изменять (адаптировать) эту сегментацию с изменением потока решаемых ВС задач.

Формализуем задачу. Пусть $p(P)$ — статистические свойства вариантов P обращения к блокам информации внешней памяти, а U — сегментация, т. е. определенное объединение блоков в сегменты. Тогда свойства $p(P)$ генерируют свойства межсегментных связей, т. е. статистические свойства переходов от одного сегмента к другому. Критерием эффективности (2.2.9) сегментации для данной задачи следует считать число обращений к межсегментным связям при ее решении, Осредняя эту величину ло потоку решаемых задач, получаем $Q(U)$ — эффективность сегментации (2.2.10). Однако для вычисления необходимо знать $p(P)$ — вероятностные свойства потребности в информации, хранящейся во внешней памяти для обслуживания того или иного потока задач, что обычно неизвестно.

Именно это обстоятельство заставляет обращаться к адаптивной сегментации внешней памяти как способу отыскания оптимальной сегментации при наличии статистической устойчивости $p(P)$. Более того, при изменении $p(P)$ необходимо иметь возможность изменить сегментацию — пересегментировать память, для чего также необходимо обращение к процедуре адаптации.

Методы, используемые при этом, связаны с эволюционными алгоритмами адаптации. Решение задачи адаптивной сегментации методами эволюционной адаптации приведено в шестой главе (§ 6.2).

2.2.5.2. Адаптация расположения информационных блоков на магнитных дисках

Эта задача является конкретизацией предыдущей и связана с минимизацией времени механического движения считывающих головок от одного цилиндра к другому. Здесь роль сегментов выполняют цилиндры, причем расстояние между ними разное, что необходимо учитывать при расположении информационных блоков. Таким образом, на оптимальную сегментацию влияют не только статистические свойства последовательности обращения к блокам, но и взаимное расположение сегментов-блоков. Так, активно взаимодействующие блоки следует хранить если не на

одном цилиндре, то во всяком случае на ближайших, чтобы минимизировать время перемещения головок.

Аппаратом решения этой задачи являются методы эволюционной адаптации, описанные в шестой главе, а само решение приведено в § 6.2.

2.2.5.3. Адаптивное распределение памяти в многомашинной вычислительной системе (сети)

При решении задач с помощью многомашинной ВС встает проблема распределения памяти по отдельным ЭВМ ВС. Вся необходимая информация хранится в центральном банке данных, реализованном на одной или нескольких ЭВМ. Кроме того, каждая ЭВМ системы имеет собственный небольшой банк данных, где хранится информация, часто используемая этой ЭВМ. Обращение к центральному банку данных связано с потерей времени, которое на порядок больше времени обращения к собственному банку.

Отсюда возникает задача об организации локальных банков данных и использовании их при распределении задач между ЭВМ ВС.

Адаптивная процедура организации локального банка данных очевидна: здесь следует хранить наиболее часто требуемую информацию, что легко учесть. Если сведения о составе локальных банков данных сообщать диспетчеру, то можно получить эффективную дисциплину направления задач на ту ЭВМ, банк которой содержит максимум необходимой информации. Заметим, что при этом происходит автоматическая специализация локальных банков данных, что, естественно, повышает эффективность всей ВС.

Таким образом, подобная адаптация локальных банков дает возможность повысить эффективность многомашинной ВС.

Аппаратом решения этой задачи является эволюционная адаптация, применение которой описано в шестой главе (§ 6.3).

2.2.5.4. Адаптивные дисциплины распределения задач в многомашинной вычислительной системе

Синтез оптимальной дисциплины распределения задач на многомашинной ВС не представляет принципиальных трудностей, если известны все параметры потока задач и пропускные возможности машин. Однако именно эти параметры не только неизвестны, но и все время изменяются неопределенным образом. В таких условиях следует построить оптимальную дисциплину и изменять ее необходимым образом, чтобы поддерживать заданный критерий в экстремальном состоянии.

Заметим, что решить эту задачу без применения процедуры адаптации невозможно. Адаптация здесь имеет альтернативный характер и заключается в построении правила перехода от одной альтернативной дисциплины к другой, с тем чтобы в конце концов прийти к оптимальной в сложившейся ситуации дисциплине, экстремизирующей заданный критерий эффективности функционирования ВС (например, среднее время решения задачи, суммарную длину очередей и т. д.).

2.2.5.5. Адаптация многомашинной вычислительной системы к конкретному пользователю

Если активные пользователи ВС обладают статистической устойчивостью своих интересов по отношению к ВС, то естественно закреплять их за определенной ЭВМ. Это можно осуществить адаптивным путем перехода для каждого пользователя от одной альтернативной ЭВМ в ВС к другой. Такой процесс адаптации должен заканчиваться выбором ЭВМ, на которой задачи данного пользователя решаются наилучшим в заданном смысле образом. Для достижения поставленной цели естественно вос- пользоваться алгоритмом альтернативной адаптации.

Таким образом, системный уровень в ВС представляет широкое поле для приложения методов адаптации.

В целом же ВС является типичным примером сложного объекта, эффективное функционирование которого просто невозможно без адаптации, позволяющей преодолеть многочисленные факторы неопределенности в самой ВС и ее среде, т. е. потоке решаемых задач.

§ 2.3. Процессы обучения

Стандартная задача обучения состоит обычно в том, чтобы обучаемый наилучшим образом запомнил определенные порции информации, которые в дальнейшем будем называть лексическими единицами. Примером таких единиц являются операторы алгоритмического языка, слова иностранного языка, грамматические правила и т. д.

Алгоритм обучения представляет собой правило выбора порции обучающей информации U, которую необходимо заучить. Эффективность Q такого обучения — например, число известных ученику лексических единиц — можно оценивать по результатам периодического контроля обучаемого. Очевидно, что эффективность зависит от алгоритма обучения U и самого ученика:

$$Q = Q(U, \omega), \qquad (2.3.1)$$

где со — индивидуальные свойства ученика как объекта обучения. Очевидно, что эти свойства априори неизвестны в явном виде и могут быть получены лишь в результате довольно громоздкого процесса идентификации.

Процесс обучения естественно сделать адаптивным, т. е. приспосабливающимся к индивидуальным особенностям со обучаемого, которые, вообще говоря, могут изменяться в процессе обучения, т. е.

$$\omega = \omega(t). \tag{2.3.2}$$

Это можно осуществить путем соответствующего выбора порции t/ обучения, т. е. решая задачу адаптации

$$Q(U, \omega(t)) \to \min_{U} \Rightarrow U^*_{\omega(t)}, \tag{2.3.3}$$

где $U^*_{\omega}(t)$ — оптимальная порция обучения, зависящая от индивидуальных черт $\omega(t)$ ученика.

Вид зависимости (2.3.1) обычно неизвестен, но иногда можно оценивать Q, т. е. иметь наблюдения функции (2.3.1) в определенные моменты времени. Поэтому для решения задачи (2.3.3) можно воспользоваться методами адаптации, т. е. такого изменения порции U в процессе обучения, чтобы поддерживать критерий (2.3.1) в экстремальном состоянии в течение всего процесса обучения. Если имеются альтернативные алгоритмы обучения, то задача выбора алгоритма, оптимального в данный момент для данного ученика, решается методами структурной или, точнее, альтернативной адаптации. Здесь число альтернатив равно числу конкурирующих алгоритмов обучения.

Однако такой подход нельзя считать эффективным. Естественным развитием поиска наилучших методов обучения является синтез модели (2.3.2) ученика как объекта управления (обучения). Синтез модели нужно проводить адаптивно, т. е. не нарушая процесса обучения. Параметризуем модель (2.3.2):

$$\omega = \varphi\,(C, t), \tag{2.3.4}$$

где φ — выбранный оператор модели, а $C=(c_1,...,c_q)$ — параметры модели конкретного ученика, которые обычно оцениваются в процессе обучения методами адаптации. Для решения этой задачи необходимо воспользоваться методами параметрической адаптации (см. § 4.2). Теперь, располагая моделью (2.3.4) и критерием (2.3.1) эффективности обучения, можно на каждом шаге, решая задачу оптимизации (2.3.3), определить оптимальное обучение U^* в виде порции информации, которую следует выучить данному ученику (точнее: ученику с данной моделью).

Эффективность такого подхода очевидна (подробнее он рассмот рен в § 4.2).

Если нужно выбрать между различными альтернативными структурами модели $\varphi_1,...,\varphi_p$, то для этого следует воспользоваться альтернативной адаптацией. Как видно, задача эффективного обучения связана с решением задачи адаптации алгоритма обучения и модели пользователя с целью максимизации эффективности процесса обучения на всех его стадиях [297].

§ 2.4. Графы

Граф является очень характерным математическим объектом адаптации. Покажем это на примере задачи о его агрегации.

Задача об агрегации графа, которую часто называют задачей о *сегментации* или *разрезании* графа, довольно широко распространена в приложениях. Содержание ее состоит в следующем.

Пусть задан граф с вершинами и ребрами, имеющими различные веса, которые, вообще говоря, могут изменяться во времени. Следует так объединить эти вершины в сегменты, чтобы:

а) суммарный вес вершин, попавших в один сегмент, не превышал некоторую постоянную этого сегмента («объем сег мента»),

б) суммарный вес связей между сегментами был мини мальным.

Легко видеть, что число возможных агрегаций графа при за данных объемах сегментов достаточно велико. Из этого мно жества допустимых вариантов агрегации» следует выбрать наи лучший по критерию минимума суммарных связей между сегментами.

Сформулируем эту задачу как задачу дискретного програм-мирования..

Пусть веса вершин графа равны $a_1,...,a_n$ веса ребер — b_{ij} (i, j=l,...,n), а объем сегментов — c_l (l=1, ..., $m<n$). Очевидно, что a_i, b_{ij}, $c_l>0$.

Агрегация задается характеристической матрицей

$$U=\begin{Vmatrix} u_{11} & \dots & u_{1n} \\ \cdot & \cdot & \cdot \\ u_{m1} & \dots & u_{mn} \end{Vmatrix} \qquad (2.4.1)$$

имеющей m строк (по числу сегментов) и n столбцов (по числу вершин графа). Элементами этой матрицы являются двоичные числа

$$u_{ij}\in\{0;\ 1\}, \qquad (2.4.2)$$

указывающие на принадлежность (u_{ij}=1) или непринадлежность (u_{ij}=0) j-й вершины i-му сегменту.

Очевидно, что при этом должно выполняться следующее условие:

$$\sum_{i=1}^{m} u_{ij}=1 \qquad (j=1,\ldots,n), \tag{2.4.3}$$

т. е. каждая вершина агрегируемого графа должна попасть только в один сегмент. В результате матрица (2.4.1) содержит лишь n единиц, а остальные ее элементы — нули.

Теперь задачу об оптимальной агрегации графа можно записать в виде следующей квадратичной задачи дискретного программирования:

$$Q(U)=\sum_{i,j}^{n}\sum_{l,k}^{m} b_{ij}u_{li}u_{kj}\varphi(l,k)\to\min_{U\in S}\Rightarrow U^*,$$

2.4.4) где S — область

допустимых значений матрицы U:

$$S:\begin{cases}\sum\limits_{i=1}^{m} u_{ij}=1 & (j=1,\ldots,m);\\ \sum\limits_{i=1}^{n} a_i u_{li}\leqslant c_l & (l=1,\ldots,m);\\ u_{ij}\in\{0;1\} & (i=1,\ldots,n;\ j=1,\ldots,m),\end{cases} \tag{2.4.5}$$

а $\varphi(\cdot,\cdot)$ —

характеристическая функция вида

$$\varphi(l,k)=\begin{cases}1 & \text{при } l\neq k;\\ 0 & \text{при } l=k.\end{cases} \tag{2.4.6}$$

Решение U^* этой задачи и является искомой оптимальной агрегацией графа.

Реализация точного решения (2.4.4) связана в той или иной степени с перебором, что не позволяет решать задачи даже средней сложности. Однако применение эвристических процедур эволюционной адаптации дает возможность получать вполне применимые оценки точного решения U^* за допустимое время.

Решение рассмотренной задачи об адаптивной агрегации графа будет рассмотрено в § 6.2.

§ 2.5. Дисциплины обслуживания

Задача адаптации дисциплины обслуживания в системе массового обслуживания возникает в связи с непредвиденными и неконтролируемыми изменениями в среде и системе обслуживания, которые неизбежно изменяют оптимальную настройку дисциплины обслуживания, если таковая была реализована в системе. Поэтому систематическая подстройка решающего правила обслуживания неизбежна при желании поддерживать систему в оптимальном режиме независимо от изменений, происходящих в среде и системе.

Пусть X — контролируемое состояние среды, E — ее неконтролируемое состояние. Таким образом, пара

$$\langle X, E \rangle \qquad (2.5.1)$$

однозначно описывает среду, в которой находится система массового обслуживания. Например, X — паспортные данные заявок на обслуживание, а E — интенсивность их поступления. Аналогично пара

$$\langle Y, H \rangle \qquad (2.5.2)$$

описывает состояние системы: Y — ее контролируемые, а H — неконтролируемые факторы. Например, Y — очереди в пунктах обслуживания, а H — интенсивность обслуживания.

Критерий эффективности системы обычно имеет экстремальный и векторный характер. Он определен на контролируемых состояниях системы и среды:

$$Q = Q\,(Y, X). \qquad (2.5.3)$$

Примером таких критериев являются среднее время пребывания заявки в системе, среднее время ожидания обслуживания, средняя длина очереди к пунктам обслуживания и т. д. — в зависимости от целей руководителя адаптируемой системы массового обслуживания.

Состояние системы Y, в свою очередь, зависит от упомянутых X, E и H, а также от дисциплины обслуживания U:

$$Y = F\,(X, E, H, U), \qquad (2.5.4)$$

где F — оператор системы.

Здесь под дисциплиной обслуживания U подразумевается решающее правило распределения заявок по системе на основе информации о состоянии среды и системы:

$$U = U\,(X, Y). \qquad (2.5.5)$$

Оптимальность дисциплины U связана с экстремизацией критерия функционирования (2.5.3). Это означает, что для синтеза оптимальной дисциплины U^* необходимо решить следующую оптимизационную задачу:

(2.5.6)

$$Q[F(X, E, H, U), X] \to \underset{U \in S}{\text{extr}} \Rightarrow U^*,$$

где S — ограничения, накладываемые на выбор дисциплины обслуживания U. Они могут быть связаны, например, со временем принятия решения (2.5.5), которое не должно быть больше заданной величины, и т. п.

Хорошо видно, что решать задачу (2.5.6) на стадии проектирования невозможно ввиду того, что априори неизвестны факторы E и H. Осреднение по этим факторам вводить нельзя, так как они имеют нестационарный характер.

Поэтому задачу синтеза оптимальной дисциплины U^* следует решать путем адаптации системы массового обслуживания, т. е. в режиме ее нормальной эксплуатации. Тогда адаптация сводится к решению задачи

$$Q(U) \to \underset{U \in S}{\text{extr}} \Rightarrow U^* \qquad (2.5.7)$$

по локальным наблюдениям оценок значения критерия Q при различных дисциплинах:

$$\hat{Q}_1 = \hat{Q}(U_1), \ldots, \hat{Q}_N = \hat{Q}(U_N). \qquad (2.5.8)$$

Алгоритм адаптации должен указывать последовательность переходов от одной дисциплины к другой

$$U_1 \to U_2 \to \ldots \to U_N \to \ldots, \qquad (2.5.9)$$

которая стремится к решению U^*, оптимальному в данной ситуации.

Если структура дисциплины определена априори:

$$U = f(X, Y, C), \qquad (2.5.10)$$

где f — заданный алгоритм дисциплины обслуживания; $C = (c_1, \ldots, c_2)$ — параметры этой дисциплины, то ее адаптация, связана с определением параметров C, т. е. с решением задачи

(2.5.11)

$$Q(C) \to \underset{C \in S_C \subset R^q}{\text{extr}} \Rightarrow C^*,$$

где S_C — множество допустимых параметров дисциплины обслуживания f.

Например, при распределении одного потока заявок на два пункта обслуживания состояние системы обслуживания можно охарактеризовать длинами очередей l_1 и l_2 к обоим пунктам. Дисциплиной обслуживания f будет правило, указывающее пункт обслуживания для поступающей заявки:

$$f:\varphi(l_1, l_2, c_1, c_2) \gtrless \begin{cases} <0, & \text{то на пункт 1;} \\ >0, & \text{то на пункт 2;} \\ =0, & \text{то с вероятностью } c_2 \text{ на пункт 1.} \end{cases} \qquad (2.5.12)$$

Здесь φ — выбранная функция — например, вида

$$\varphi = c_1 l_1 - (1 - c_1)\, l_2, \qquad (2.5.13)$$

причем параметрами этой дисциплины являются c_1 и c_2 ($k = 2$) с естественными ограничениями

$$S_C: \begin{cases} c_1 > 0; \\ 0 \leqslant c_2 \leqslant 1. \end{cases} \qquad (2.5.14)$$

Как видно, выбор оптимальных значений c_1 и c_2, экстремизирующих заданный критерий функционирования (например, среднюю суммарную длину очереди), является задачей параметрической адаптации.

Если структуру дисциплины обслуживания определить трудно, то эту задачу следует решать методом структурной адаптации. Здесь возможно использовать оба подхода, описанных в § 1.5, — альтернативную и эволюционную адаптацию.

В первом случае следует определить альтернативные дисциплины:

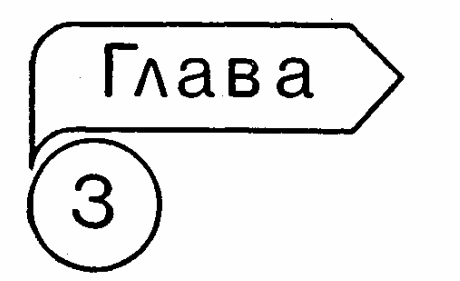
$$U \in \{U_1, \ldots, U_q\} \qquad (2.5.15)$$

и адаптация заключается в «переключении» указанных альтернативных дисциплин (2.5.15) таким образом, чтобы поддерживать в каждый момент дисциплину, доставляющую экстремальное значение критерию адаптации [296].

При эволюционном подходе алгоритм адаптации реализуется конечным автоматом, входом которого являются состояния среды X и системы обслуживания Y, а выходом — решение об обслуживании в данный момент (например, направление очередной заявки на определенный прибор обслуживания). Структура этого автомата, т. е. функции переходов и выходов, определяется в режиме эволюционной адаптации, т. е. путем моделирования эволюции графа такого автомата, реализующего адаптируемую динамику обслуживания.

Таким образом, адаптация дисциплины обслуживания в любой системе массового обслуживания, подверженной неконтролируемым воздействиям, является эффективной мерой поддержания системы обслуживания в оптимальном состоянии независимо от действия различных непредвиденных внешних и внутренних факторов. Эти задачи рассмотрены подробно в § 6.3.

Здесь приведены лишь некоторые примеры объектов адаптации. Объекты адаптации «возникают» всегда, когда приходится эксплуатировать объект, относительно которого нет четких представлений, т. е. нет его адекватной модели. Это типичный «черный ящик», который приспособлен удовлетворять наши потребности. Эффективность его функционирования очень важна для нас, что и порождает проблему адаптации. К сожалению, таких «черных ящиков» нас окружает значительно больше, чем «прозрачных», особенно в области технологии. Известно, что зачастую технологический процесс возникает, служит несколько лет и снимается (заменяется другим), так и не получив хорошего объяснения в виде адекватной модели. А если вспомнить, что наша цивилизация — технологическая, то легко понять, как важно, актуально и перспективно разрабатывать методы адаптации. Они описаны в последующих главах.



Случайный поиск в задачах оптимизации и адаптации

> Нет такого печального случая, из которого многие люди не извлекли бы какую-либо выгоду, нет такого счастливого случая, который люди неосторожные не обратили себе во вред.
>
> *Франсуа де Ларошфуко*

Случайный поиск, возникший как научное направление более двадцати лет назад [159, 288], в настоящее время стал распространенным и эффективным средством решения сложных задач оптимизации и адаптации. В этой главе рассказывается о предпосылках появления случайного поиска и об идеях основных его алгоритмов. Пожалуй, самой привлекательной стороной случайного поиска является его огромная эвристическая сила, которая позволяет широко использовать при синтезе алгоритмов аналогии как биологического, так и социального характера [286].

§ 3.1. Рандомизация управления

Случайный поиск как метод управления и решения сложных задач опирается на вероятностный характер окружающего нас мира. «Господство вероятности» вовсе не является абсолютным: оно не исключает наличия строго детерминированных закономерностей и законов, которые и породили известные регулярные методы поиска типа методов градиента и т. п. Специфика процесса познания заставляет нас прежде всего искать именно неслучайные, детерминированные закономерности, и только если их не удается найти, мы с большой неохотой обращаемся к статистическим, вероятностным закономерностям. Долго такого рода закономерностям отказывали в праве на существование, что было сформулировано в печально известном девизе «Наука — враг случайностей», по сути дела запрещающем использование статистических закономерностей и во всяком случае исключающем их применение в науке. В общем и целом порочная, эта точка зрения имела тем не менее свои психологические основания.

Дело в том, что психология нашего познания трудно мирится с недетерминированностью моделей. Такие модели «некомфортны» для нашего обыденного сознания. За всякой случайностью всегда хочется увидеть скрытую невыясненную детерминированную закономерность. Это желание вполне обоснованно, так

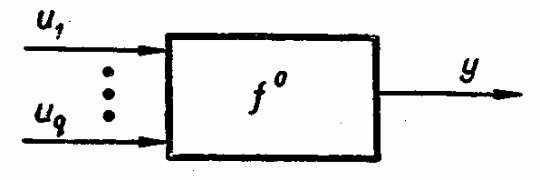


Рис. 3.1.1. Модель изучаемого явления.

как очень часто явление считается случайным в силу его плохой изученности, по мере же изучения удается построить его детерминированную модель.

Рассмотрим модель этого феномена. Пусть изучаемое явление представляет собой некий преобразователь n причин u_1, u_2,...,u_n в следствие y (рис. 3.1.1):

$$y = f^0(u_1,...,u_n), \qquad (3.1.1)$$

где f^0 — регулярный оператор объекта, а причины u_1,...,u_n изменяются под воздействием среды или экспериментатора. Очевидно, что такое явление будет вести себя вполне предсказуемо, т. е. не случайно, если будет известно состояние всех его причин. Только в этом случае можно точно предсказать значение следствия. Но достаточно не знать хотя бы одну из причин, чтобы явление стало непредсказуемым, а следовательно, могло рассматриваться как случайное. (Заметим, что непредсказуемость и случайность не тождественны. Случайность всегда подразумевает наличие определенных устойчивых статистических свойств явления — вероятности, среднего, дисперсии и т. д., в то время как непредсказуемость этого не требует.)

Как видно, появление случайности здесь является прямым следствием отсутствия необходимой информации. Чем большей информацией о причинах изучаемого явления мы располагаем, тем более оно определенно для нас. С этих позиций можно легко провести грань между стохастическими и детерминированными явлениями. Действительно, детерминированное явление отличается малым числом легко наблюдаемых причин, а стохастическое — большим количеством причин, контроль которых затруднен.

Отсюда ясно, почему стохастические явления начали изучать сравнительно недавно. Естественно, что наука сначала рассматривала лишь простейшие явления, зависящие от небольшого числа причин, и только потом перешла к изучению сложных, требующих создания стохастических моделей.

Действительно, всегда предпочитается детерминированное описание явления, исключающее двусмысленность и многовариантность. Однако такое описание не во всех случаях удается построить и приходится ограничиваться статистическими закономерностями типа: «в ситуации S с вероятностью p происходит событие A...». Можно ли отказать подобным закономерностям

в существовании? Разумеется, нельзя! Первыми это поняли физики, когда в конце первой четверти нашего века осознали неизбежность «странного» стохастического мира. Было установлено, что законы микромира вообще не могут быть регулярными, они в принципе имеют стохастический характер. Например, нельзя сказать наверняка, попадет данная микрочастица в мишень или нет, — самый точный ответ в этом случае может состоять в указании вероятности интересующего нас события. И только!

Такая вероятностная точка зрения на закономерности окружающего нас мира быстро распространилась и на другие области науки и техники. Не только трудности сбора информации при изучении сложных явлений заставили обратиться к идее стохастичности. Есть и еще одно довольно неожиданное преимущество стохастических моделей: они очень просты и удобны. Дело в том, что для практических целей очень редко возникает необходимость в точном предсказании поведения интересующего нас явления, или потребность иметь его детерминированную модель. Ведь модель нужна не сама по себе, а либо для прогнозирования, либо для управления, т. е. для принятия каких-то управляющих решений, которые, как правило, не требуют точных моделей (точнее: малочувствительны к точности модели). Это обстоятельство позволяет широко и эффективно пользоваться стохастическими моделями. В данном случае рандомизация выступает в качестве способа упрощения моделей. Случайный механизм почти всегда проще детерминированного. Именно это обстоятельство заставляет обращаться не к детерминированным, а к стохастическим моделям.

Стохастические модели — это способ описания окружающего нас мира. Выражается он в указании, какие и насколько частые отклонения возможны в детерминированной модели. Стохастическая модель включает в себя детерминированную как частный случай, как результат некоего осреднения. Кроме того — что самое главное — она описывает отклонения от этой модели с указанием их вероятности. Здесь нет случайности в явном виде — это лишь осмысление результатов действия случайности.

Человек уже давно заметил, как сильно он зависит от случайных факторов, которые буквально осаждают его со всех сторон. Живя в вероятностном мире, человечество волей-неволей должно было выработать меры борьбы с негативными последствиями случая. Изобретение стохастических моделей явилось одной из наиболее эффективных мер.

Другим способом борьбы с такого рода случайностью является управление, т. е. организация и реализация мер по устранению негативных последствий случайного фактора. Сам по себе процесс управления заключается в построении таких воздействий

на объект, с помощью которых его можно перевести в требуемое целевое состояние независимо от случайных внешних и внутренних факторов, действующих на этот объект. Не будь случайных факторов — не нужно было бы и управления. Таким образом, изобретение управления является реакцией на стохастичность нашего мира, это способ борьбы со случайностью, мера элиминирования, исключения случая, который мешает нам добиваться и достигать поставленных целей.

Однако, как всем известно, случай бывает не только неприятным — он может быть вполне приемлемым, желательным и даже счастливым. Едва ли стоит приводить примеры: у каждого из нас есть в памяти много удивительных счастливых случайностей, которые произошли в нашей жизни. Но не о них сейчас речь. Естественно задать вопрос: а нельзя ли к счастливой случайности относиться не пассивно, т. е. не дожидаться ее реализации, как «манны небесной», а активно идти ей навстречу, искать ее и использовать для своих нужд?

Эта соблазнительная идея «утилизации» случайности в последнее время получает значительное распространение. Здесь имеют место два направления использования случайности. Первое связано с обобщающими возможностями вероятностных моделей, которые содержат в себе как частный случай регулярные модели и закономерности. Такого рода обобщающая сила случая является его ценнейшим качеством, которое дало новые направления современной науки и техники в виде теорий вероятностей, массового обслуживания, надежности, принятия решений и т. д. [217—219]. Однако не это направление будет нас здесь интересовать.

Обратимся ко второму направлению, связанному со случайным поиском новой информации и принятием решения для оптимизации сложных объектов. Под «объектом», как и в § 1.1, будем подразумевать некую часть окружающего нас мира, состояние которой нас не удовлетворяет, в силу чего и организуется управление. Такой «частью» может быть модель объекта, и объектом в этом случае является модель, с помощью которой определяется оптимальное управление или решение, реализуемое впоследствии на практике. Дело в том, что для принятия решения об очередном акте управления объектом необходимо получить информацию о его состоянии. Чем сложнее объект, тем больше такой информации необходимо иметь. И процесс сбора информации, и принятие решений можно сделать случайными или, как часто говорят, рандомизированными (от английского слова *random* — случайный). Оказывается, что для сложных объектов управления такая рандомизированная стратегия поиска решения часто бывает более выгодной, чем регулярная, детерминированная.

Проще говоря, случайное поведение в процессе управления сложным объектом часто бывает более эффективным, чем регулярное. Этот на первый взгляд парадоксальный результат имеет глубокие обоснования как математического, так и методологического характера, которые будут приведены ниже.

Случайный поиск — или стохастический, статистический, вероятностный, рандомизированный поиск — представляет собой достаточно универсальный метод решения сложных задач, выдвигаемых современной наукой, техникой и народным хозяйством. Рассмотрим основные предпосылки, черты, особенности и специфику применения случайного поиска как очень эффективного современного метода оптимизации и адаптации сложных объектов.

§ 3.2. Предпосылки случайного поиска

Случайный поиск как метод управления имеет свою историю, связанную в основном с отстаиванием «права на существование». Однако случайный поиск возник не на пустом месте: его появлению предшествовал период предыстории, когда инструмент случайности оттачивался на решении различного рода прикладных и теоретических задач. Рассмотрим их.

I. Простейший, всем известной схемой, использующей идею случайного поиска, является метод, называемый обычно **методом проб и ошибок**. Строго говоря, его нельзя даже назвать методом, так как довольно трудно точно определить, в чем же он состоит. В действительности правильнее было бы говорить о методе случайных проб и исправления ошибок.

Как видно, метод предполагает возможность случайных экспериментов (проб) с некоторыми объектами, причем допустим неудачный результат экспериментов. Эти неудачи не имеют рокового характера, и их последствия легко устранимы (исправимы), т. е. стоимость исправления допущенной ошибки невелика. Такова ситуация, типичная для применения метода проб и ошибок. Само применение метода связано прежде всего с использованием идеи случайности: здесь пробы (эксперименты) всегда случайны, поскольку для их организации нет никаких иных соображений. В самом деле, если бы такие соображения были, то отпала бы необходимость в методе проб и ошибок.

Итак, метод проб и ошибок опирается на два существенных предположения — нероковой характер ошибок и отсутствие априорных соображений о том, как и в каком направлении делать пробы. Эти предположения почти неизбежно приводят к случаю. Действительно, случайность в такой ситуации является

единственно разумной мерой: случайность почти ничего не стоит и всегда содержит в себе искомое решение. Это значит, что достаточно длительный поиск всегда приведет к решению задачи.

Сказанное, безусловно, оправдывает применение метода проб и ошибок для решения практических задач, несмотря на его несколько одиозную славу. (Вспомним, что часто мы так «обзываем» всякую неудачную постановку эксперимента, когда априорных сведений достаточно для организации детерминированного поведения, т. е. для построения регулярного плана эксперимента. Однако подобная ситуация складывается не столь часто, как хотелось бы: априорных данных чаще бывает недостаточно, особенно в поисковых экспериментах, и неизбежно приходится обращаться к пресловутому методу проб и ошибок с его «неразумной», но спасительной случайностью.)

II. Метод Монте-Карло появился в 1943 г. и с тех пор стал одним из самых распространенных вычислительных методов. Метод Монте-Карло представляет собой наиболее эффективное (и эффектное) применение случайности. Математически он, как правило, связан с вычислением многомерных интегралов в заданных областях, *а* содержательно — с определением характеристик моделируемых случайных процессов. Популярность этого метода обусловлена прежде всего тем, что он явился основой для моделирования сложных процессов (например, технологических) с помощью ЭВМ.

Дело в том, что очень часто мы много знаем о деталях интересующего нас процесса, но почти ничего не можем сказать о его поведении в целом. Иначе говоря, по локальному поведению процесса необходимо определить его интегральные свойства. В простейшем случае, когда локальные свойства процесса удается описать в виде регулярных дифференциальных отношений, получаем дифференциальное уравнение, интегрирование которого и позволяет определить глобальные свойства процесса. Примером такого уравнения является известный второй закон Ньютона.

Однако для сложных процессов, обладающих стохастическими локальными свойствами, в лучшем случае получается стохастическое дифференциальное уравнение, интегрирование которого представляет огромные трудности, а в худшем — лишь правила перехода процесса из одного состояния в другое. Эти правила имеют определенный стохастический характер, т. е. всегда можно «разыграть» с помощью ЭВМ конкретные случайные реализации переходов и тем самым построить различные траектории процесса. Очевидно, что такие траектории случайны и каждая из них в отдельности почти ни о чем не говорит. Однако возможность построить с помощью ЭВМ большое число траекторий нашего процесса открывает широкие перспективы, которые и и использует метод Монте-Карло.

Действительно, поскольку каждая полученная траектория моделирует поведение реального процесса, то интересующие нас свойства процесса можно определить, обрабатывая траекторий как различные наблюдения реального процесса. Например, таким образом можно определить поведение в среднем, максимальные уклонения, дисперсию и другие вероятностные свойства нашего процесса. Как видно, суть метода Монте-Карло состоит в многократном моделировании («разыгрывании») случайного поведения процесса и принятии каких-то необходимых решений на этой основе. Для разыгрывания нужно уметь многократно моделировать локальное случайное поведение процесса с целью определения его интегральных, глобальных характеристик, на основе которых применяются управленческие решения.

Приведем пример применения метода Монте-Карло для решения такой простой задачи, как определение среднего случайной переменной. Прежде всего нужно уметь моделировать эту случайную переменную, для чего необходимо знать ее распределение, с помощью которого описывается локальное поведение исследуемого процесса. Пусть $p(x)$ — плотность распределения (x — значение случайной величины).

Располагая этим распределением, можно поступить двояко. С одной стороны, естественно воспользоваться известной формулой для среднего как математического ожидания, которое вычисляется с помощью интеграла:

$$\bar{x} = \int x p(x)\, dx. \tag{3.2.1}$$

С другой стороны, можно для оценки $\bar{x}$ применить метод Монте-Карло. В данном случае он заключается в том, чтобы оценить среднее $\bar{x}$ с помощью ряда специально генерированных случайных чисел

$$x_1, \ldots, x_N, \tag{3.2.2}$$

имеющих плотность распределения $p(X)$. Тогда оценка среднего x равна среднему этих чисел, т. е.

$$\hat{\bar{x}}_N = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{3.2.3}$$

Здесь через $\hat{\bar{x}}_N$ обозначена монте-карловская оценка $\bar{x}$ на базе N чисел (3.2.2). Как видно, все трудности метода заключаются в «разыгрывании» этого ряда случайных чисел в соответствии с заданным распределением $p(x)$. Если же мы располагаем возможностью наблюдать реальный процесс, то ряд (3.2.2) представляет собой результаты конкретных наблюдений этого процесса.

Таким образом, метод Монте-Карло обходит трудности, связанные с наблюдением реального процесса, и позволяет получить ряд (3.2.2) путем моделирования случайной величины x в соответствии с имеющейся плотностью распределения $p(x)$. Дальнейшая обработка (3.2.2) производится аналогично.

Этот подход легко обобщается На любую функцию $y=f(x)$ от случайного аргумента x. Выражение

$$\hat{\bar{y}}_N=\frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad (3.2.4)$$

является монте-карловской оценкой среднего

$$\bar{y}=\int f(x)\,p(x)\,dx. \quad (3.2.5)$$

Естественно задать вопрос: не проще ли — а главное, не точнее ли — вычислить интеграл (3.2.1), например, методом трапеций, чем оценивать его по формуле (3.2.3) с помощью ряда специально разыгранных чисел (3.2.2)? Ответ зависит от сложности и требуемой точности вычисления данного интеграла.

В одномерном случае, когда x — скаляр ($x\in R^1$), среднее $\bar{x}$, безусловно, удобнее вычислять с помощью стандартных методов численного интегрирования. Однако в случае векторного аргумента

$$X=(x_1, \ldots, x_n) \quad (3.2.6)$$

вычисление многомерного интеграла

$$\bar{y}=\int f(X)\,p(X)\,dX \quad (3.2.7)$$

представляет уже довольно сложную задачу, точность решения которой уменьшается с ростом размерности n интеграла. Это и создает преимущество для метода Монте-Карло. В формуле (3.2.7) $f(X)$ — заданная скалярная функция вектора X, $p(X)$ — плотность распределения вектора X, которая в случае независимости компонент вектора X равна $p(X) = p_1(x_1) \ldots p_n(x_n)$, где $p_j(x_j)$ — плотность распределения j-й компоненты x_j. Здесь $X\in R^n$ и случайный вектор X распределен по заданному многомерному закону $p(X)$. При этом оценка многократного (n-крат-ного) интеграла (3.2.7) имеет вид

$$\hat{\bar{y}}_N=\frac{1}{N}\sum_{i=1}^{N} f(X_i), \quad (3.2.8)$$

где случайные векторы X_i $(i=1,...,N)$ распределены в соответствии с функцией $p(X)$, определенной в n-мерном пространстве.

Оценка (3.2.8) является случайной величиной: для другого ряда случайных чисел X_i она будет иной. Как всякую случайную величину, монте-карловскую оценку $\hat{\bar{y}}_N$ удобно характеризовать ее математическим ожиданием $M\hat{\bar{y}}_N$ и дисперсией $D\hat{\bar{y}}_N$. Эти величины обладают следующими интересными свойствами:

1. Оценка $\hat{\bar{y}}_N$ не смещена:

$$M\hat{\bar{y}}_N = \bar{y}, \tag{3.2.9}$$

т. е. в среднем совпадает с точным значением интеграла (3.2.7).

2. Оценка $\hat{\bar{y}}_N$ состоятельна:

$$\lim_{N\to\infty} \hat{\bar{y}}_N = \bar{y}, \tag{3.2.10}$$

т. е. в пределе совпадает со значением интеграла (3.2.7).

3. Ее дисперсия зависит от N следующим образом:

$$D\hat{\bar{y}}_N = \frac{c}{N}, \tag{3.2.11}$$

где c — некоторая не зависящая от N постоянная.

Эти свойства метода Монте-Карло делают его очень эффективным при решении сложных задач. Действительно, первое свойство (несмещенность оценки) гарантирует, что при любом N, даже при $N=1$, полученная оценка в среднем совпадает с искомым значением. Второе свойство (состоятельность оценки) гарантирует ее сходимость к искомому значению с увеличением N, а третье — определяет скорость этой сходимости (как $1/N$), что позволяет оценить, насколько монте-карловская оценка отличается от точного значения, или каким должно быть N, чтобы эта оценка отличалась от искомой на величину не более заданной.

Таким образом, случайность в методе Монте-Карло используется для моделирования (имитирования) локального поведения процесса. Это и обеспечивает методу широкие возможности и огромные перспективы. Так, в последнее время бурно развивается метод имитационного моделирования, который является развитием метода Монте-Карло для решения задач моделирования поведения сложных систем.

III. Метод рандомизации предложен в 30-х годах Фишером, заложившим основы математической теории эксперимента, и широко используется в планировании эксперимента. Метод также

опирается на специально генерируемую случайность. Суть проблемы заключается в следующем. Пусть необходимо построить модель изучаемого явления, зависящего от ряда факторов. Для простоты положим, что таких факторов два — x_1 и x_2. Оба находятся в распоряжении исследователя, причем интерес представляет влияние лишь первого (x_1), а второй (x_2) является досадной помехой, которую следует исключить (элиминировать). Таким образом, исследователя интересует модель

$$y = f(x_1), \tag{3.2.12}$$

а в эксперименте он наблюдает отдельные реализации зависимости

$$Y = f^0(x_1, x_2), \tag{3.2.13}$$

где f^0 — оператор объекта. Как видно, задача заключается в синтезе модельного оператора f. Прежде всего исследователь строит план эксперимента, т. е. фиксирует конкретные значения обоих факторов, при которых необходимо поставить эксперимент и определить реакцию исследуемого явления. План по первому фактору должен быть таким, чтобы как можно точнее определить модель f. А что делать со вторым фактором, который не входит в модель (3.2.12)? Его следует элиминировать, т. е. исключить. Если оператор f^0 объекта известен, то исключение реализуется путем обычного осреднения по исключаемому фактору:

$$f(x_1) = \int f^0(x_1, x_2)\, p_2(x_2)\, \mathrm{d}x_2, \tag{3.2.14}$$

где $p_2(x_2)$ — плотность распределения фактора x_2. Определим интеграл (3.2.14) методом Монте-Карло — с применением формулы (3.2.4):

$$\hat{f}_N(x_1) = \frac{1}{N}\sum_{i=1}^{N} f^0(x_1, x_{2i}), \tag{3.2.15}$$

где f_N — оценка искомой модели на базе N экспериментов, в которых случайно изменяется лишь второй фактор, т. е. x_{2i} (i=1,...,N) — случайные значения фактора x_2, распределенные в соответствии с заданной функцией плотности его распределения $p_2(x_2)$. Здесь случайность фактора x_2 понадобилась для реализации осреднения по нему.

Описанный путь нереален, так как исследователь никогда не располагает зависимостью $f^0(x_1, x_2)$, но интересен тем, что здесь содержится идея рандомизации: элиминируемый фактор намеренно делается случайным. При этом необходимо знать плотность

его распределения $p_2(x_2)$ для тех случаев, когда будет использоваться синтезируемая модель *f,* в чем, пожалуй, и заключается самая большая трудность использования метода рандомизации. Однако, если такой информации нет, естественно считать это распределение самым неинформативным, т. е. равномерным. Так обычно и поступают. План эксперимента, состоящий из N опытов, выглядит следующим образом:

$$\langle x_{1i}, x_{2i} \rangle \quad (i=1,...,N), ... \quad (3.2.16)$$

где x_{1i} — значение первого фактора в i-м опыте, которое определяется стандартными методами (например, в соответствии с ортогональным или D-оптимальным планом), а x_{2i} (i = 1, .., N) — случайные значения второго фактора, распределенные в соответствии с заданной функцией плотности распределения вероятностей $p_2(x_2)$.

Использование генерированных случайных значений элиминируемого фактора и образует основу процедуры рандомизации планируемого эксперимента. Здесь случайность играет роль элиминирующего, исключающего фактора, сглаживающего влияние посторонних воздействий на изучаемое явление.

IV. Стохастическая идентификация динамических систем также использует случайность в виде белого шума, который, как известно, имеет δ-образную автокорреляционную функцию:

$$R(\tau) = S^2\delta(\tau), \quad (3.2.17)$$

где S^2 — мощность шума. Известно, что линейная динамическая система однозначно характеризуется своей импульсной переходной функцией $w(t)$, которая представляет собой реакцию этой системы на импульсное воздействие. Эту переходную функцию обычно идентифицируют с помощью так называемого уравнения Винера—Хопфа:

$$R_{xy}(t) = \int_0^\infty w(\tau) R_{xx}(t-\tau)\, d\tau, \quad (3.2.18)$$

где $R_{xx}(t)$ — автокорреляционная функция входного сигнала $x(t)$, *а* $R_{xy}(t)$ — взаимно-корреляционная функция входа $x(t)$ и выхода $y(t)$. Метод стохастической идентификации заключается в том, что в качестве входного сигнала используется белый шум, описываемый выражением (3.2.17). Подставляя (3.2.17) в уравнение (3.2.18), получаем

$$w(t) = \frac{1}{S^2} R_{xy}(t), \quad (3.2.19)$$

т. е. импульсная переходная функция объекта в этом случае с точностью до постоянной совпадает со взаимно-корреляционной функцией белого шума и реакции объекта на этот шум.

Такой способ идентификации динамического объекта обладает тем преимуществом, что не вызывает резонансных явлений в объекте. Здесь эффективно используется «всечастотность» случайности.

V. Гомеостат Эшби является машиной, реализующей, по сути дела, один из простейших алгоритмов случайного поиска. Эшби предложил свой гомеостат в 1948 г. и описал его в книге [246].

Гомеостат представляет собой линейную динамическую систему

$$\frac{dX}{dt} = AX, \quad (3.2.20)$$

где $X = (x_1, ..., x_n)$; A — квадратная матрица $n \times n$, которая может изменяться случайным образом и координаты которой ограничены:

$$-1 \le x_i \le 1 \quad (i = 1, ..., n). \quad (3.2.21)$$

Известно, что система (3.2.20) в случае устойчивости имеет одно устойчивое состояние в начале координат (0, 0, ..., 0). При неустойчивости $X \to \infty$ и срабатывают упоры: $|x_i| = 1$.

Задача заключалась в том, чтобы автоматически определять такие матрицы A, которые делали бы гомеостат устойчивым. Эшби предложил в момент обнаружения неустойчивости, т. е. при

$$|x_i| = 1 \text{ хотя бы для одного } i = 1, ..., n, \quad (3.2.22)$$

изменять матрицу A случайно, т. е. вводить в гомеостат элемент случайности. Алгоритм управления гомеостатом имеет вид

$$A \simeq \begin{cases} \xi & \text{при } |x_i| = 1 \text{ хотя бы для одного } i; \\ \text{const} & \text{при } |x_i| < 1 \text{ для всех } i \quad (i = 1, ..., n); \end{cases} \quad (3.2.23)$$

где ξ — реализация случайной матрицы, коэффициенты которой, например, распределены равномерно в интервале [-1; 1]. Эшби экспериментально и теоретически показал, что гомеостат довольно быстро случайно находит «устойчивую» матрицу A и фиксирует ее. Если намеренно вывести гомеостат из устойчивого состояния (например, изменив ряд коэффициентов его матрицы), то он снова случайно найдет тот вариант матрицы, который делает его устойчивым, и зафиксирует его.

Здесь случайность ξ выступает в качестве источника возможностей, которые реализует гомеостат. Если благоприятных возможностей не слишком мало и не приходится дорого платить за

ошибку, то выбор устойчивой матрицы целесообразно доверить случаю. Это значительно проще, чем искать ее по известному, но громоздкому критерию отрицательной действительной части всех собственных чисел матрицы A.

Нетрудно заметить, что гомеостат по сути дела реализует метод проб и ошибок, рассмотренный выше. Здесь критерием успешности случайной пробы является выполнение условия $|x_i| < 1$ для всех $i = 1, ..., n,$ а сам случай реализуется в виде матрицы со случайными коэффициентами, что и дало возможность Эшби реализовать гомеостат с помощью довольно простой электрической схемы при $n = 4$.

Аналогичная идея реализуется так называемым «усилителем мыслительных способностей», также предложенным Эшби [245]. Несколько претенциозное название этого усилителя связано с любопытной мыслью о том, что случай содержит ответы на все вопросы, а правильный ответ может реализоваться путем многоэтапной селекции, отбора. К сожалению, Эшби не указал, каким образом делать этот отбор. Во всяком случае, отбор должен опираться на проверку истинности полученного промежуточного результата путем определенных процедур вывода, позволяющих отсеивать заведомую нелепость. Эшби даже доказал теорему, что такой процесс заканчивается за конечное время. Идея доказательства опирается на очевидное соображение, что случайное исходное разнообразие можно исчерпать за конечное число этапов селекции, если на каждом этапе редукция (снижение) разнообразия конечным образом отличается от нуля.

VI. Случайные поисковые сигналы могут быть эффективно использованы для экстремального управления непрерывным объектом [118], т. е. для минимизации его выхода путем соответствующего воздействия на вход.

Пусть объект управления представляет собой статический $(n+1)$ - полюсник (см. рис. 3.1.1) с n входами $x_1, \ldots, x_n$ и одним (скалярным) выходом $Q = Q(X)$, определяющим показатель качества этого объекта. Зависимость $y = Q(X)$ априори неизвестна, и нужно в процессе управления решить задачу оптимизации:

$$Q(X) \to \min_{X} \Rightarrow X^*, \tag{3.2.24}$$

где X^* — искомое состояние входа объекта, минимизирующее его выход.

При экстремальном управлении к каждому входу x_i добавляется аддитивно малая случайная функция $f_i(t)$ $(i = 1, ..., n)$. После операции синхронного детектирования, т. е. умножения выхода объекта на $f_i(t)$, и осреднения получаем сигналы

$$M[Q(X+F(t))f_i(t)] = \frac{\sigma_i^2}{2}\frac{\partial Q}{\partial x_i} \qquad (i=1,\ldots,n), \tag{3.2.25}$$

где $F(t)$ — центрированный вектор малого случайного возмущения всех входов:

$$F(t) = (f_1(t), ..., f_n(t));\qquad (3.2.26)$$

σ_i^2 — дисперсия i-го возмущения: $\sigma_i^2 = Df_i(t)$. Как видно, выход i-го синхронного детектора (3.2.25) пропорционален частной производной dQ/dxi, что сразу определяет необходимое направление изменения i-го входа. Так, при $\partial Q/\partial x_i > 0$ для минимизации Q следует уменьшать x_i, и наоборот. Это обстоятельство и используется при экстремальном управлении. На исполнительный механизм (интегратор), изменяющий параметр x_i, подается сигнал, пропорциональный выходу i-го синхронного детектора, что и реализует обратную связь экстремального управления. Получаем систему дифференциальных уравнений, описывающих поведение управляемого объекта:

$$\frac{dx_i}{dt} = -\mu_i \frac{\sigma_i^2}{2} \frac{\partial Q}{\partial x_i} \quad (i = 1, \ldots, n),\qquad (3.2.27)$$

где μ_i — параметр i-й обратной связи. Если для простоты положить $\mu_i = \mu$ и $\sigma_i = \sigma$, то эта система уравнений в векторной форме принимает вид

$$\frac{dX}{dt} = -\mu \frac{\sigma^2}{2} \nabla Q(x),\qquad (3.2.28)$$

где ∇ — знак $\left(\nabla = \left(\frac{\partial}{\partial x_i}, \ldots, \frac{\partial}{\partial x_n}\right)\right)$, что параметры X объекта изменяются в направлении антиградиента минимизируемой функции $Q(X)$, что и обеспечивает отыскание ее экстремума (если, разумеется, она унимодальна, т. е. имеет один экстремум).

Как видно из (3.2.28), при отсутствии случайных возмущений: $F(t)$, т. е. при $\sigma = 0$, никакой минимизации не происходит, так как $X = \text{const}$.

Таким образом, использование случайных сигналов $F(t)$ дает возможность эффективно управлять непрерывными объектами. Здесь случайность выступает в роли разнообразия, с помощью которого удается оценить градиентное направление минимизируемого объекта.

VII. Метод случайного баланса, предложенный Саттерзвайтом [264], использует случайный план для определения сущест-

венных факторов управляемого объекта. Пусть для простоты объект представляется линейной моделью

$$y=\sum_{i=1}^{n} a_i x_i+\varepsilon, \qquad (3.2.29)$$

где a_i — коэффициент влияния i-го фактора x_i (i=1,...,n), а ξ — остаточная ошибка модели и случайные ошибки наблюдения. Существенными факторами естественно называть те, изменение которых наиболее существенно изменяет выход y объекта, т. е. факторы, коэффициенты a_i которых велики.

Задача заключается в том, чтобы выявить эти существенные факторы, т. е. сводится к оценке коэффициентов a_i с помощью малого числа N экспериментов ($N<n$). С точки зрения строгой математики эта задача не имеет смысла, так как нельзя определить n параметров из N экспериментов в случае $n>N$. Однако использование статистического подхода позволяет эффективно оценить искомые коэффициенты. Именно на этом основан метод случайного баланса. Суть его состоит в том, что эксперименты ставятся в случайных точках

$$X_j = (x_{1j}, \ldots, x_{nj}) \quad (j=1,\ldots, N<n), \qquad (3.2.30)$$

координаты которых являются независимыми и центрированными случайными величинами. Тогда имеет место

$$\sum_{j=1}^{N} x_{ij}x_{lj}\approx 0 \quad (i\neq l), \qquad (3.2.31)$$

т. е. эта сумма мала, так как она с точностью до множителя $1/N$ является оценкой коэффициента корреляции i-й и l-й координат, которые по условию независимы. Поэтому, умножая (3.2.29) на x_{lj} и суммируя по j, получаем с учетом (3.2.31) оценку для искомых коэффициентов влияния:

$$\hat{a}_l=\frac{\sum_{j=1}^{N} y_j x_{lj}}{\sum_{j=1}^{N} (x_{lj})^2} \quad (l=1,\ldots,n), \qquad (3.2.32)$$

где y_j — реакция объекта на эксперимент в точке X_j.

Очевидно, что эти оценки приближенны вследствие приближенности выражения (3.2.31) и влияния случайных помех ξ. Строго

говоря, возможно получить любые значения этих оценок, сколь угодно сильно отличающиеся от действующих. (С этого упрека методу случайного баланса обычно и начинают его критику.) Но это примерно так же маловероятно, как существенное искажение модели из-за влияния «хвостов» центрированной помехи в объекте. Приближенность и смещенность оценок, вызванные малостью числа экспериментов, не должны никого смущать, поскольку нас здесь интересует лишь соотношение между коэффициентами влияния, а не их абсолютные значения. С помощью случайных экспериментов (3.2.30) удается выделить переменные, которые претендуют на существенность. Более того, их «претензии» можно даже проранжировать по полученным оценкам $\hat{a}_i$ из (3.2.32). Теперь из x_i ($i = 1,\ldots,n$) следует отобрать $k<N$, имеющих наибольший ранг. Пусть это будут x_1, ..., x_k. Тогда проведенные эксперименты позволяют построить линейную регрессию

$$y=\sum_{i=1}^{k} b_i x_i, \quad (3.2.33)$$

где коэффициенты b_i определяются стандартным для регрессионного метода образом. Ввиду того что $k<N$ и, следовательно, имеются «лишние» эксперименты, естественно проверить адекватность полученной модели (3.2.33) — например, с помощью Критерия Фишера. Это позволяет путем последовательных итераций выбрать такие факторы, которые дают адекватную модель при минимальном k, — искомые значимые факторы задачи. Теория и опыт показывают эффективность метода случайного баланса.

Как видно, случайность плана (3.2.30) в методе случайного баланса гарантирует решение поставленной задачи, т. е. выделение существенных факторов.

VIII. Смешанные стратегии в теории матричных игр с нулевой суммой впервые рассмотрены Нейманом еще в 1928 г. Ситуация матричной игры двух лиц (игроков) описывается следующим образом. Пусть первый игрок имеет n стратегий поведения, а второй — m. Тогда игра задана, если определен результат игры для всех возможных вариантов использования стратегий (число таких вариантов равно nm). Результат игры численно определяется платежной матрицей $||b_{ij}||$ ($i=1,\ldots,n$; $j=1,\ldots,m$), где b_{ij} — платеж первого игрока, использовавшего i-ю стратегию, второму, который применял j-ю стратегию. Выбор стратегий игроками происходит независимо.

Оказывается, что при отсутствии седловой точки в платежной матрице, т. е. при

$$\max_i \min_j b_{ij} \neq \min_j \max_i b_{ij}, \quad (3.2.34)$$

оптимальной является смешанная (рандомизированная) стратегия, определяемая для первого игрока вероятностями

$$p_1, \ldots, p_n \quad \left(\sum_{i=1}^{n} p_i = 1 \right), \tag{3.2.35}$$

где p_i — вероятность использования i-й стратегии (эта теорема доказана в работе [139]). Аналогичные вероятности определяют смешанную стратегию второго игрока.

Здесь рандомизация связана не столько с максимизацией выигрыша, сколько с минимизацией проигрыша. Случай здесь играет роль «дымовой завесы», не позволяющей противнику действовать наверняка. При отсутствии активности одного из игроков оптимальное решение всегда лежит в области чистых, или детерминированных, стратегий. Чистая стратегия оптимальна также при наличии в платежной матрице седловой точки, т. е. когда в выражении (3.2.34) стоит знак равенства.

Таким образом, случай в антагонистических играх выступает в роли фактора, мешающего противнику выиграть.

Как видно, случайность уже давно стала активным фактором, которым человек широко пользуется не только в обыденной жизни, но и в науке и технике. Случайный поиск является следующим этапом освоения случайности, ее приспособления для решения с помощью ЭВМ, пожалуй, самых распространенных задач нашего времени — задач оптимизации и адаптации сложных систем.

§ 3.3. Алгоритмы случайного поиска 3.3.1.

Структура поискового метода

Метод (алгоритм) решения задачи оптимизации и адаптации представляет собой последовательную процедуру, имеющую рекуррентный характер [149]. Это означает, что процесс поиска состоит из повторяющихся этапов, каждый из которых определяет переход от одного решения к другому, лучшему, что и образует процедуру последовательного улучшения решения:

$$U_0 \to U_1 \to \ldots \to U_N \to U_{N+1} \to \ldots \tag{3.3.1}$$

В этой последовательности каждое последующее решение в определенном смысле лучше, предпочтительнее предыдущего, т. е.

$$U_N \succ U_{N-1} \quad (N = 1, \ldots). \tag{3.3.2}$$

Здесь смысл знака предпочтения «$\succ$» может быть разным. Например, если $U_{N-1}, U_N \in S$, то (3.2.2) может быть связано со значениями критерия, т. е. $Q(U_N) < Q(U_{N-1})$. Если $U_{N-1} \notin S$, то предпочтение (3.3.2) естественно связать с выполнением $U_N \in S$.

Алгоритм поиска оптимального решения, таким образом, связывает следующие друг за другом решения. В простейшем случае

$$U_N = F(U_{N-1}); \qquad (3.3.3)$$

где F — алгоритм поиска, указывающий, какие операции следует сделать в одной точке (U_{N-1}), чтобы получить следующую (U_N), более предпочтительную (3.3.2).

В чем состоит специфика алгоритма F и почему его называют поисковым? Дело в том, что информации в точке U_{N-1} совершенно недостаточно для перехода в другую, лучшую точку U_N. Необходимо хотя бы иметь представление, как изменяется оптимизируемая функция $Q(U)$ и как ведут себя ограничения S в области точки U_{N-1}. Без этого нельзя составить разумный алгоритм F, обеспечивающий условие улучшения (3.3.2). Именно поэтому алгоритм F определения нового решения состоит обычно из двух частей. На первую возлагается задача получения информации о поведении функции $Q(U)$ и ограничений S, т. е. производится поиск информации. Вторая часть алгоритма F должна принимать решение, какое состояние U следует рекомендовать в качестве исходного для следующего этапа поиска.

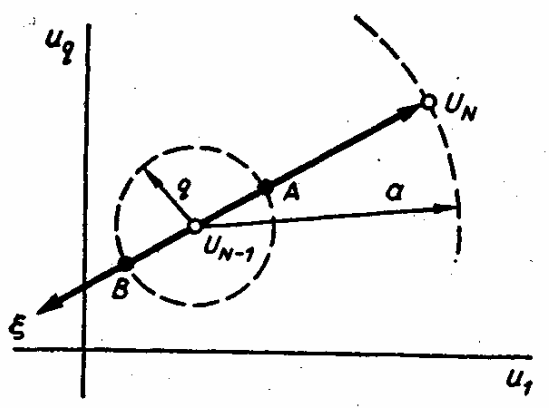
Таким образом, стандартный алгоритм F решения задачи оптимизации в сущности состоит из двух — алгоритма сбора информации и алгоритма принятия решения.

Заметим, что здесь говорится о стандартном алгоритме поиска. Возможны и отклонения от этой схемы, когда обе функции совмещены и неразделимы. Однако они обязательно сохраняются.

Приведем пример алгоритма случайного поиска, в котором указанные части выделены явно. Это так называемый алгоритм случайного поиска с парными пробами, применяемый обычно для решения задач без ограничений, т. е. при $S = R^q$. В нем процедура сбора информации сводится к выбору случайного направления ξ и определению значения показателя качества Q в двух точках $A = U_{N-1} - g\xi$ и $B = U_{N-1} + g\xi$ (ξ — единичный случайный вектор) (рис. 3.3.1).

Решение, которое принимается на базе этой информации, заключается в том, чтобы сделать шаг ΔU_N величиной a в направлении уменьшения (а в случае максимизации — в сторону увеличения) функции Q (см. рис. 3.3.1), т. е.

$$\Delta U_N = \pm a\xi, \qquad (3.3.4)$$



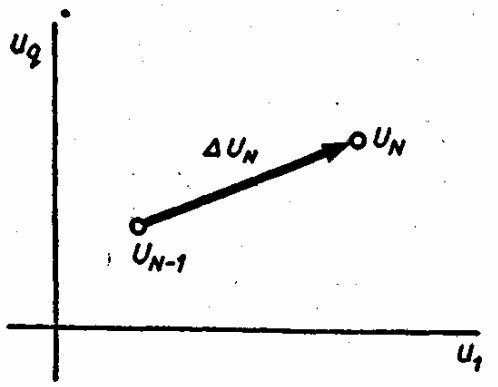


Рис. 3.3.1. Организация рабочего шага алгоритма случайного поиска с парными пробами.

Рис. 3.3.2. Шаг поиска в пространстве параметров.

где знак шага зависит от значений функции качества в точках A и B (так, на рис. 3.3.1 $Q(A) < Q(B)$).

В целом такой алгоритм записывается в виде

$$U_N = U_{N-1} - a\xi \operatorname{sign}\left[Q(U_{N-1} + g\xi) - Q(U_{N-1} - g\xi)\right], \qquad (3.3.5)$$

где sign — функция знака:

$$\operatorname{sign} z = \begin{cases} 1 & \text{при } z > 0; \\ 0 & \text{при } z = 0; \\ -1 & \text{при } z < 0. \end{cases} \qquad (3.3.6)$$

Как видно, здесь на первом этапе собирается информация о поведении функции качества в области точки U_{N-1}, для чего оценивается ее приращение в случайном направлении ξ. Радиус g определяет сферу сбора информации (на рис. 3.3.1 показана пунктиром). Сферу радиуса a естественно назвать сферой принимаемого решения (заметим, что величина рабочего шага a может быть и меньше, чем пробного g, что характерно для окончания процесса оптимизации). Свое название этот алгоритм случайного поиска получил от двух проб в точках A и B, доставляющих информацию, необходимую для принятия решения.

Продолжим рассмотрение случайного поиска как метода решения оптимизационных задач. Удобно алгоритм поиска представлять в виде правила вычисления приращения (шага) на каждом этапе поиска:

$$\Delta U_N = U_N - U_{N-1} \qquad (3.3.7)$$

(рис. 3.3.2). Алгоритм поиска F в этом случае определяет один шаг перехода от одного решения к другому, следующему за ним:

$$\Delta U = F(U_{N-1}). \qquad (3.3.8)$$

Здесь F — по-прежнему алгоритм поиска, т. е. правило, инструкция, указание, с помощью которого, находясь в точке U_{N-1}, можно определить смещение ΔU к другой, более предпочтительной точке ΔU.

Формально всякий алгоритм должен обладать свойством однозначности, т. е. при одинаковых исходных данных результат его работы должен быть одинаковым. Такое ограничение позволило построить довольно стройную теорию алгоритмов, хотя и сузило возможности этого понятия. Случайный поиск расширяет понятие алгоритма и снимает «проклятие детерминизма», допуская тем самым неоднозначность результата при одинаковых исходных данных.

Естественно подразделить все возможные алгоритмы поиска на два класса:

— детерминированные, регулярные алгоритмы поиска $\{F\}$, обладающие указанным свойством однозначности;

— недетерминированные (случайные, стохастические, вероятностные и т. д.) алгоритмы поиска $\{F_\xi\}$, не обладающие свойством однозначности, результат работы которых имеет статистический характер.

Легко показать, что

$$\{F\} \subset \{F_\xi\}, \qquad (3.3.9)$$

т. е. регулярные алгоритмы поиска являются частным, точнее — вырожденным случаем стохастических алгоритмов. Действительно, так как результатом работы стохастического алгоритма яри одних и тех же исходных данных может быть целое множество значений рабочего шага $\{\Delta U\}$, то всякий такой алгоритм устанавливает на этом множестве некоторое распределение вероятностей

$$F_\xi \to p(\Delta U), \qquad (3.3.10)$$

указывающее вероятностную меру каждого конкретного шага ΔU. При изменении алгоритма будет изменяться и распределение $p(\Delta U)$. В том случае, когда это распределение вырождается в δ-функцию, т. е. множество $\{\Delta U\}$ состоит из одного элемента, случайный поиск, ее порождающий, становится детерминированным. Нетрудно заметить, что этим случайный поиск обобщает регулярный, а любой детерминированный поиск является δ-вырождением хотя бы одного из алгоритмов случайного поиска.

3.3.2. Некоторые простейшие алгоритмы случайного поиска

Существует огромное число алгоритмов случайного поиска (из сказанного в конце предыдущего подраздела явствует, что их значительно больше, чем детерминированных). Рассмотрим наиболее характерные из них. Начнем с локальных алгоритмов, глобальным будет посвящен отдельный параграф (§ 3.6).

3.3.2.1. Случайный поиск с линейной тактикой

Случайный поиск такого рода построен с помощью только двух операторов: случайного шага (ξ) и повторения (+) предыдущего шага. Действие каждого из этих операторов может привести к одному из двух результатов: минимизируемая функция Q либо уменьшится ($\Delta Q<0$), либо не уменьшится ($\Delta Q\geq 0$). Здесь

$$\Delta Q_N = Q_N - Q_{N-1}, \qquad (3.3.11)$$

а $Q_N = Q(U_N)$ — значение минимизируемой функции на N-м этапе поиска. В зависимости от результата «включается» тот или иной оператор.

Алгоритм случайного поиска с линейной тактикой опирается на следующее очевидное предположение относительно объекта оптимизации: вероятность удачи ($\Delta Q<0$) в удачном ранее направлении больше, чем в случайном. Иначе говоря, целесообразно повторять удачные шаги, а при неудаче ($\Delta Q\geq 0$) делать случайный шаг, т. е. обращаться к оператору ξ.

Линейность тактики алгоритма заключается в имитации линейного поведения, т. е. в прямом повторении удачного шага. Такая тактика отличает живые существа (см. § 3.7, где рассмотрены бионические аспекты случайного поиска).

Алгоритм случайного поиска с линейной тактикой удобно изобразить в виде двух ориентированных графов переходов от одного оператора к другому в случае удачного и неудачного шагов (рис. 3.3.3). Как видно, при удаче — уменьшении минимизируемой функции — алгоритм повторяет (+) тот шаг ΔU, который привел к этой удаче. При неудаче ($\Delta Q\geq 0$) алгоритм

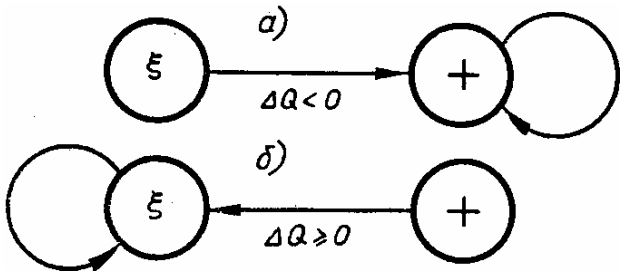


Рис. 3.3.3. Графы алгоритма случайного поиска с линейной тактикой: а — при удаче, *б* — при неудаче.

обращается к оператору случайности ξ и повторяет его до тех пор, пока не реализуется условие $\Delta Q<0$, т. е. до первого удачного случайного шага.

Этот алгоритм можно записать в рекуррентной форме:

$$\Delta U_N = \begin{cases} a\xi & \text{при} \quad \Delta Q_{N-1} \geqslant 0; \\ \Delta U_{N-1} & \text{при} \quad \Delta Q_{N-1} < 0, \end{cases} \quad (3.3.12)$$

где a — величина шага ($|\Delta U| = a$), а ξ — единичный ($|\xi| = 1$) случайный вектор, равномерно распределенный по всем направлениям пространства оптимизируемых параметров $\{U\}$. Это означает, что все направления случайного вектора ξ равновероятны.

Алгоритм можно изобразить и более компактно в виде одного графа, имеющего условные переходы (рис. 3.3.4, условия реализации переходов — рядом с дугами соответствующих переходов от одного оператора к другому).

Данный алгоритм имеет очень простую геометрическую интерпретацию. Это по сути дела спуск шагами a в случайном направлении ξ пространства параметров $\{U\}$: последовательное применение оператора повторения естественно интерпретировать как спуск по функции $Q(U)$ вдоль направления ξ с заданным шагом a до тех пор, пока значение функции не начнет увеличиваться. Это является сигналом, что выбранное направление исчерпано и надо выбрать новое, случайное. Отсюда и второе название — *алгоритм случайного спуска.*

Как видно, случайный фактор ξ в алгоритме вводится в момент неудачи ($\Delta Q \geq 0$) и, как в гомеостате Эшби, действует до тех пор, пока не будет найдено перспективное направление, после чего начинается спуск. Поскольку случай здесь связан с преодолением неудач и вводится как своеобразное «наказание» за неудачу, этот алгоритм иногда называют *случайным, поиском с наказанием случайностью.*

Теперь рассмотрим специфику и возможности описанного алгоритма. Прежде всего об области целесообразного применения. Пусть p — вероятность того, что случайный шаг $\Delta U = a\xi$ удачен, а q — вероятность повторения удачного шага, т. е.

$$p = \text{Вер}\,(\Delta Q < 0 \mid \Delta U = a\xi);$$

$$q = \text{Вер}\,(\Delta Q_N < 0 \mid \Delta Q_N < 0,\ \Delta X_N = \Delta X_{N-1}). \quad (3.3.13)$$

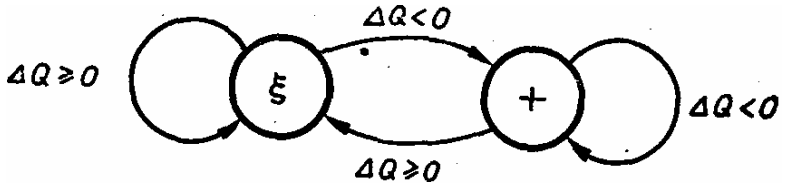
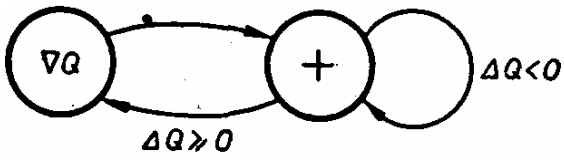


Рис. 3.3.4. Граф алгоритма случайного спуска с условными переходами.

Рис. 3.3.5. Граф алгоритма наискорейшего спуска.



Легко заметить, что описанный алгоритм целесообразно применять в таких ситуациях, когда часто действует оператор повторения «+», так как именно в эти моменты происходит уменьшение минимизируемой функции. Значит, для эффективной работы алгоритма необходимо, чтобы вероятность q была велика, а p — не мала. Например, для линейной функции $Q(U)$ имеем: $q = 1$ и $p = ½$, т. е. идеальные условия. Следовательно, указанный алгоритм целесообразно применять при оптимизации квазилинейных функций, т. е. вдали от экстремума U^* [275].

Любопытно сравнить этот алгоритм случайного поиска с известным методом наискорейшего спуска [95], который отличается от случайного поиска тем, что вместо оператора случайного шага | используется оператор вычисления градиента ∇:

$$\nabla = \left(\frac{\partial}{\partial u_1}, \ldots, \frac{\partial}{\partial u_q} \right) \quad (3.3.14)$$

минимизируемой функции, т. е. $\nabla Q(U)$. Граф алгоритма наискорейшего спуска приведен на рис. 3.3.5. Как видно, спуск в этом случае производится в антиградиентном направлении, т. е. в самом эффективном направлении минимизации для данной локальной ситуации. (Отсюда появилось ошибочное мнение, что лучшего метода, чем наискорейший спуск, и быть не может.) Сравним оба метода для случая, когда величина шага a стремится к нулю. Это означает, что минимум в процессе спуска определяется точно, т. е. решается задача минимизации

$$Q(U + \alpha\eta) \underset{\alpha>0}{\rightarrow} \min \Rightarrow \alpha^*, \quad (3.3.15)$$

где для случайного поиска $\eta = \xi$ — случайный вектор, а для наискорейшего спуска $\eta = -\nabla Q(U)$. Решение α^* этой задачи дает точку лучше исходной:

$$U_{N+1} = U_N + \alpha^*_N \eta_N, \quad (3.3.16)$$

где индекс N определяет номер этапа поиска.

Естественно сопоставить оба алгоритма для простейших объектов оптимизации. Таким простейшим объектом является квадратичная форма вида

$$Q(U) = c_1 u_1^2 + \ldots + c_q u_q^2. \quad (3.3.17)$$

Критерием сопоставления можно выбрать среднее значение отношения $Q(U_{N+1})$ / $Q(U_N)$. Очевидно, чем меньше это отношение, тем лучше алгоритм поиска. Для метода наискорейшего спуска оно имеет вид [95]

$$M\left(\frac{Q(U_{N+1})}{Q(U_N)}\right) \leqslant 1 - \frac{4}{\left(\sqrt{\rho} + \frac{1}{\sqrt{\rho}}\right)^2}. \qquad (3.3.18)$$

Здесь ρ — число обусловленности: $\rho = \max\limits_i c_i$ при $\min\limits_i c_i = 1$, а осреднение произведено по случайной начальной точке U_N. Для случайного спуска при $q = 2$ имеет место следующая оценка [145]:

$$M\left(\frac{Q(U_{N+1})}{Q(U_N)}\right) \leqslant 1 - \frac{1}{1+\sqrt{\rho}}, \qquad (3.3.19)$$

где M — знак математического ожидания по случайному направлению ξ.

Хорошо видно, что по крайней мере при $\rho > 16$ случайный спуск всегда будет эффективнее наискорейшего. Это означает, что плохо обусловленные задачи (их часто называют овражными) лучше решать методом случайного поиска. Действительно, чем больше ρ, тем хуже обусловленной и более «овражной» является задача. Случайный поиск всегда — даже для очень овражных задач — дает возможность минимизировать функцию (3.3.17) за один этап при любых начальных условиях. Антиградиентное движение при наискорейшем спуске исключает такую возможность. Преимущество данного алгоритма случайного поиска возникает за счет того, что он обладает большим спектром возможных направлений спуска, чем регулярные алгоритмы.

Теперь рассмотрим другой алгоритм случайного поиска, который построен в определенном смысле обратным образом. Здесь случайность вводится лишь при удачном шаге и является как бы поощрением. Такое поведение алгоритма нелинейно, что и послужило основанием для его наименования.

3.3.2.2. Случайный поиск с нелинейной тактикой

Этот вид поиска фактически модулирует метод проб и ошибок. Алгоритм построен из двух операторов — случайного шага (ξ) и оператора «—», причем, как уже говорилось, случайный шаг вводится в качестве реакции на предыдущую удачу, а при неудаче делается обратный шаг, т. е. реализуется оператор возврата.

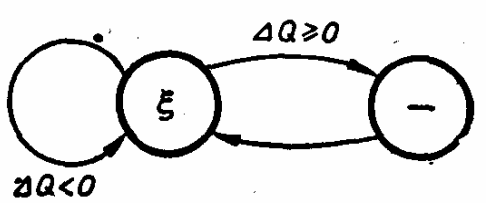


Рис. 3.3.6. Граф алгоритма случайного поиска с нелинейной тактикой.

Граф алгоритма с нелинейной тактикой показан на рис. 3.3.6. Как видно, он проще алгоритма с линейной тактикой. Здесь успех происходит за счет того, что используются только те случайные шаги, которые удачны, а неудачные устраняются (исправляются) с помощью операции возврата.

Рекуррентная формула алгоритма имеет вид

$$\Delta U_N = \begin{cases} a\xi & \text{при} \quad \Delta Q_{N-1} < 0; \\ -\Delta U_{N-1} & \text{при} \quad \Delta Q_{N-1} \geqslant 0. \end{cases} \tag{3.3.20}$$

Рассмотрим область целесообразного использования этого алгоритма. Элементарный анализ показывает, что его следует применять в ситуациях со значительной нелинейностью, когда нецелесообразно повторять удачные шаги, так как вероятность повторного успеха (3.3.13) мала ($q \approx 0$). Такой бывает ситуация в районе экстремума U^* или на дне оврага минимизируемой функции. Именно в этих случаях нелинейная тактика имеет существенное преимущество перед линейной.

Сравним этот алгоритм с известным детерминированным методом поиска — градиентным алгоритмом:

$$\Delta U_N = -\gamma \nabla Q(U_{N-1}), \tag{3.3.21}$$

где γ — определенный коэффициент, а $\nabla Q(U_{N-1})$ — градиент функции $Q(U)$ в точке U_{N-1} Будем сравнивать эти алгоритмы по удельным потерям на поиск, т. е. потерям «на единицу успеха». Потери обычно связаны с вычислением значения минимизируемой функции и их естественно измерять количеством вычислений этой функции. Так, для реализации одного шага случайного поиска необходимо вычислить ΔQ, т. е. дважды определить значение функции (3.3.11), следовательно, затраты будут равны двум единицам. Каждый шаг по методу градиента (3.3.21) требует $q+1$ вычислений значений функции Q. Действительно, для оценки частных производных градиента (3.3.14) необходимо вычислить

$$\frac{\widehat{\partial Q}}{\partial u_i} = \frac{1}{g}[Q(U+ge_i) - Q(U)] \qquad (i=1, \ldots, q), \tag{3.3.22}$$

где g — база оценки, а e_i — i-й орт. Очевидно, что успех одного этапа обоих

алгоритмов будет разным: успех случайного поиска,

разумеется, будет меньше, чем градиентного, который сделает шаг в лучшем (антиградиентном) направлении.

Разделив потери на успех, получим удельные потери на поиск (K), поведение которых в функции размерности задачи q показано на рис. 3.3.7 для обоих алгоритмов. Хорошо видно, что при решении простых задач, т. е. при $q<q^*$, более целесообразно применять метод градиента (МГ), а для задач большой размерности ($q>q^*$) лучше случайный поиск (СП), у которого потери на поиск пропорциональны $\sqrt{q}$. Любопытно, что критическая сложность q^*, разграничивающая оба метода, невелика ($q^* = 3$—5). Для линейного объекта $q^* = 3$ [161].

Интересно рассмотреть влияние близости к экстремуму U^* на сравнительную эффективность обоих алгоритмов. Пусть r — расстояние до экстремума $r = |U - U^*|$. Тогда на плоскости параметров r и q имеет место разделение на зоны целесообразного использования алгоритмов по критерию удельных потерь на поиск, показанное на рис. 3.3.8. Из рисунка видно, что зона случайного поиска достаточно велика и его лучше использовать для приближенного решения сложных задач. Уточнять решение ($r\to 0$) следует методом градиента. Процесс оптимизации заключается в движении по вертикали (см. рис. 3.3.8) с переходом из зоны случайного поиска (СП) в зону метода градиента (МГ).

Таким образом, алгоритм случайного поиска с нелинейной, тактикой является эффективным методом оптимизации многопараметрических задач. Его эффективность падает с ростом размерности пропорционально $1/\sqrt{q}$, а не $1/q$, как у регулярных методов. Это делает случайный поиск надежным методом решения сложных задач.

Выше уже говорилось, что случайный поиск можно рассматривать как способ сбора информации о поведении минимизируе-

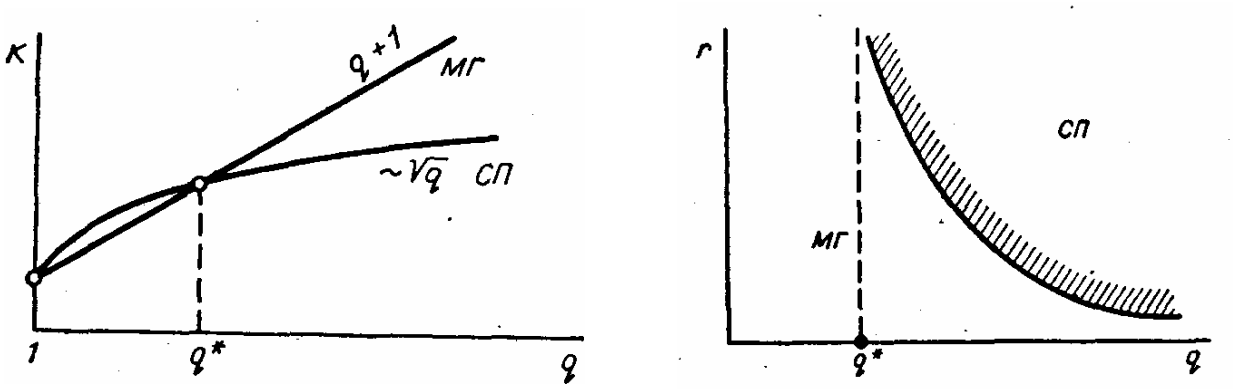


Рис. 3.3.7. Поведение потерь на поиск при изменении размерности оптимизируемого объекта. СП — случайный поиск (с нелинейной тактикой), МГ — метод градиента.

Рис. 3.3.8. Области целесообразного применения алгоритмов случайного поиска (СП) и метода градиента (МГ).

мой функции. Такой информацией облагает оценка градиента, указывающего направление наиболее интенсивного увеличения функции. Поэтому естественно рассмотреть алгоритмы случайного поиска, использующие стохастические оценки градиента. Рабочий шаг при этом организуется так же, как и у градиентного алгоритма (3.3.21):

$$\Delta U = -a\hat{\nabla}_{\xi} Q(U), \quad (3.3.23)$$

где $\hat{\nabla}_{\xi}$ — стохастическая оценка градиента. Способ такой оценки и отличает алгоритм поиска.

3.3.2.3. Случайный поиск по наилучшей пробе

Этот алгоритм сводится к определению значений минимизируемой функции в m случайных точках:

$$U_i = U + g\xi_i \quad (i = 1, ..., m), \quad (3.3.24)$$

где ξ_i — i-я реализация единичного случайного, вектора, равномерно распределенного в пространстве $\{U\}$. Выбор направления ξ^* наилучшей пробы определяется очевидным выражением:

$$Q(U+g\xi^*) = \max_{i=1,...,m} Q(U+g\xi_i) \quad (3.3.25)$$

и дает оценку градиента в виде

$$\hat{\nabla}_{\xi} Q(U) = \frac{\xi^*}{g}[Q(U+\xi^* g) - Q(U)]. \quad (3.3.26)$$

Очевидно, что при $m \to \infty$ эта оценка стремится к точному значению градиента, т. е.

$$\lim_{m\to\infty} \hat{\nabla}_{\xi} Q(U) = \nabla Q(U).$$

3.3.2.4. Метод стохастического градиента

Стохастический градиент оценивается по формуле [164]:

$$\hat{\nabla}_{\xi} Q(U) = \frac{q}{mg} \sum_{i=1}^{m} [Q(U+g\xi_i) - Q(U)]\xi_i, \quad (3.3.27)$$

т. е. представляет собой сумму всех случайных векторов ξ_i с весами, равными приращениям минимизируемой функции в данных случайных направлениях.

Если в качестве ξ_i взять орты, т. е. $\xi_i = e_i$, то эта оценка при $m = n$, как легко заметить из (3.3.22), дает точное значение градиента.

Обе описанные оценки градиента могут эффективно применяться при любых значениях m, в том .числе и при $m<n$, что существенно отличает их от детерминированного способа оценивания (3.3.22), для которого строго $m = n$. Это же обстоятельство подтверждает, что детерминированные методы обобщаются случайными (см. конец подраздела 3.3.1). Приведем еще пример такого обобщения.

Градиентный поиск (3.3.21) является частным случаем по крайней мере двух алгоритмов случайного поиска. Первый алгоритм:

$$\Delta U_N = -\gamma \hat{\nabla} Q(U_{N-1}) + l\xi,$$

где ξ — по-прежнему единичный случайный q-мерный вектор. Это известный градиентный алгоритм случайного поиска [145, 247]. Второй алгоритм имеет вид (3.3.23), но оценка градиента вычисляется по формуле

$$\hat{\nabla} Q(U_N) = \sum_{i=1}^{q} [Q(U_{N-1} + ge_i + l\xi_i) - Q(U_{N-1})] \frac{(ge_i + l\xi_i)}{|ge_i + l\xi_i|^2}.$$

При l=0, как легко заметить, оба алгоритма вырождаются. в градиентный алгоритм поиска (3.3.21).

Таким образом, случайный поиск является естественным расширением, продолжением и обобщением известных регулярных методов поиска.

Другой особенностью случайного поиска, которая открывает широкие возможности для его эффективного применения, является использование оператора случайного шага ξ в качестве стохастической модели сложных регулярных операторов отыскания направлений поиска в пространстве оптимизируемых параметров $\{U\}$.

Так, алгоритм случайного поиска с линейной тактикой (3.3.12) является стохастической моделью алгоритма наискорейшего спуска:

$$\Delta U_{N+1} = \begin{cases} a\hat{\nabla} Q(U_N) & \text{при } \Delta Q_N \geqslant 0; \\ \Delta U_N & \text{при } \Delta Q_N < 0, \end{cases}$$

в которой случайный вектор ξ моделирует оценку градиента. Любопытно, что подобную «оценку» нельзя даже назвать грубой, так как ее стохастические свойства и не напоминают свойств оцениваемого градиента. Однако, как показано выше, алгоритм случайного поиска не только работоспособен, но в ряде случаев и более эффективен, чем алгоритм наискорейшего спуска. Здесь

оператор случайного шага ξ заменяет громоздкий оператор оценки градиента, например, по формуле (3.3.22).

Следующей особенностью случайного поиска, выгодно отличающей его от регулярных методов, является глобальность, проявляющаяся прежде всего в локальных алгоритмах случайного поиска, не предназначенных для отыскания глобального экстремума. Так, алгоритм случайного спуска может найти глобальный экстремум, а регулярный алгоритм наискорейшего спуска в принципе не допускает такой возможности, поскольку он построен для отыскания локального экстремума.

Следовательно, глобальность алгоритмов случайного поиска является как бы «премией» за использование случайности и чем-то вроде «бесплатного приложения» к алгоритму. Это обстоятельство особенно важно при оптимизации объектов с неизвестной структурой, когда нет полной уверенности в одноэкстремальности задачи и возможно (хотя и не ожидается) наличие нескольких экстремумов. Использование в таком случае методов глобального поиска было бы неразумной расточительностью. Методы локального случайного поиска здесь наиболее приемлемы, так как они эффективно решают локальную задачу и могут в принципе решить глобальную, если таковая будет иметь место. Это обеспечивает случайному поиску своеобразную психологическую надежность, которую очень ценят пользователи.

Алгоритмическая простота случайного поиска делает его привлекательным в первую очередь для потребителей [229]. Опыт показывает, что известные алгоритмы случайного поиска являются лишь «канвой», на которой пользователь в каждом конкретном случае «вышивает узоры» новых алгоритмов, отражающих не только его вкусы и наклонности (что нельзя не учитывать), но и специфику оптимизируемого объекта. Последнее создает благоприятную основу для реализации известного принципа, что алгоритм должен конструироваться «под объект». Наконец, алгоритмическая простота случайного поиска обусловливает простоту его аппаратурной реализации. Это не только дает возможность строить простые, компактные и надежные оптимизаторы с неограниченным числом оптимизируемых параметров [183], но и позволяет довольно просто организовать их оптимальный синтез на ЭВМ [142].

3.3.3. Автоматные алгоритмы случайного поиска

В основе этих алгоритмов лежит понятие автомата. Рассмотрим это понятие подробнее. Автоматом традиционно называют пятерку

$$A = \langle Q, W, V, \mu(q,w), \nu(q,w) \rangle, \qquad (3.3.28)$$

где $Q = \{q_1, ..., q_k\}$ — алфавит входов; $W = \{w_1, ..., w_m\}$ — алфавит внутренних состояний автомата; $V = \{v_1, ..., v_l\}$ — алфавит его выходов; $w' = \mu(q, w)$ — функция переходов, определяющая, в какое состояние w' переходит автомат из состояния w под действием входного сигнала q; $v = v(q, w)$ — функция выхода — определяет значение выхода v автомата, находящегося в состоянии w при входе q. Эти функции могут быть стохастическими, тогда и автомат будет стохастическим. Если автомат находится в какой-то среде, то v представляет собой его воздействие на среду, а q — воздействие среды на автомат.

Представим алгоритм поиска в виде такого автомата. Средой в этом случае является объект оптимизации, преобразующий оптимизируемые параметры U в значение функции $Q(U)$. Входом автомата поиска является значение Q (или приращение ΔQ), а выходом будет U (или шаг ΔU).

Введем следующие упрощающие условия:

1. Автомат поиска $\Delta U = F(U)$ декомпозируется на q автоматов $\Delta u_i = F_i(U)$ $(i = 1, ..., q)$, каждый из которых воздействует на один оптимизируемый параметр

$$u_{i, N+1} = u_{iN} + \Delta u_{i, N+1}. \qquad (3.3.29)$$

2. Алфавит входов автомата поиска имеет лишь два $(k = 2)$ знака:

$$Q = \{q_1, q_2\},$$

где $q_1 = 0$ («нештраф») соответствует $\Delta Q < 0$, а $q_2 = 1$ («штраф») соответствует $\Delta Q \geq O$.

3. Алфавит выходов также имеет два знака:

$$V_i = \{v_{i1}, v_{i2}\},$$

где $v_{i1} = a_i > 0$ и $v_{i2} = -a_i$, т. е. каждый автомат делает шаг величиной a_i в одну или другую сторону. Пусть для простоты $a_i = 1$
и все автоматы одинаковы, т. е. индекс i можно снять. Теперь алгоритм случайного поиска задается видом функций μ и v. Здесь целесообразно, различать два варианта:

1) μ — стохастическая функция, v — детерминированная;

2) μ — детерминированная функция, v — стохастическая.

Первый случай соответствует направлению оптимизации с помощью коллектива стохастических автоматов с целесообразным поведением [140], второй — случайному поиску с самообучением [181]. Рассмотрим каждое направление отдельно.

3.3.3.1. Коллектив оптимизирующих автоматов с целесообразным поведением [140, 141]

Такой . коллектив автоматов характеризуется следующими функциями.

Функция выхода v определяется простым выражением

$$v = \begin{cases} v_1 \text{ при } w \in \{w_{m/2+1}, \ldots, w_m\}; \\ v_2 \text{ при } w \in \{w_1, \ldots, w_{m/2}\}. \end{cases} \quad (3.3.30)$$

Значит, в первых $m/2$ состояниях автомата *(m* четное) выход автомата поиска равен v_2=-1 (значение соответствующего оптимизируемого параметра уменьшается), а в остальных v_2=1 (оптимизируемый параметр увеличивается).

Функция переходов $\mu(q, w)$ в этом случае является стохастической и задается двумя стохастическими матрицами вида

$$P = \begin{cases} P^0 = \|p_{ij}{}^0\| \text{ при } q = q_1 \quad (\text{«нештраф»}); \\ P^1 = \|p_{ij}{}^1\| \text{ при } q = q_2 \quad (\text{«штраф»}). \end{cases} \quad (3.3.31)$$

Эти матрицы определяют вероятность p_{ij} переходов из одного состояния w_i в другое w_j при той или иной реакции на рабочий шаг. Обычно матрицы диагональны, т. е. граф переходов имеет вид, показанный на рис. 3.3.9. Здесь значения вероятностей $p > 1/2$ и $q > 1/2$ однозначно определяют матрицы. Как видно, при $\Delta Q < 0$, т. е. в случае удачи, автомат «старается» закрепить то действие, которое привело к удаче, и наоборот, при $\Delta Q \geq 0$ он стремится к смене неудачного действия на обратное. Таково свойство обоих графов на рис. 3.3.9 при $p, q > 1/2$.

Естественно возникает мысль: а не лучше ли сделать автомат детерминированным, т. е. положить $p = q = 1$? Ведь тогда перестройка автомата с одного действия на другое будет происходить

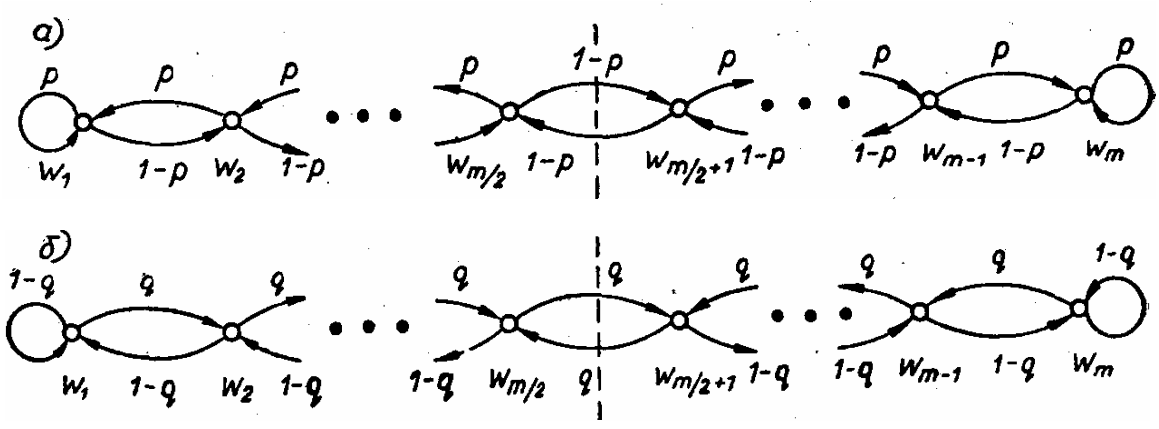


Рис. 3.3.9. Графы поведения автоматного алгоритма случайного поиска при удаче (а — $x = 0$) и неудаче (*б* — $x = 1$).

с максимальной скоростью! Однако эксперименты показывают, что в этом случае автоматы «зацикливаются» и процесс оптимизации останавливается. Значит, условие

$$\frac{1}{2}<p, q<1 \tag{3.3.32}$$

является необходимым для эффективной работы поиска, т. е. без элемента случайности оптимизация в данном случае просто невозможна.

Параметры p, q и m каждого автомата, очевидно, определяют эффективность его работы в различных условиях. Так как нет правила выбора этих параметров, то для их определения следует применить методы адаптации и, в частности, эволюционную адаптацию (см. шестую главу настоящей книги).

3.3.3.2. Автоматный случайный поиск с самообучением [181]

Автоматный поиск с самообучением отличается тем, что функция μ детерминированна, т. е. является регулярной зависимостью нового состояния w' от результата x и предыдущего состояния w. Ее можно представить, например, в виде двух детерминированных графов, показанных на рис. 3.3.10.

Функция выхода $v(\bullet,\bullet)$ автомата поиска здесь случайная и задается, как и (3.3.30), на состояниях w_j $(j = 1, ..., m)$ в виде вероятности шага в положительном направлении:

$$v=\begin{cases} v_1 & \text{с вероятностью } 1-\psi(j); \\ v_2 & \text{с вероятностью } \psi(j), \end{cases} \tag{3.3.33}$$

где $\psi(j)$ — монотонно-возрастающая функция, например, вида

$$\psi(j)=\frac{j-1}{m-1} \quad (j=1, \ldots, m). \tag{3.3.34}$$

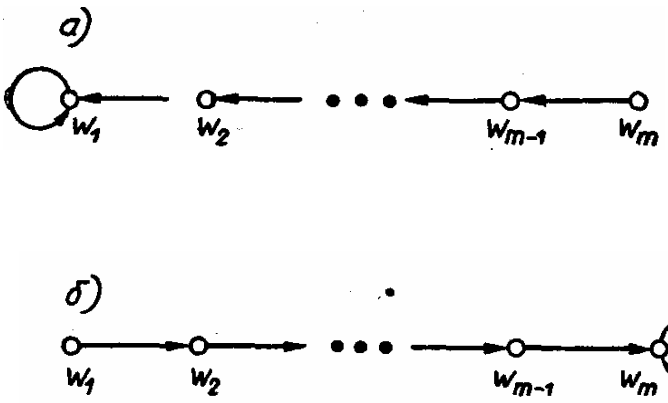


Рис. 3.3.10. Графы алгоритма случайного поиска с самообучением при различных ситуациях: *а* — $\Delta u_i \Delta Q \geq 0$, *б* — $\Delta u_i \Delta Q < 0$.

Термин «самообучение» в данном случае используется потому, что аналогичные алгоритмы описывают процесс самообучения живых организмов (опыты с мышью в лабиринте и т. п.). Более подробно такие бионические алгоритмы случайного поиска, моделирующие поведение живых существ, будут рассмотрены в §3.7.

Оба описанных алгоритма случайного поиска эффективно работают при оптимизации сложных объектов. Их эффективность в одинаковых условиях одинакова. Однако коллектив оптимизирующих автоматов еще имеет возможность эволюционировать (см §3.7).

В заключение отметим, что здесь описаны лишь основные алгоритмы локального случайного поиска, с помощью которых обычно создают его многочисленные модификации, приспособляемые для решения конкретных задач оптимизации сложных систем.

§ 3.4. Учет ограничений в процессах случайного поиска

Проблема многопараметрической оптимизации

$$Q(U) \to \min_{U \in S} \qquad (3.4.1)$$

в реальных задачах всегда имеет условный характер, т. е. связана с обязательным выполнением ограничений S, имеющих прежде всего ресурсный характер. Разные методы решения этой условной задачи различаются в значительной степени способами выполнения условия S, т. е. учета ограничений. Методы, связанные со сведением условной задачи (3.4.1) к безусловной, т. е. методы типа штрафных функций, нивелируют специфику ограничений и приводят к появлению других, не менее существенных трудностей типа овражности и многоэкстремальности штрафной функции. Рассмотрим поэтому методы, учитывающие и выявляющие специфику и особенность ограничений [169].

Случайный поиск как метод решения условной задачи (3.4.1) отличается рядом преимуществ по сравнению с детерминированными методами. Здесь у случайного поиска имеется ряд возможностей, связанных со случайным характером поиска, которых в принципе не может иметь ни один детерминированный метод решения задачи (3.4.1).

Представляют интерес те аспекты случайного поиска, которые открывают ему новые возможности для учета ограничений S

по сравнению с регулярными методами. Именно этому посвящен данный параграф.

Прежде всего рассмотрим различные виды ограничений. Они могут иметь троякий характер.

3.4.1. Типы ограничений

Во-первых, могут быть ограничены какие-то определенные функции от оптимизируемых параметров $U = (u_1, ..., u_q)$:

$$S_H : h_i (U) > 0 \quad (i = 1, ..., m), \tag{3.4.2}$$

где h_i (•) — заданные скалярные функции, которые удобно представить в виде векторной функции

$$H (U) = (h_1 (U), ..., h_m(U)). \tag{3.4.3}$$

Тогда ограничения типа неравенства (3.4.2) записываются в виде

$$S_H : H (U) \geq 0. \tag{3.4.4}$$

Во-вторых, ограничения могут иметь характер равенств:

$$S_G : g_j (U) = 0 \quad (j = 1,..., n<q), \tag{3.4.5}$$

где g_j(•) — заданные скалярные функции. Для. удобства введем векторную функцию

$$G (U) = (g_1 (U), ..., g_n (U)), \tag{3.4.6}$$

с помощью которой ограничения типа равенств записываются в виде

$$S_G : G (U) = 0. \tag{3.4.7}$$

И наконец, ограничения могут быть связаны с дискретностью ряда функций от оптимизируемых параметров:

$$S_D : f_Z (U) \in \{f_Z^{(1)}, \ldots, f_Z^{(k_Z)}\} \qquad (Z = 1, \ldots, p), \tag{3.4.8}$$

где f_Z (•) — заданные функции, а $f_Z^{(l)}$ — l-е заданное значение, которое может принимать Z-я функция. В частном случае при. $f_Z (U) = u_Z$ получаем

$$S_D : u_Z \in \{u_Z^{(1)}, \ldots, u_Z^{(k_Z)}\}, \tag{3.4.9}$$

где $u_Z^{(l)}$ — значения (например, целочисленные), которые может принимать Z-я переменная. Введем обозначения

$$F (U) = (f_1(U), ..., f_p (U)). \tag{3.4.10}$$

Тогда «дискретные» ограничения можно записать в виде

$$S_D: F(U) \in D, \quad (3.4.11)$$

где D — заданное конечное множество значений, которые может принимать векторная функция $F(U)$.

Очевидно, что область поиска S может быть образована путем различных комбинаций пересечения областей S_H, S_G и S_D — при условии, разумеется, что она будет содержать хотя бы один элемент (так, пересечение S_G и S_D может не иметь ни одного элемента). Поэтому разумными комбинациями в общем случае являются лишь две.

Первая —

$$S = S_H \cap S_G, \quad (3.4.12)$$

как известно, связана с непрерывными задачами математического программирования, а вторая —

$$S = S_H \cap S_D \quad (3.4.13)$$

— с задачами дискретного программирования. В случае, когда различные типы ограничений связаны с различными группами переменных, допустимы любые комбинации ограничений.

Рассмотрим специфику процессов случайного поиска при учете ограничений различного рода. Методы типа штрафных функций [225] рассматривать не будем, хотя и при этом появление овраж-ности и многоэкстремальности преодолевается методом случайного поиска эффективнее и проще, чем детерминированными методами (см. § 3.6).

3.4.2. Случай $S = S_H$

Ограничения типа неравенств образуют обычно область S размерности q. Выход за границу области $(U \notin S)$ является сигналом о необходимости учета ограничений S. При случайном поиске это можно осуществить множеством способов. Рассмотрим наиболее эффективные из них.

3.4.2.1. Использование возврата

Этот способ отличается тем, что нарушение ограничений $(U \notin S)$ отождествляется с точки зрения поиска с неубыванием показателя качества ($\Delta Q \geq 0$), для чего вводится оператор воз-

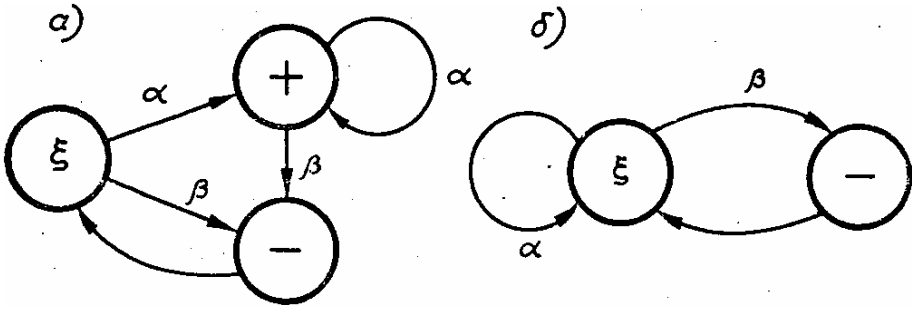


Рис. 3.4.1. Графы алгоритмов поиска с линейной (а) и нелинейной (*б*) тактикой: «ξ» — оператор случайного шага, «+» — оператор «так же» ($\Delta U_{N+1} = \Delta U_N$), «—» — оператор возврата ($\Delta U_{N+1} = -\Delta U_N$).

врата. Таким образом, в процессе поиска различаются лишь две ситуации — удачный и неудачный шаг:

$$\alpha: (U \in S) \wedge (\Delta Q < 0);$$
$$\beta: (U \notin S) \vee (\Delta Q \geqslant 0), \qquad (3.4.14)$$

т. е. удачным шагом считается случай (α), когда ограничения не нарушены и одновременно уменьшился показатель качества. Неудачным шагом (β) считается такой, при котором нарушены ограничения или увеличился показатель качества.

Реакцией на β в этом случае является возврат в предыдущее состояние ($\Delta U_{N+1} = -\Delta U_N$). На рис. 3.4.1 показаны графы двух алгоритмов поиска с линейной (*а*) и нелинейной (*б*) тактикой, реализующие этот подход. Работоспособность их очевидна [302].

3.4.2.2. Использование самообучения в виде адаптации распределения случайного шага

Этот способ также позволяет учитывать ограничения типа неравенств. Пусть W — вектор памяти:

$$W = M(\xi), \qquad (3.4.15)$$

где M — знак математического ожидания, а ξ — случайный шаг. Это означает, что плотность распределения $p(\xi)$ случайных шагов ξ не равномерна и имеет «снос» W. Тогда адаптация этого сноса в форме

$$w_{N+1} = kw_N + \gamma\delta\Delta U_{N+1}, \qquad (3.4.16)$$

где $0 \leq k \leq 1$ — параметр забывания, $\delta > 0$ — параметр скорости

самообучения, а γ — фактор, учитывающий удачность случайного шага:

$$\gamma = \begin{cases} +1 \text{ при } \alpha; \\ -1 \text{ при } \beta, \end{cases} \quad (3.4.17)$$

решает задачу учета ограничений $S=S_H$. Как легко заметить, формула (3.4.16) является естественным обобщением самообучения случайного поиска (см. § 3.5) при нарушении ограничений S.

3.4.2.3. Адаптация величины шага

Такая адаптация (см. также § 3.5) в алгоритме с нелинейной тактикой (3.3.20) позволяет получить алгоритм, сходящийся при наличии ограничений и отсутствии помех. Он имеет вид

$$U_{N+1} = \begin{cases} a_N \xi_N \text{ при } \alpha; \\ -\Delta U_N + a_N \xi_N \text{ при } \beta. \end{cases} \quad (3.4.18)$$

Алгоритм адаптации

$$a_{N+1} = \begin{cases} \gamma_1 a_N \text{ при } \alpha; \\ \gamma_2 a_N \text{ при } \beta, \end{cases} \quad (3.4.19)$$

где $\gamma_1>1$, $\gamma_2<1$, причем $\gamma_1\gamma_2>1$ (вопрос выбора параметров γ_1 и γ_2 рассмотрен подробно в работе [214]).

3.4.3. Случай $S=S_G$

Учет ограничений типа равенств в процессе случайного поиска связан с организацией движения вдоль указанных ограничений. Его можно осуществить путем воздействия на вероятностные свойства случайного шага: организовать изменение распределения $p(\xi)$ так, чтобы случайные шаги ξ производились преимущественно в окрестности области S_G, размерность которой равна $q - n$.

Такую задачу решает, например, следующий способ. Пусть $U_i \in S_G$ ($i = 1, \ldots, q - n + 1$) и $Q(U_i) \geq Q(U_{i+1})$. Образуем линейное подпространство размерности $q — n$, натянутое на эти точки:

$$U = \sum_{i=1}^{q-n+1} \lambda_i U_i, \quad (3.4.20)$$

где. ($\sum_{i=1}^{q-n+1}\lambda_i = 1$) случайную точку U_ξ будем выбирать в этом пространстве — например, на расстоянии a от U_{n-q+1}, т. е.

$$| U_\xi - U_{n-q+1} | = a. \tag{3.4.21}$$

Теперь задача определения U_ξ сводится к решению системы из двух уравнений относительно $q - n + 1$ параметров λ_i:

$$\left| \sum_{i=1}^{q-n+1} \lambda_i U_i - U_{n-q+1} \right| = a;$$

$$\sum_{i=1}^{q-n+1} \lambda_i = 1, \tag{3.4.22}$$

где λ_i — случайные числа, например, распределенные равномерно в пределах $-1 \le \lambda_i \le 1$ ($i = 1, ..., q - n + 1$). Полученная таким образом (3.4.20) новая точка U'_{q-n+2} может, естественно, не принадлежать S. В этом случае ее следует спроектировать на S любым локальным методом — например, градиентным методом (3.3.21):

$$U_{q-n+2} = \mathrm{пр}_S U'_{q-n+2} \in S,$$

3.4.23) т. е. решить минимизационную задачу

$$| G(U'_{q-n+2} + \Delta U_{q-n+2}) | \to \min_{\Delta U_{q-n+2}} \Rightarrow \Delta U^*_{q-n+2}, \tag{3.4.24}$$

решение которой ΔU^*_{q-n+2} и дает искомую точку

$$U_{q-n+2} = U'_{q-n+2} + \Delta U^*_{q-n+2}. \tag{3.4.25}$$

Варьируя распределения $p_i(\lambda_i)$ в системе уравнений (3.4.22), можно изменять вероятностные свойства случайной точки U'_{q-n+2}.

Теперь из $q - n + 2$ точек отбрасывается худшая по критерию оптимизации, а оставшиеся $q - n + 1$ точки переиндексируются так, чтобы лучшей была U_{a-n+1}, и т. д.

Можно предложить и другие способы образования случайных точек на S_G. Одним из них является способ простого проецирования на S_G случайных точек, который был использован в овражном методе Гельфанда и Цетлина [55].

3.4.4. Случай $S = S_H \cap S_G$

Учет такого рода ограничений в процессах случайного поиска не представляет труда, так как осуществляется путем прямого комбинирования методов, используемых в двух рассмотренных выше случаях.

3.4.5. Случай $S = S_D$

При решении дискретных задач оптимизации очень естественно использовать метод случайного поиска. Это связано с тем, что всякого рода градиентные представления, свойственные детерминированным методам, в дискретном случае теряют смысл.

Простейшая схема случайного поиска здесь опирается на случайный выбор точки в ε-окрестности исходной. Пусть ε-окрестность исходной точки U_N имеет вид

$$| U - U_N | \leq \varepsilon \qquad (3.4.26)$$

и пусть для простоты множество S_D образовано целочисленными векторами U. Иначе говоря, принимаем, что все координаты векторов U имеют целочисленные значения (более общий случай легко сводится к этому). Пусть D^{ε}_N — множество целочисленных точек, попавших в ε-окрестность, т. е. удовлетворяющих условию (3.4.26). Например, при ε=1 таких точек будет $2n$ и образуются они изменением каждой из координат U_N на ± 1.

Тогда процедура случайного поиска на (N+1)-м шаге будет связана со случайным выбором такой точки множества D^{ε}_N, для которой выполняются очевидные условия

$$Q\,(U_{N+1}) < Q\,(U_N); \qquad (3.4.27)$$

$$U_{N+1} \in D^{\varepsilon}_N.$$

Введем случайное целочисленное единичное смещение ξ:

$$U_{N+1} = U_N + \xi. \qquad (3.4.28)$$

Например, для ε = 1 вектор смещения ξ будет

$$\xi = (\xi_1, \ldots, \xi_q), \qquad (3.4.29)$$

где лишь одна случайная компонента равна ±1, а остальные равны нулю, т. е.

$$\sum_{i=1}^{q} |\xi_i| = 1, \quad \xi_i \in \{0; 1\}. \qquad (3.4.30)$$

Можно сделать векторы ξ не равновероятными и изменять их вероятности в соответствии с выбранным законом самообучения — например, увеличивая вероятность тех векторов смещений, которые на предыдущем шаге дали уменьшение критерия качества (с последующим нормированием, разумеется). Процесс самообучения здесь естественно дополнить условием запоминания уже проверенных точек, с тем чтобы не повторять проверку условий (3.4.27) в одной и той же точке.

Критерием остановки процесса является отсутствие такой точки при достаточно большом ε. Легко заметить, что при $\varepsilon \to \infty$ метод вырождается в обычный случайный перебор. Поэтому величина ε не должна быть слишком большой и в начале поиска $\xi \approx 1$.

Очевидно, что этот процесс можно варьировать в широких пределах — например, изменять характер меры в неравенстве (3.4.26) или адаптировать ее в процессе поиска. Большинство существующих эффективных методов решения дискретных задач оптимизации в той или иной степени использует изложенный подход (см., например, [109, 200]).

3.4.6. Случай $S = S_D \cap S_H$

Учет ограничений типа неравенств в задаче дискретной оптимизации незначительно усложняет процедуру случайного поиска. В этом случае к условиям (3.4.27) добавляется еще одно:

$$U_{N+1} \in S_H, \tag{3.4.31}$$

которое легко проверяется.

§ 3.5. Адаптация алгоритмов случайного поиска 3.5.1.

Анализ задачи адаптации поиска

Имеются два источника причин, вызывающих необходимость адаптации алгоритмов поиска, — появление новых ситуаций и возникновение новых задач. Рассмотрим их подробнее.

Многочисленные ситуации, складывающиеся в процессе оптимизации одного и того же объекта, — типа «холм», «яма», «плоскогорье», «хребет», «овраг» и т. д. — заставляют искать средства такой перестройки алгоритма, чтобы эти ситуации преодолевались с минимальными затратами. Подобная перестройка, реализуемая формально, и является адаптацией алгоритма поиска. Если затраты на адаптацию невелики, то целесообразность и

эффективность такой процедуры очевидны. Это один источник адаптации.

Другой источник связан с потребностью решать различные задачи оптимизации. Обилие алгоритмов оптимизации обусловлено прежде всего тем, что каждый новый алгоритм предлагается для решения какой-то новой задачи, отличающейся от предыдущих. Совершенно ясно, что такая тенденция порочна и недолговечна, в силу чего и возникает необходимость в адаптации известного алгоритма поиска к новой задаче оптимизации. Пользователь часто применяет известные алгоритмы для решения своих задач, пытаясь как-то приспособить их к этим задачам. Успех здесь целиком зависит от того, насколько пользователь знает структуру своей оптимизационной задачи и понимает механизм работы поиска. Как правило, ничего хорошего из такой «кустарной адаптации» не получается. Это и заставляет создавать специальные процедуры адаптации алгоритмов оптимизации, благодаря которым алгоритмы могут эффективно действовать в изменяющихся условиях. Как уже говорилось, существуют два способа изменения алгоритма поиска — воздействие на его параметры и изменение его структуры. Параметрическая адаптация поиска не всегда может значительно повысить эффективность -поиска. Более того, есть алгоритмы — например, метод наискорейшего спуска или метод Гаусса—Зейделя, которые вообще не имеют параметров и поэтому не могут адаптироваться параметрически. Эти обстоятельства обусловливают переход к адаптации структуры алгоритма.

Таким образом, задача адаптации процесса поиска ставится тогда, когда алгоритм нужно изменять в процессе поиска, чтобы поддерживать его эффективность на необходимом уровне.

Эта задача всегда возникает при оптимизации и адаптации сложных объектов, для которых нельзя заранее предугадать, в какую ситуацию попадет алгоритм поиска. Такого рода неопределенность и требует введения процедуры адаптации, т. е. приспособления алгоритма поиска к новой ситуации путем его целенаправленного изменения, коррекции параметров или структуры.

3.5.2. Параметрическая адаптация алгоритмов случайного поиска

Основными параметрами алгоритма случайного поиска являются величина рабочего шага ΔU (см. формулы (3.3.12), (3.3.20) и (3.3.23)) и параметры плотности распределения $p(\xi)$ случайного шага ξ. Заметим, что случайный поиск отличается от любого, детерминированного метода именно наличием такого распределения, изменение которого позволяет адаптировать случайный поиск.

Это качество выгодно отличает случайный поиск от регулярных алгоритмов, не имеющих такого «рычага управления» процессом поиска.

Рассмотрим адаптацию по каждому из указанных факторов отдельно.

3.5.2.1. Адаптация величины рабочего шага

Такая адаптация связана с необходимостью уменьшить величину шага по мере приближения к положению экстремума U^*. Программные алгоритмы изменения рабочего шага (типа условий Дворецкого: $\sum_{i=1}^{\infty} a_i = \infty; \sum_{i=1}^{\infty} a_i^2 < \infty$ [237]) не учитывают складывающуюся в процессе оптимизации ситуацию и поэтому, как показал опыт, неэффективны при решении практических задач.

Очевидно, что в процессе поиска необходимо и увеличивать, и уменьшать рабочий шаг. Очень плодотворной оказалась следующая эвристическая процедура: уменьшить шаг а при неудачном случайном шаге и увеличить — при удачном. В основе такой эвристики лежат естественные соображения: неудачный шаг, как правило, свидетельствует о том, что цель где-то близко и искать ее следует более мелкими шагами, а удачный — о том, что до цели далеко и следует увеличить шаг. Конечно, эти соображения не более чем эвристика, но они дали возможность построить очень эффективный алгоритм адаптации рабочего шага случайного поиска [184]:

$$a_{N+1} = \begin{cases} \gamma_1 a_N & \text{при } \Delta Q_N < 0; \\ \gamma_2 a_N & \text{при } \Delta Q_N \geqslant 0, \end{cases} \quad (3.5.1)$$

где в соответствии с указанными соображениями $\gamma_1 > 1$, $\gamma_2 < 1$. Элементарный анализ показывает, что при $\gamma_1 \approx 1$ величина шага будет уменьшаться по мере приближения к экстремуму.

Зададим вопрос: какими должны быть значения γ_1 и γ_2, чтобы как можно, скорее попасть в точку экстремума U^* или хотя бы в ее достаточно малую окрестность? Это зависит от вероятности того, что случайный шаг будет удачен, т. е. от вероятности события $\Delta Q < 0$. В процессе адаптации следует стремиться к тому значению шага, при котором приближение к цели , будет наибольшим. Пусть вероятность удачного случайного шага при такой оптимальной величине равна p^*. Тогда оптимальные значения γ_1 и γ_2 связаны следующим соотношением [212]:

$$\gamma_1^{p^*} \gamma_2^{1-p^*} = 1, \quad (3.5.2)$$

где величина p^* с ростом n стремится к 0,27. Используя это выражение, можно добиться оптимального режима адаптации алго-

ритма, что позволяет значительно ускорить решение задачи оптимизации.

Однако условие (3.5.2) получено теоретически для довольно частного случая объекта оптимизации. В реальных задачах оптимизации это соотношение иное и различно для каждого конкретного объекта. Очевидно, что необходимо определять параметры γ_1 и γ_2 не заранее, а в процессе поиска, т. е. надо их адаптировать. Следовательно, необходимо создать второй контур адаптации, адаптирующий первый. Эта проблема еще не решена, здесь нужны идеи. Ясно одно: без решения этой задачи не может быть эффективной адаптации величины рабочего шага при оптимизации достаточно сложных систем.

3.5.2.2. Адаптация распределения случайного шага

Этот вид адаптации заключается в том, что получаемая на каждом шаге поиска информация об успехе или неуспехе случайного шага используется для такой деформации распределения случайного шага, при которой эффективность процесса поиска возрастает.

Пусть $p(\xi)$ — плотность распределения случайного шага ξ. Основной характеристикой всякого распределения является его математическое ожидание $M\xi$. Изменяя $M\xi$, можно эффективно воздействовать на процесс поиска. Практически это сведется к прибавлению вектора W к ξ^0 — случайному вектору с нулевым математическим ожиданием:

$$\xi = \xi^0 + W, \qquad (3.5.3)$$

откуда получаем $M\xi = W$.

Вектор W, как видно, определяет «снос» случайного блуждания в процессе поиска. Естественно, что этот снос должен быть направлен в сторону цели, т. е. искомого положения экстремума U^*. Вектор W должен отражать предысторию работы поиска и выявлять перспективное направление движения при оптимизации.

Деформация распределения p(ξ) представлена на рис. 3.5.1. Исходное равномерное по всем направлениям распределение случайного вектора (рис. 3.5.1, *а*) взаимодействует по (3.5.3) с вектором памяти W (рис. 3.5.1, *б*), в результате чего полученное распределение $p(\xi)$ имеет явную тенденцию (снос) в сторону W (рис. 3.5.1, *в)*. Как видно, все сводится к разумной организации вектора W и его оперативному изменению.

Здесь помогает довольно очевидная эвристика: направление следует формировать как взвешенную сумму случайных шагов, причем удачные шаги ($\Delta Q<0$) следует брать с положительными

весами, а неудачные — с отрицательными. Предпочтение должно отдаваться более свежей информации. Эта эвристика реализуется, например, такой простой рекуррентной формулой изменения W в процессе поиска:

$$W_N = kW_{N-1} - \delta\Delta U_N \Delta Q_N, \qquad (3.5.4)$$

где $0 \le k \le 1$ — коэффициент забывания, а $\delta > 0$ — коэффициент интенсивности учета новой информации. Это и есть алгоритм адаптации распределения.

Легко показать, что такая организация изменения W отражает требования, предъявляемые к этому вектору. Сначала рассмотрим крайние случаи. При отсутствии новой информации, т. е. при $\Delta U_N = 0$, в силу $k<1$ вектор $W_N \to 0$, т. е. поиск становится равновероятным, что естественно в данных условиях. Если «забыть» всю предысторию ($k=0$), то $W_{N+1} = -\delta\Delta U_N \Delta Q_N$, т. е. снос будет происходить либо в направлении предыдущего удачного шага, либо противоположно неудачному, что также вполне разумно.

В общем случае взаимодействие векторов W_N и ΔU_N будет таким, что при удаче W_{N+1} будет разворачиваться в сторону удачного шага, а при неудаче — в сторону, противоположную неудачному.

Можно показать, что в неизменной ситуации направление вектора W будет стремиться к антиградиенту, т. е. в сторону наиболее интенсивного уменьшения показателя качества. Происходит это по стандартным законам сложения векторных величин (рис. 3.5.2).

Как видно, вектор W интегрально отражает предысторию поиска, и поэтому его часто называют вектором предыстории. Случайный поиск, снабженный адаптацией вероятностных свойств, обладает, как правило, повышенным быстродействием.

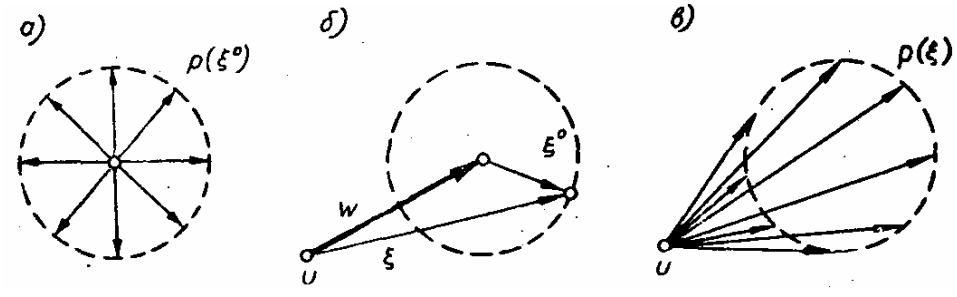


Рис. 3.5.1. Образование плотности распределения случайного шага со сносом.
а — исходное распределение без сноса, *б* — взаимодействие вектора сноса со случайным вектором, в — результирующее распределение со сносом.

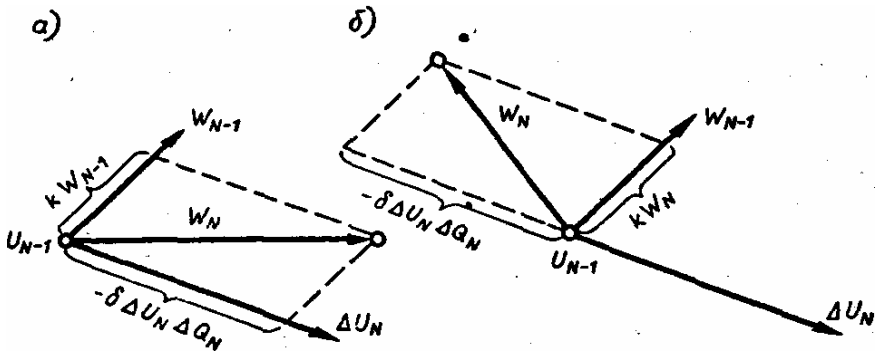


Рис. 3.5.2. Корреляция вектора предыстории при удачном (*а*) и неудачном (*б*) шагах поиска, а — $\Delta Q<0$, *б* — $\Delta Q>0$.

Чтобы такая адаптация была успешной, ситуация, которая складывается в процессе оптимизации, не должна изменяться слиш-. ком быстро, иначе она станет «неуловимой» для адаптации. Поэтому введение адаптации такого рода (часто называемой самообучением) не всегда улучшает процесс поиска, но зато и не ухудшает его.

Здесь следует отметить, что параметры величины рабочего шага (*а*), забывания (*k*) и запоминания (δ) взаимосвязаны и для эффективной адаптации должны определяться в зависимости от специфики ситуации, сложившейся при оптимизации. Так возникает задача адаптации параметров адаптации, т. е. адаптация следующего уровня. Плодотворных подходов к ее решению пока нет, но совершенно ясно, что ее необходимо решать, т. е. нужно построить алгоритм изменения параметров *k* и δ, *c* тем чтобы процесс оптимизации протекал наилучшим образом.

Более полное изложение методов параметрической адаптации случайного поиска имеется в монографии [135].

3.5.3. Структурная адаптация алгоритмов поиска

Адаптация структуры алгоритмов поисковой оптимизации в соответствии с общей классификацией (см. рис. 1.5.2) может осуществляться различным образом. Прежде всего, возможен путь *параметризации структуры* алгоритма поиска, т.е. введение параметров, определяющих структуру алгоритма, например, следующим образом.

Пусть

$$\{F_i\} \quad (i = 1, ..., q) \tag{3.5.5}$$

— конечное множество различных алгоритмов поиска. Тогда их линейная комбинация

$$F = \sum_{i=1}^{q} u_i F_i \quad (3.5.6)$$

образует поиск, параметрами которого являются числа u_1, ..., u_q, вариация которых позволяет получать алгоритмы с различной

структурой. Действительно, при $\sum_{i=1}^{q} u_i = 1, \quad u_i \in \{0; 1\}$, получаем

структуру всех алгоритмов $F=F_i$ (i=1, ..., q). Кроме того, снимая ограничение $u \in \{0; 1\}$, можно получить гибридные структуры. *Эволюционная адаптация* алгоритма поиска связана с представлением алгоритма поиска в виде графа. Для этого достаточно воспользоваться автоматной моделью, аналогичной (3.3.28):

$$\langle \{\Delta Q\}, \{W\}, \{\Delta U\}, \mu(\Delta Q, W), \upsilon(\Delta Q, W) \rangle, \quad (3.5.7)$$

где $\{\Delta Q\}$ — конечный алфавит входов автомата поиска (это множество приращений показателя качества объекта оптимизации, вызванных воздействием ΔU автомата поиска); $\{W\}$ — конечный алфавит внутренних состояний автомата; $\{\Delta U\}$ — конечный алфавит выходов автомата (это множество изменений входа объекта U); μ — функция перехода:

$$W' = \mu(\Delta Q, W), \quad (3.5.8)$$

определяющая новое состояние W' автомата, в которое он переходит из состояния W при входе ΔQ; υ — функция выхода:

$$\Delta U = \upsilon(\Delta Q, W), \quad (3.5.9)$$

определяющая выход автомата в зависимости от его состояния W и входа ΔQ.

Граф этого автомата определяется следующим образом. Множество вершин графа образуется состояниями $\{W\}$ автомата, а его ребра определяются функцией перехода μ.

Эволюционная адаптация такого алгоритма сводится к эволюции графа, которая будет рассмотрена в § 6.1. Здесь функция выхода υ адаптируется параметрическим образом, где параметрами являются значения шагов поиска $\{\Delta U\}$.

И, наконец, процесс структурной адаптации алгоритмов поиска может быть решен средствами альтернативного выбора, т. е. путем *альтернативней адаптации,* которая рассмотрена специально в пятой главе. Там же (§ 5.3) будет исследовано применение методов альтернативной адаптации для адаптации алго-

ритмов поиска, т. е. для адаптивного переключения с одного алгоритма поиска на другой.

Как отмечено выше, процесс структурной адаптации должен сопровождаться подстройкой параметров алгоритма, т. е. параметрической адаптацией. Однако, так как не могут работать одновременно два алгоритма, параметрически адаптируется лишь один — работающий. Поэтому в новой ситуации новый включающийся алгоритм оказывается неадаптированным и, следовательно, малоэффективным. Здесь возникает важная проблема: как адаптировать «безработные» алгоритмы по наблюдениям за одним работающим? В этом направлении, от развития которого почти полностью зависит эффективность структурной адаптации, сделано еще очень мало.

В заключение отметим, что проблема адаптации алгоритмов поисковой оптимизации является сейчас наиболее острой в оптимизации. Разработано много методов поиска, а их адаптация к конкретному объекту оптимизации пока находится на полукустарном уровне.

§ 3.6. Глобальный поиск

Поиск глобального экстремума минимизируемой функции, имеющей несколько локальных экстремумов, является одной из труднейших задач оптимизации. Дело здесь в том, что в процессе глобального поиска должны решаться одновременно две противоречивые задачи: нужно искать каждый конкретный минимум и одновременно уклоняться от него, чтобы найти глобальный минимум. Эта двойственность глобального поиска отражается и на затратах, которые значительно выше затрат на поиск локального экстремума.

Другой специфической чертой глобального поиска является отсутствие полной уверенности, что найденный за конечное время экстремум является глобальным. Действительно, ввиду того что глобальный экстремум может, вообще говоря, оказаться в любой точке области 5 поиска, всегда существует риск утери этого экстремума в процессе поиска. И лишь при неограниченном увеличении времени поиска вероятность такой утери будет сколь угодно малой.

Это неизбежное обстоятельство породило несколько пессимистическое отношение к проблеме глобальной оптимизации: появилось стремление вообще не обращаться к глобальному поиску, а сформулировать задачу так, чтобы она имела лишь один экстремум. Анализ некоторых глобальных задач показывает, что их можно преобразовать к локальным. Однако не всегда это возможно, в силу чего проблема глобального поиска с каждым

годом становится все острее. Число возникающих многоэкстремальных задач неуклонно растет, особенно в области оптимального проектирования [21, 156, 209].

Дело здесь все в том же дефиците времени, который не позволяет сделать тщательный анализ возникшей оптимизационной задачи и заставляет свалить ее решение на «железные» плечи ЭВМ, работающей по алгоритму глобального поиска.

Если относительно локального поиска еще высказывается порой мнение, что он должен быть детерминированным, то относительно глобального поиска, пожалуй, все единодушны: он должен быть стохастическим. Рассмотрим основные алгоритмы глобального случайного поиска.

3.6.1. «Набросовые» алгоритмы

Так обычно называют алгоритмы глобального поиска, в которых используется процедура случайного наброса, т. е. случайного распределения пробных состояний объекта.

3.6.1.1. Случайный наброс с локальным поиском

На каждом i-м этапе из случайной начальной точки U_i делается локальный спуск в ближайший минимум U_i^* любым локальным методом поиска. За глобальный минимум U^{**} принимается тот, для которого показатель качества минимален, т. е.

$$U^{**}_N = \arg \min_{i=1,\ldots,N} Q(U^*_i). \quad (3.6.1)$$

Очевидно, что при $N \to \infty$ вероятность .того, что U^{**}_N является положением глобального минимума, стремится к единице, т. е.

$$\lim_{N\to\infty} \text{Вер}\ (U^{**}_N = U^{**}) = 1. \quad (3.6.2)$$

При конечном N вероятность утери глобального экстремума, т. е. $U^{**}_N \neq U^{**}$, не равна нулю.

Однако использование локального поиска совершенно необязательно при работе набросовых алгоритмов. Это скорее дань локальному поиску. Рассмотрим другие алгоритмы.

3.6.1.2. Адаптивный набросовый алгоритм

Этот алгоритм связан с адаптивным изменением плотности распределения наброса. Пусть $p\ (U \mid W, \sigma)$ — плотность распреде-

ления, параметры которого определяются вектором W, равным, математическому ожиданию случайного вектора U:

$$W = \int U_p (U \mid W, \sigma)\, dU, \qquad (3.6.3)$$

и σ — некоторой скалярной мерой рассеяния этого распределения (типа среднеквадратичного отклонения), такой, что при σ = 0 распределение вырождается в δ-функцию и имеем $U = W$, а с увеличением а область наброса расширяется пропорциональна а по всем направлениям пространства $\{U\}$. Алгоритм поиска заключается в генерировании последовательности случайных точек; $U1$, ..., U_N, ... и выборе точки с наименьшим значением показателя качества аналогично (3.6.1):

$$U^{**}_N = \arg \min_{i=1,\ldots,N} Q(U_i). \qquad (3.6.4)$$

При этом параметры W и а распределения адаптируются, например, следующим образом:

$$\begin{cases} W_{N-1} & \text{при } Q(U_N) \geqslant Q(U^{**}_{N-1}); \\ U_N & \text{при } Q(U_N) < Q(U^{**}_{N-1}), \end{cases} \qquad (3.6.5)$$

т. е. W_N является лучшей из всех предыдущих точек, и

$$\sigma_N = \begin{cases} \gamma_1 \sigma_{N-1} & \text{при } Q(U_N) \geqslant Q(U^{**}_{N-1}); \\ \gamma_2 \sigma_{N-1} & \text{при } Q(U_N) < Q(U^{**}_{N-1}), \end{cases} \qquad (3.6.6)$$

где параметры γ_1 и γ_2 могут выбираться исходя из различных соображений.

В случае $\gamma_1<1$ и $\gamma_2>1$, т. е. при расширении зоны поиска с каждой удачей и сужении — с неудачей, получаем локализирующийся алгоритм, который стремится «стянуть» наброс вокруг лучшей точки. Темп такого стягивания определяет степень глобальности алгоритма. Если стягивание происходит быстро, то, очевидно, будет найден ближайший локальный экстремум; если же медленно, то шансы найти экстремум лучше ближайшего, локального повышаются.

В случае $\gamma \approx 1$ вероятность отыскания глобального экстремума, при $N \to \infty$ стремится к единице (при этом еще необходимо, чтобы p ($U \mid W$, σ) $\neq 0$ для любой точки, т. е. чтобы плотность вероятности-появления любой допустимой точки не была равна нулю).

Возможен и иной подход к выбору параметров γ_1 и γ_2. Если логику адаптации (3.6.6) как бы «вывернуть наизнанку», т. е. положить $\gamma_1>1$, а $\gamma_2<1$, то получим несходящуюся процедуру, которая расширяет зону поиска при неулучшении и сужает ее

при отыскании лучших точек. Оба подхода обладают своими достоинствами и недостатками. Применение их связано с учетом конкретных свойств объекта.

3.6.1.3. Набросовый алгоритм глобального поиска с идентификацией распределений

Этот интересный алгоритм связан с восстановлением (идентификацией) плотности распределения $p(Q)$ значений минимизируемой функции в случайных точках U_i, равномерно распределенных в области поиска S [135]. Если область поиска произвольно разделить на две части и в каждой определить плотность $p_1(Q)$ и $p_2(Q)$, то наиболее перспективной для изучения будет та область, для которой левый «хвост» распределения (напомним, что рассматривается случай минимизации) простирается дальше. Для широкого класса сложных объектов в качестве такого распределения можно воспользоваться логнормальным распределением, левый «хвост» которого ограничен. Наименьшая из оценок этих границ и решает вопрос о выборе наиболее перспективной зоны для дальнейшего аналогичного исследования. Таким образом, рассекая область поиска S последовательно на измельчающиеся зоны, можно выйти в район глобального экстремума и определить его любым локальным методом.

3.6.2. «Блуждающие» алгоритмы

Многие интересные и эффективные алгоритмы глобального лоиска используют процедуру случайного блуждания. Вот некоторые из них.

3.6.2.1. Метод «зашумления» градиента

Если на оценку градиента ∇Q в градиентном поиске наложить вектор ξ, то получаем случайный поиск вида

$$U_{N+1} = U_N - a\nabla Q(U_N) + \sigma\xi, \tag{3.6.7}$$

где σ — некоторая постоянная. Запишем этот процесс в непрерывной форме:

$$\frac{dU}{dt} = -a\nabla Q(U_N) + \sigma\xi(t), \tag{3.6.8}$$

где $\xi(t)$ — случайный векторный процесс типа белого шума с единичными дисперсиями. При $\sigma = 0$ поиск является градиент-

ным, и поэтому каждая его траектория приводит в ближайший локальный минимум, т. е. поиск имеет локальный характер. При $\sigma \neq 0$ на эту локальную тенденцию накладывается случайное блуждание типа броуновского. В результате образуется случайный процесс [232], распределение вероятностей которого имеет вид

$$p(U) = \frac{1}{c} \exp\left[-\frac{2aQ(U)}{\sigma^2}\right], \tag{3.6.9}$$

где c — нормирующий коэффициент. Из этого выражения хорошо видно, что плотность вероятности пребывания функции $Q(U)$ в глобальном минимуме максимальна и чем меньше Q, тем больше эта плотность.

Процесс такого поиска происходит следующим образом. Во время блуждания точка U попадает в локальные экстремумы функции $Q(U)$ и задерживается в каждом из них тем дольше, чем меньше значение $Q(U)$ в соответствующем экстремуме. Если теперь медленно уменьшать a, то в соответствии с (3.6.9) точка U попадает в зону глобального экстремума и уже никогда из нее не выйдет.

Как видно, этот алгоритм гарантирует отыскание глобального экстремума лишь при очень медленном уменьшении параметра σ (строго говоря, для этого нужна нулевая скорость убывания σ). Риск не найти глобальный экстремум тем выше, чем больше скорость уменьшения σ.

3.6.2.2. Метод сглаживания

Если минимизируемая функция $Q(U)$ имеет вид, показанный на рис. 3.6.1, *а,* т. е. образована путем наложения на «хорошую» унимодальную функцию (см. пунктир на рисунке) мелких случайных отклонений, то для отыскания глобального экстремума можно воспользоваться идеей сглаживания [103, 152, 238]. Делается это следующим образом.

В районе точки U осредним функцию $Q(U)$ *с* неотрицательным весом $\rho(Y)$, поведение которого показано на рис. 3.6.1, *б*. Получаем

$$\bar{Q}(U) = \int Q(U+Y)\rho(Y)\,dY. \tag{3.6.10}$$

Эта функция уже ближе к «хорошей», так как здесь несколько сглажена колебательная составляющая. Однако определить этот q-кратный интеграл очень трудно. Указанную трудность можно преодолеть, если для его вычисления воспользоваться методом Монте-Карло, рассмотренным в § 3.1. Действительно, весовую

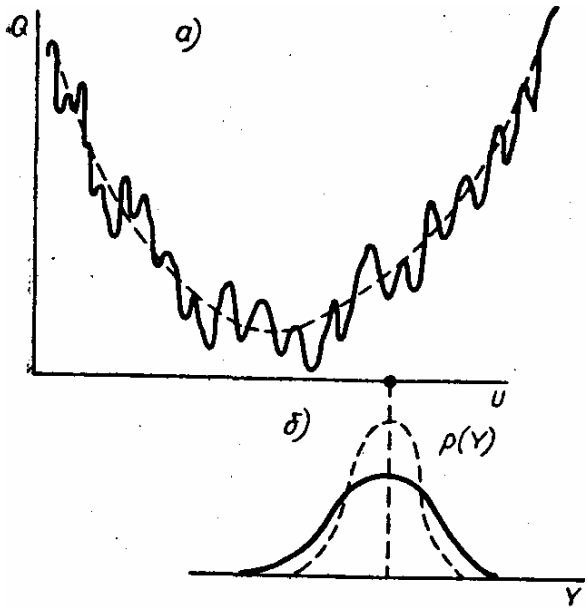


Рис. 3.6.1. Сглаживание многоэкстремальной функции.
а — многоэкстремальная функция, образованная наложением случайных колебаний на унимодальную функцию (пунктирная линия); *б* — весовая функция.

функцию $\rho(\cdot)$ всегда можно выбрать так, чтобы она была многомерной плотностью, т. е. удовлетворяла условиям

$$\int \rho(Y)dY = 1;\ \rho(Y)\geq 0.$$

Тогда, генерируя в соответствии с этой плотностью случайные векторы $Y_1,...,Y_N$, можно оценить значение интересующего нас интеграла следующим образом:

$$\bar{Q}_N(U) = \frac{1}{N}\sum_{i=1}^{N} Q(U+Y_i). \qquad (3.6.11)$$

Выражение (3.6.11) есть монте-карловская оценка, совпадающая с (3.6.10) при $N\to\infty$ и отличающаяся от (3.6.10) при конечном N на случайную величину с нулевым средним и дисперсией порядка $1/N$. Отсюда вытекает возможность подбирать объем N случайной выборки таким образом, чтобы свойства оценки (3.6.11) были приемлемыми.

На оценку (3.6.11) можно воздействовать и путем изменения распределения $\rho(\cdot)$. Так, если дисперсии этого распределения уменьшаются (см. пунктир на рис. 3.6.1, *б*), то дисперсия оценки (3.6.11) также уменьшается, причем, естественно, ухудшается сглаживание. Поэтому выбор $\rho(\cdot)$ и N должен зависеть от свойств минимизируемой функции $Q(U)$ и поставленных целей. Теперь, располагая значением оценки (3.6.11) в любой точке и зная, что функция $\bar{Q}(U)$ унимодальна, можно обратиться к любому локальному методу поиска для решения задачи

$$\bar{Q}(U) \to \min_{U\in S}, \qquad (3.6.12)$$

где значение $\bar{Q}(U)$ определяется путем оценивания по (3.6.11), что, как видно, эквивалентно оптимизации в обстановке случайных помех.

Воспользуемся градиентным методом и для простой оценки градиента $\nabla\bar{Q}(U)$ получим аналитический вид распределения

ρ(•). Для этого преобразуем интеграл (3.6.10) с помощью подстановки $Z = U + Y$ к виду

$$\bar{Q}(U) = \int Q(Z)\rho(Z-U)\,dZ.$$

3.6.13) Градиент функции $Q(U)$ можно определить следующим образом:

$$\nabla\bar{Q}(U) = \int Q(Z)\nabla_U\rho(Z-U)\,dZ, \quad (3.6.14)$$

где операция вычисления градиента $\nabla_U\rho(Z-U)$ осуществляется аналитически. Оценивать (3.6.14) следует также методом Монте-Карло:

$$\hat{\nabla}\bar{Q}(U) = \frac{1}{N}\sum_{i=1}^{N}\frac{Q(Z_i)\nabla_U\rho(Z_i-U)}{p(Z_i)}, \quad (3.6.15)$$

где Z_i — случайные реализации вектора Z, распределенного в соответствии с заданной плотностью $p(Z) \neq 0$ для $Z \in S$.

Как легко заметить, такой поиск является случайным, так как здесь используются случайные шаги Z_i. Эффективная организация поиска требует изменения ρ(•) и N в процессе поиска, т. е. их адаптации таким образом, чтобы вдали от экстремума сглаживание было сильным, а N — малым и чтобы по мере приближения к глобальному экстремуму N возрастало при уменьшении сглаживания.

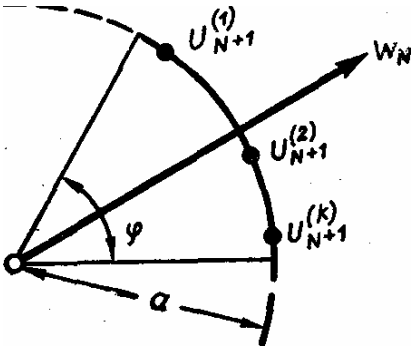
3.6.2.3. Метод направляющего конуса

Очень часто глобальный экстремум находится на «дне» оврага минимизируемой функции $Q(U)$. Под оврагом принято понимать ситуацию, отличающуюся следующей особенностью: почти вдоль всех направлений функция $Q(U)$ сильно увеличивается (это «склоны» оврага), и только в сравнительно узком конусе направлений эта функция слабо уменьшается (это направление «дна» оврага). Если глобальный экстремум лежит на дне такого оврага, то следует воспользоваться методом случайного поиска, называемым поиском с направляющим конусом. Суть его состоит в следующем.

На специфическом дне конуса с вершиной в U_N и осью W_N ($|W_N| = 1$) делаются k случайных проб $U_{N+1}^{(1)}$, ..., $U_{N+1}^{(k)}$ (рис. 3.6.2). Следующее (N+1)-е состояние определяется по наилучшей пробе:

$$U_{N+1} = \arg\min_{i=1,\ldots,k} Q(U_{N+1}^{(i)}), \quad (3.6.16)$$

а ось следующего конуса —



$$W_{N+1}=\frac{1}{a}(U_{N+1}-U_N),$$

(3.6.17)

т. е. вдоль сделанного рабочего шага. Легко видеть, что эта процедура позволяет отслеживать направление оврага независимо от того, вверх или вниз идет овраг.

Рис. 3.6.2. К образованию рабочего шага алгоритмом случайного поиска с направляющим конусом.

Угол между следующими друг за другом шагами не превышает половины угла φ раскрытия направляющего конуса. Поэтому траектория поиска при малом φ имеет очень плавный характер. При φ=2π получаем поиск по наилучшей пробе, рассмотренный в подразделе 3.3.2.3. Варьируя величину угла φ, можно получить адаптивные процедуры овражного глобального поиска [94].

Существует еще много разнообразных процедур глобального случайного поиска. Все они в различной мере используют описанные выше подходы, связанные со случайным набросом и случайным блужданием в пространстве оптимизируемых параметров $\{U\}$.

§ 3.7. Бионические алгоритмы случайного поиска

Интерес, проявленный в свое время к бионике, связан с весьма заманчивыми перспективами использования «патентов природы» в научно-технической практике, т. е. с применением идей биологических механизмов и принципов их функционирования для создания новых алгоритмов работы технических устройств. Скажем прямо: далеко не все надежды, возложенные в свое время на бионику, оправдались и были реализованы в виде действующих алгоритмов и приборов.

Однако самое главное, что внесла бионика в современную науку и технику, — это широкое использование биологических аналогий, а иногда и аллегорий, что тоже вполне оправдано, если дает положительный эффект. Именно в таком бионическом направлении получены наиболее интересные и эффективные алгоритмы случайного поиска. Они моделируют различные уровни биологической организации — от популяционного, связанного со свойствами биологической эволюции, до субклеточного, где тончайшие регуляторные механизмы управления клеткой дают

много интересных идей для синтеза алгоритмов случайного поиска.

Рассмотрим алгоритмы случайного поиска, моделирующие функционирование некоторых уровней биологической организации [270].

3.7.1. Эволюционные алгоритмы

Использование идеи биологической эволюции для синтеза алгоритмов случайного поиска явилось первой бионической идеей, которая была реализована немедленно и в многочисленных модификациях. Вот некоторые из них.

3.7.1.1. Эволюционный алгоритм случайного поиска

Простейший алгоритм глобального поиска моделирует эволюцию следующим образом. Пусть U_1, ..., U_N — произвольные точки в q-мерном пространстве оптимизируемых параметров — моделируют отдельные особи. Степень приспособленности каждой из особей определяется значением минимизируемой функции $Q(U)$. Чем меньше значение этой функции, тем более приспособлена особь и, следовательно, тем больше у нее шансов на выживание.

Пусть имеем N выживших особей. Каждая из них дает потомство, отличающееся от родителя случайным образом. Модель такого потомка имеет вид

$$U_{ji} = U_j + \xi_{ji} \quad (i = 1, ..., k_j), \tag{3.7.1}$$

где k_j — число потомков j-й особи; ξ_{ji} — случайное отклонение, моделирующее отличие i-го потомка от j-й родительской особи в силу случайных мутаций. Это случайный вектор, модуль которого, $a_{ji} = |\xi_{ji}|$, отражающий интенсивность мутаций, обычно не слишком велик, чтобы в соответствии с пословицей «яблочко падало недалеко от яблоньки».

Теперь среди $k = \prod_{j=1}^{N} k_j$ потомков вступает в действие «естественный отбор», при котором с большей вероятностью вымирает потомок, имеющий большее значение минимизируемой функции. Моделируется это следующим образом.

Пусть вероятность «вымирания» пропорциональна значению минимизируемой функции, т. е.

$$p_{ij} = \frac{Q(U_{ij})}{\sum_{i,j} Q(U_{ij})} \qquad (i = 1, \ldots, k_j;\ j = 1, \ldots, N). \tag{3.7.2}$$

Далее разыгрываем процесс вымирания до тех пор, пока не останется N «выживших» потомков. Популяция сформирована и может давать потомство для следующего этапа отбора.

Опыт .показывает, что такая грубая модель эволюции (даже при $k_j = \underset{j}{\text{const}}$) позволяет очень эффективно оптимизировать многоэкстремальные функции при наличии ограничений $U \in S$. В этом случае потомок, нарушивший ограничение, т. е. $U_{ij} \notin S$, «вымирает» сразу независимо от значения $Q(U_{ji})$ [16, 154, 155].

Описанный глобальный поиск, эксплуатирующий эту грубую модель эволюции, привлекателен еще и тем, что может легко усовершенствоваться. Например, можно ввести прижизненную адаптацию потомков, которая легко моделируется смещением U_{ij} в точку ближайшего локального экстремума (для этого можно использовать любой из локальных методов поиска, описанных в § 3.2 и 3.3). Целесообразно также моделировать изменение числа потомков k_j особи и интенсивности случайных мутаций в зависимости от приспособленности особи. Так, хорошие результаты в процессе глобальной оптимизации дает моделирование известного в биологии факта, что в неблагоприятных условиях интенсивность мутаций (a_{ji}) возрастает, а число потомков k_j уменьшается.

Описанный алгоритм глобального случайного поиска моделирует механизм естественного отбора (выживает и дает потомство наиболее приспособленный). Однако для целей искусственного процесса оптимизации удобнее воспользоваться моделированием искусственного отбора, при котором в процессе селекции сохраняются те особи, которые в большей мере обладают необходимыми свойствами.

В практической биологии этот феномен играет огромную роль и во многом определяет механизм эволюции окультуренных видов. Рассмотренный в § 3.3 (подраздел 3.3.3.1) случайный поиск с помощью коллектива независимых автоматов с целесообразным поведением является хорошей основой для моделирования указанных биологических явлений в процессах случайного поиска. Рассмотрим некоторые из них [136].

3.7.1.2. Популяционный алгоритм случайного поиска

Пусть имеется некая популяция автоматов $A_1, ..., A_N$ ($N>q$), каждый из которых имеет свои параметры $q_1, ..., q_l$, характеризующие его (например, глубина памяти, вероятности условных переходов и т. д.). Эффективность каждого i-го автомата определяется его анкетой, или лицевым счетом:

$$Si = (s_1, ..., s_{qi}) \quad (i = l, ..., N), \tag{3.7.3}$$

где компоненты анкеты определяют прошлую работу автомата на каждом канале оптимизируемого объекта следующим образом:

$$s_{ji}=\begin{cases} >0, \text{ если работа на } j\text{-м канале была} \\ \quad \text{успешной, т. е. } \Delta Q<0; \\ =0, \text{ если опыта не было;} \\ <0, \text{ если работа на } j\text{-м канале была} \\ \quad \text{неуспешной, т. е. } \Delta Q\geqslant 0. \end{cases}$$

В процессе оптимизации выбор автомата для работы на определенном j-м канале зависит от величин $s_{ji}, \ldots, s_{jN}$ и с большей вероятностью выбирается тот автомат, у которого величина s_{ji} больше.

Теперь введем скрещивание. Для этого надо оценить эффективность автомата. Она определяется суммой

$$\alpha_i=\sum_{j=1}^{q} s_{ji} \qquad (i=1, \ldots, N). \quad (3.7.4)$$

Легко заметить, что чем больше эта величина, тем лучше работает автомат. Поэтому величина α_i определяет приоритет i-го автомата в выборе пары. Первым, естественно, выбирает пару автомат с максимальной эффективностью α, причем с большей вероятностью он выбирает себе в пару автомат с большим α (как видно, этот алгоритм моделирует механизм образования брачных пар в животном мире). Далее по тому же принципу выбирает себе пару автомат с меньшим значением эффективности и т. д. Естественно вводить и «холостые» автоматы, которым с определенной вероятностью не удается найти себе пары. Эта вероятность должна увеличиваться с уменьшением α, моделируя «неприспособленность» автоматов с малым α.

После образования пар происходит скрещивание, т. е. образование автомата-потомка. В соответствии с законами наследственности он наследует параметры своих родителей так, что одни параметры он берет с определенной вероятностью у одного родителя, а другие — у второго. Вводятся и случайные мутации в виде случайных изменений этих параметров. Новый автомат получает и свою анкету, которая образуется путем осреднения анкет родителей (это моделирует применение известного правила селекционеров оценивать молодые особи по их родителям), и сразу включается в процесс оптимизации.

Рождение новых автоматов сопровождается «гибелью» других, моделирующей селекцию при искусственном отборе.

Вероятность сохранения автомата при селекции пропорциональна его эффективности a, что естественно при реализации искусственного отбора.

Эксперименты с алгоритмами случайного поиска [136], моделирующими описанный механизм искусственного отбора, показали их большую гибкость, надежность и эффективность.

3.7.2. Поведенческие алгоритмы

Многие алгоритмы случайного поиска моделируют поведение живых организмов в неизвестной среде. В данном случае моделируется присущая живым организмам способность стремиться к комфортным ситуациям и уклоняться от некомфортных. Эта способность, выработанная эволюцией, глубоко рациональна и широко используется в процессах управления. Используется она и в случайном поиске. Назовем прежде всего алгоритм с линейной тактикой, рассмотренный в § 3.3 (3.3.12). Этот алгоритм моделирует поведение живого существа в среде, состояния которой могут быть для него комфортными и некомфортными. Попадая в такого рода среду, животное и человек обучаются: у них появляется стремление к комфортным ситуациям, в чем и состоит линейность их поведения. (Линейность поведения широко используют при дрессировке животных и при... воспитании. Поощрение в процессах воспитания дает результат именно благодаря этому свойству человеческого поведения.)

Рассмотрим модель поведения мыши в Т-образном лабиринте (рис. 3.7.1). Подобные эксперименты неоднократно проводились в различных лабораториях. Мышь здесь должна выбрать одно из двух направлений. Пусть p_i — вероятность ее поворота влево в i-м эксперименте. Легко представить результаты такого рода

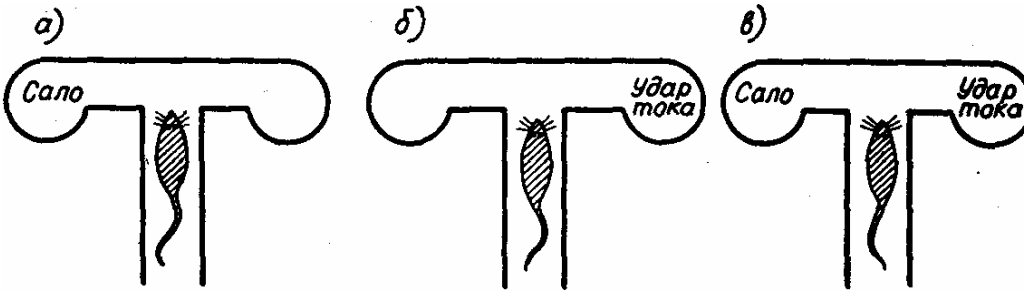


Рис 3.7.1. Варианты экспериментов по выработке определенного поведения:
а — с поощрением, *б* — с наказанием, *в* — сочетание поощрения с наказанием.

экспериментов. В первом эксперименте (рис. 3.7.1, *а*) зверек обучится устойчиво поворачивать влево (p=1) в результате поощрения, получаемого при нужной реакции (кусочек сала). Динамика такого обучения выражается следующей рекуррентной формулой [39]:

$$p_{i+1} = (1 - \alpha)\, p_i + \alpha. \qquad (3.7.5)$$

Вероятность поворотов влево на каждом шаге будет увеличиваться на $\alpha(1 — p_i)$, т. е. $p_{i+1} = p_i + \alpha\,(1 — p_i)$, где $0<\alpha<1$ — параметр обучения при поощрении. Чем больше а, тем быстрее обучается зверек. Если же изменить положение приманки, то процесс обучения будет описываться несколько иначе:

$$p_{i+1} = (1 - \alpha)\, p_i, \qquad (3.7.6)$$

т. е. вероятность будет уменьшаться, так как $1-\alpha < 1$.

Совершенно иначе построен второй эксперимент (рис. 3.7.1, *б*), в котором для обучения используется наказание (удар тока). Мышь и здесь обучится поворачивать влево, причем процесс ее обучения можно представить аналогичной формулой $p_{i+1} = = (1 — \beta)\, p_i + \beta$, где $0<p<1$ — также параметр обучения, который, вообще говоря, не совпадает с α.

И, наконец, тот же результат будет получен при обучении по «принципу кнута и пряника» (рис. 3.7.1, *в*), *но с* иным параметром обучения γ.

Как видно, одного и того же поведения — поворота влево — можно добиться от животного разными способами — поощрением, наказанием и их, сочетанием.

Все эти биологические модели могут быть использованы при организации случайного поиска следующим образом. Пусть p_{ji} — вероятность положительного смещения вдоль j-й координаты на i-м шаге поиска:

$$p_{ji} = \text{Вер}\,(\Delta u_{ji} > 0) \qquad (3.7.7)$$

(пусть величина смещения будет равна единице, т. е. $|\,\Delta u_j\,| = 1$). Естественно эту вероятность изменять так, чтобы увеличить число удачных шагов, для которых $\Delta Q<0$.

Здесь можно применить все три рассмотренные выше схемы обучения:

1. Обучение только поощрением (только при $\Delta Q<0$):

$$p_{j,i+1} = (1 - \alpha)\, p_{ji} + \alpha\ \text{sg}\ u_{ji} \quad (j = 1, ..., q), \qquad (3.7.8)$$

где

$$\operatorname{sg} z = \begin{cases} 1 \text{ при } z > 0; \\ 0 \text{ при } z \leqslant 0. \end{cases}$$

Хорошо видно, что эта вероятность увеличивается, если шаг $\Delta u_{ji}>0$ оказался удачным, и уменьшается в обратном случае, что вполне логично. В случае неудачи ($\Delta Q \geq 0$) $p_{j,i+1}=p_{ji}$ (j = 1, ..., q), т. е. вероятности, естественно, не изменяются. При оптимизации алгоритмом поиска

$$\Delta u_{ji} = \begin{cases} 1 \text{ с вероятностью } p_{ji}; \\ -1 \text{ с вероятностью } 1-p_{ji} \end{cases} \quad (3.7.9)$$

процесс поиска будет иметь выраженный глобальный характер. Его будет «нести» в направлении первого же удачного шага, пока не будет сделан другой удачный, изменяющий направление «сноса», и т. д.

2. Обучение только наказанием (при $\Delta Q \geq 0$):

$$p_{j,i+1} = (1 - \beta)\, p_{ji} + \beta \operatorname{sg}(-\Delta u_{ji}) \quad (j = 1, ..., q). \qquad (3.7.10)$$

При удаче ($\Delta Q<0$) $p_{j,i+1} = p_{j,i}$ (j = 1, ..., q), т. е. вероятности не изменяются.

Легко видеть, что в этом случае поиск имеет локальный характер, что приводит к устойчивым блужданиям в районе ближайшего локального экстремума.

3. Обучение поощрением и наказанием можно построить дво яко. Величину поощрения и наказания можно сделать одинаковой, что приводит к следующей схеме обучения:

$$p_{j,i+1} = (1- \gamma)\, p_{ji} + \gamma \operatorname{sg}(-\Delta u_{ji}\, \Delta Q) \quad (j = 1, ..., q). \qquad (3.7.11)$$

Однако более гибка и интересна другая схема, объединяющая обучение по формуле (3.7.8) при $\Delta Q<0$ и по формуле (3.7.9) при $\Delta Q \geq 0$. Уровень поощрения и наказания может быть различным (при $\alpha=\beta=\gamma$ эта схема вырождается в (3.7.11)). Здесь, варьируя величины α и β можно придать различный характер поведению алгоритма случайного поиска (3.7.9), моделируя при этом различные способы обучения (точнее — дрессировки).

Любопытно, что при $\alpha>\beta$, т. е. при более интенсивном поощрении, поиск напоминает поведение любознательного зверька: он ищет в более широкой области, чем при $\beta<\alpha$, когда поиск имеет локальный характер. Сильное наказание сужает область поиска зоной ближайшего локального экстремума.

Как видно, анализ и моделирование поведения живых существ в процессе их обучения дают возможность построить новые эффективные алгоритмы случайного поиска.

3.7.3. Клеточные и субклеточные алгоритмы

Спонтанная активность сети нейронов и стохастическое поведение некоторых внутриклеточных структур нейрона послужили основой для создания новых алгоритмов случайного поиска, моделирующих эти известные биологические феномены.

В такой сети в процессе поиска изменяются матрица параметров связи нейронов и их пороги чувствительности. При этом считается, что минимизируемая целевая функция $Q(U)$ определяется управляющим нервным центром и ее значение по эфферентному каналу сообщается каждому нейрону в сети.

Анализ этих феноменов дал возможность создать новый алгоритм случайного поиска, названный матричным [72, 73]. Матричный алгоритм отличается тем, что случайно возмущается матрица преобразования рабочего шага поиска, за счет чего к реализуется алгоритм случайного поиска.

Траектории спуска полученного таким образом алгоритма матричного случайного поиска имеют криволинейный характер, что создает ему преимущество при оптимизации сложных объектов.

Выше приведено несколько примеров успешного использования некоторых результатов биологических исследований для синтеза алгоритмов случайного поиска. Но в последнее время наметилась и противоположная тенденция: случайный поиск начал оказывать влияние на фундаментальные представления о функционировании различных биологических систем, в частности такой сложной системы, как живая клетка. Так, интенсивно и плодотворно развивается представление о «случайно-поисковом» характере поведения некоторых регуляторных механизмов клетки, минимизирующих энергетические затраты на внутриклеточную пассивную регуляцию. Источником случайности здесь является стохастическое поведение микроструктурных образований («эндоплазматического ретикулума») клетки, которое создает режим случайного поиска ее состояния, соответствующего энергетически наиболее экономичному обеспечению специфической функции клетки [71].

Именно на этих фактах было основано предложение ввести в модель нервной клетки механизм случайного поиска оптимальных значений ее основных параметров — коэффициентов чувствительности специфических (синаптических) входов.

Модель клетки, в которую включен механизм случайного поиска, оптимизирующего эти параметры [69, 71], позволяет минимизировать «дискомфорт» этой клетки в данной среде. Несмотря на крайнюю упрощенность такой модели, проведенные эксперименты продемонстрировали достаточно хорошее ее

соответствие наблюдаемым нейроцитохимическим и нейрофизиологическим результатам поведения живого нейрона, что, несомненно, обусловлено именно включением в нее механизмов случайного поиска.

Таким образом, случайный поиск в данном случае выступает уже в качественно новой форме — как объяснительное начало, оказывая тем самым влияние на фундаментальные представления не только о механизмах внутриклеточных регуляторных процессов, но и о процессах в биосфере [269].

Приведенный в этой главе материал имеет обзорный характер, причем использованы главным образом работы автора. Здесь совершенно не затронуты теоретические аспекты случайного поиска, в основе которых лежит обобщающая сила случайного поиска (см. п. 3.3.1). Автор убежден, что создание общей теории адаптации возможно именно на этом пути. Залогом являются прежде всего бионические предпосылки: ведь природа «работает» методом случайного поиска, и все естественные тончайшие механизмы адаптации образовались благодаря случайным процессам эволюции, которая, по сути дела, реализовала алгоритмы случайного поиска.

Более того, всеобщность процессов случайного поиска позволяет утверждать, что он является основой многих известных детерминированных закономерностей как живой, так и неживой природы.

Глава 4

Параметрическая адаптация

...Звуки умертвив,
Музыку я разъял, как труп. Поверил Я алгеброй гармонию.

Пушкин. Моцарт и Сальери

В тех случаях, когда объект удается «разъять» до параметров, т. е. параметризировать, адаптацию ввести наиболее просто. В этой области имеются хорошая математическая теория и множество практических приложений. Однако опыт использования параметрических алгоритмов требует введения адаптации в сам процесс адаптации. Такая адаптация второго уровня позволяет поддерживать свойства адаптивного процесса на заданном уровне.

Данная глава посвящена решению прикладных задач параметрической адаптации объектов различной природы. Здесь рассматриваются процессы обучения, распознавания и идентификации, синтез логических элементов, планов эксперимента и датчиков случайных чисел с заданными свойствами. Указанные задачи решаются с использованием стохастических алгоритмов адаптации, или методов случайного поиска. Полученные результаты показывают эффективность этих алгоритмов.

§ 4.1. Некоторые алгоритмы параметрической адаптации

Задача параметрической адаптации сводится к оптимизации функции $Q(U)$ в обстановке случайных помех, т. е. по ее наблюдениям, зашумленным аддитивной помехой:

$$Q'(U) = Q(U) + \varepsilon(\sigma), \qquad (4.1.1)$$

где U — вектор оптимизируемых параметров, а $\varepsilon(\sigma)$ — независимые случайные помехи с нулевым математическим ожиданием и дисперсией σ^2, которая может быть неизвестна. Для решения задачи оптимизации функции

$$Q(U) \to \min_{U \in S}, \qquad (4.1.2)$$

заданной лишь своими наблюдениями (4.1.1), разработаны многочисленные адаптивные методы, некоторые из них рассмотрены

в предыдущей главе (см., например, § 3.3, 3.5—3.7). Рассмотрим теперь алгоритмы, специально ориентированные на решение задачи (4.1.2).

4.1.1. Метод стохастической аппроксимации

Этот метод является по сути дела градиентным методом с программно изменяемыми пробными и рабочими шагами:

$$\Delta U_{N+1} = -a_N \hat{\nabla} Q(U_N),$$

4.1.3) где компоненты оценки градиента $\hat{\nabla} Q(U_N)$ имеют вид

$$\frac{\partial \hat{Q}}{\partial u_i} = \frac{1}{2g_N}[Q'(U_N + g_N e_i) - Q'(U_N - g_N e_i)] \quad (i = 1, \ldots, q).$$

(4.1.4)

Здесь e_i — i-й орт (i = 1, ..., q). Для сходимости этой процедуры при достаточно широком классе функций $Q(U)$ необходимо, чтобы коэффициенты рабочего (a_N) и пробного (g_N) шагов удовлетворяли следующим соотношениям [41, 237]:

$$\sum_{i=1}^{\infty} a_N = \infty; \quad \sum_{i=1}^{\infty} \left(\frac{a_N}{g_N}\right)^2 < \infty; \quad \lim_{N\to\infty} a_N = \lim_{N\to\infty} g_N = 0. \quad (4.1.5)$$

Первое из этих соотношений требует, чтобы величина a_N уменьшалась не слишком быстро, а второе — чтобы пробный шаг уменьшался значительно медленнее рабочего. Условиям (4.1.5) удовлетворяют, например, следующие соотношения:

$$a_N = \frac{c}{N^{\alpha}}; \quad g_N = \frac{c}{N^{\beta}}, \quad (4.1.6)$$

где $0<\alpha\leq 1$; $\beta>0$; $2(\alpha-\beta)>1$.

Хотя сходимость этого метода доказана строго математически, его практическое применение при решении прикладных задач обычно не удовлетворяет пользователя. Дело здесь в том, что указанная сходимость метода проявляется при $N\to\infty$, что, естественно, не играет роли при практических расчетах. На практике необходимо в процессе адаптации достаточно быстро попасть в некоторую малую окрестность экстремума заданного критерия. С другой стороны, программный характер изменения параметров (4.1.6) не учитывает ситуации, сложившейся в процессе адаптации, что не может не повлиять на эффективность процесса поиска. Наконец, условия (4.1.5) не

учитывают дрейф

экстремума, которым отличается всякий реальный объект адаптации.

Указанными обстоятельствами и объясняется, что стохастическая аппроксимация не используется для решения практических задач адаптации, хотя и хорошо исследована математически. Потребности практики заставляют искать иные, более эффективные пути решения задачи параметрической адаптации. Рассмотрим некоторые из них.

4.1.2. Сглаживание помех

Идея сглаживания [132, 102], примененная в § 3.6 к решению многоэкстремальных задач, может быть использована и для сглаживания случайных помех. Действительно, процедура сглаживания

$$\bar{Q}(U) = \int Q'(U+Y)\rho(Y)\,dY \quad (4.1.7)$$

позволяет избавиться не только от локальных экстремумов, но и от случайных помех.

Для исследования сглаживания помех представим сглаженную функцию (4.1.7), как и в § 3.6, в виде суммы, полученной методом Монте-Карло:

$$\bar{Q}_N(U) = \frac{1}{N}\sum_{i=1}^{N} Q'(U+Y_i), \quad (4.1.8)$$

где Y_i — случайные реализации вектора Y в соответствии с его заданной плотностью распределения $\rho(Y)$.

Подставив в сумму (4.1.8) выражение (4.1.1), получим:

$$\bar{Q}_N(U) = \frac{1}{N}\sum_{i=1}^{N} Q(U+Y_i) + \frac{1}{N}\sum_{i=1}^{N} \varepsilon_i(\sigma). \quad (4.1.9)$$

Легко видеть, что случайная составляющая сглаженной функции (4.1.9) имеет пониженную дисперсию:

$$D\left(\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i(\sigma)\right) = \frac{\sigma^2}{N}. \quad (4.1.10)$$

Пусть распределение $p(\cdot)$ конечно и его границы достаточно близки. В этом случае $Q(U+Y_i)$ можно представить в виде линейной формы по Y:

$$Q(U+Y) = Q(U) + [\nabla Q(U), Y], \quad (4.1.11)$$

где квадратными скобками обозначено скалярное произведение. Тогда

$$\bar{Q}_N(U) = Q(U) + \delta Q_N, \quad (4.1.12)$$

где

$$\delta Q_N = \left[\nabla Q(U), \frac{1}{N} \sum_{i=1}^{N} Y_i \right] + \varepsilon \left(\frac{\sigma}{\sqrt{N}} \right). \quad (4.1.13)$$

Отсюда вытекает, что при центрированном распределении $p(\bullet)$, т. е. при $M(Y) = 0$,

$$\lim_{N \to \infty} \delta Q_N = 0. \quad (4.1.14)$$

Это означает, что оценка (4.1.8) состоятельна (разумеется, в случае справедливости линейного представления (4.1.11)). Указанное обстоятельство обеспечивает фильтруемость случайной помехи при использовании оператора сглаживания (4.1.7).

Аналитическая форма задания плотности распределения $p(\bullet)$ дает возможность [166] оценить градиент «зашумленной» функции (4.1.1). Для этого достаточно продифференцировать (4.1.7) по U. Представим (4.1.7) в виде

$$\bar{Q}(U) = \int Q'(Z) \rho(Z - U)\, dZ, \quad (4.1.15)$$

где $Z = U + Y$. Теперь операция вычисления градиента сглаженной функции осуществляется очень просто:

$$\nabla_U \bar{Q}(U) = \int Q'(Z) \nabla_U \rho(Z - U)\, dZ,$$

4.1.16) где ∇_U — градиент по вектору U:

$$\nabla_U = \left(\frac{\partial}{\partial u_1}, \ldots, \frac{\partial}{\partial u_q} \right). \quad (4.1.17)$$

Производные

$$\varphi_i(Z, U) = \frac{\partial}{\partial u_i} \rho(Z - U) \quad (i = 1, \ldots, q), \quad (4.1.18)$$

образующие компоненты вектора градиента, вычисляются аналитически. При вычислении интеграла (4.1.16) естественно также воспользоваться методом Монте-Карло. Для этого запишем (4.1.16) в виде

$$\nabla_U \bar{Q}(U) = \int Q'(U + Y) \frac{\Phi(U + Y, U)}{p(Y)} p(Y)\, dY. \quad (4.1.19)$$

Здесь $p(Y)$ — некоторая заданная плотность распределения, не равная нулю в области интегрирования; Ф — вектор с компонентами (4.1.18):

$$\Phi(\bullet,\bullet) = (\varphi_1(\bullet,\bullet), \ldots, \varphi_q(\bullet,\bullet)). \tag{4.1.20}$$

Монте-карловская оценка градиента (4.1.16) имеет вид

$$\hat{\nabla}\bar{Q}(U) = \frac{1}{N}\sum_{i=1}^{N} Q'(U+Y_i)\,\frac{\Phi(U+Y_i, U)}{p(Y_i)}, \tag{4.1.21}$$

где Y_i — случайные реализации вектора, имеющего плотность распределения $p(Y)$. Располагая оценкой (4.1.21), легко построить градиентную процедуру адаптации

$$U_{j+1} = U_j - a_j\hat{\nabla}\bar{Q}(U_j). \tag{4.1.22}$$

Здесь оценка градиента (4.1.21) имеет стохастический характер, что и относит процедуру (4.1.22) к алгоритмам случайного поиска с параметром N. Величина этого параметра позволяет управлять уровнем помех, накладывающихся на градиент минимизируемой функции.

Однако оценку градиента можно получить и иным образом.

4.1.3. Стохастическое накопление

Воспользуемся для вычисления градиента его стохастической оценкой (3.3.28), рассмотренной в §3.3. Эту оценку можно определить как оценку с центральной пробой:

$$\hat{\nabla}Q(U) = \frac{q}{mg}\sum_{i=1}^{m}\xi_i[Q'(U+g\xi_i) - Q'(U)]. \tag{4.1.23}$$

Естественно рассмотреть и стохастическую оценку градиента с парными пробами:

$$\hat{\nabla}Q(U) = \frac{q}{2mg}\sum_{i=1}^{m}\xi_i[Q'(U+g\xi_i) - Q'(U-g\xi_i)]. \tag{4.1.24}$$

Здесь ξ_i — i-я реализация единичного случайного вектора, равномерно распределенного по всем направлениям пространства адаптируемых параметров $\{U\}$.

Рассмотрим оценку (4.1.24). Пусть параметр пробного шага g мал, а функция качества достаточно гладка, чтобы линейное

представление (4.1.11) было адекватным. Тогда, подставляя (4.1.11) в (4.1.24), получаем с учетом (4.1.1):

$$\hat{\nabla}Q(U)=\frac{q}{m}\sum_{i=1}^{m}[\nabla Q(U),\xi_i]\xi_i+\frac{q}{2mg}\sum_{i=1}^{m}\varepsilon_i(\sqrt{2}\sigma)\xi_i. \quad (4.1.25)$$

Используем стохастические свойства этой оценки. Начнем со свойств суммы

$$L_i=[\nabla Q(U),\xi_i]\xi_i+\frac{1}{2g}\varepsilon_i(\sqrt{2}\sigma)\xi_i, \quad (4.1.26)$$

которая образует оценку (4.1.25):

$$\hat{\nabla}Q(U)=\frac{q}{m}\sum_{i=1}^{m}L_i. \quad (4.1.27)$$

Рассмотрим проекцию L_i на градиентное направление:

$$l_i=np_{\nabla^0}L_i=k[\nabla^0,\xi_i]^2+\frac{1}{2g}\varepsilon_i[\nabla^0,\xi_i], \quad (4.1.28)$$

где ∇^0 — направление градиента:

$$\nabla^0=\frac{\nabla Q(U)}{|\nabla Q(U)|}, \quad (4.1.29)$$

а k — его модуль:

$$k=|\nabla Q(U)|, \quad (4.1.30)$$

т. е.

$$\nabla Q(U)=k\nabla^0. \quad (4.1.31)$$

Введем φ_i — угол между векторами ∇^0 и ξ_i. Тогда (4.1.28) записывается в виде

$$l_i=k\cos^2\varphi_i+\frac{1}{2g}\varepsilon_i(\sqrt{2}\sigma)\cos\varphi_i. \quad (4.1.32)$$

Известно [161], что распределение случайного угла φ имеет вид

$$p(\varphi)=\frac{\Gamma\left(\frac{q}{2}\right)}{\sqrt{\pi}\Gamma\left(\frac{q-1}{2}\right)}\sin^{q-2}\varphi, \quad (4.1.33)$$

где Г — гамма-функция.

Определим основные статистические характеристики проекции l_i.

Математическое ожидание:

где

$$B_q = \left(\int_0^{\pi} \sin^{q-2} \varphi d\varphi \right)^{-1} = \frac{\Gamma\left(\frac{q}{2}\right)}{\sqrt{\pi}\,\Gamma\left(\frac{q-1}{2}\right)},$$

(4.1.34)

(4.1.35)

откуда окончательно получаем

$$Ml = \frac{k}{q}. \qquad (4.1.36)$$

Дисперсия

$$Dl = k^2 \left[1 - \frac{2B_q}{B_{q+2}} + \frac{B_q}{B_{q+4}} \right] + \frac{\sigma^2}{2qg^2} = \frac{3k^2}{q(q+2)} + \frac{\sigma^2}{2qg^2}.$$

(4.1.37)

В процессах адаптации реальных объектов важно, чтобы вероятность неудачного шага была меньше заданной, т. е.

$$p = \text{Вер}\,(\Delta Q > 0) \le p^*, \qquad (4.1.38)$$

где p^* — заданная вероятность неудачного шага. Легко заметить, что вероятность (4.1.38)

$$p = \text{Вер}\left(\sum_{i=1}^{m} l_i < 0 \right). \qquad (4.1.39)$$

При достаточно большом m (как. обычно и

бывает при адаптации) сумма $\sum_{i}^{m} l_i$ распределена нормально, что позволяет легко определить вероятность p:

$$p=\frac{1}{2}\left[1-\Phi\left(\frac{mk}{q\sqrt{2Dl}}\right)\right], \quad (4.1.40)$$

где Ф — интеграл Лапласа [42].

Как видно, располагая информацией о значениях k и σ, легко с помощью накопления объемом m реализовать выполнение условия (4.1.38).

Дискретный вариант адаптации, когда каждый адаптируемый параметр варьируется на двух уровнях, рассмотрен в работе [182], а релаксация — в [267].

§ 4.2. Адаптация в процессах обучения

Задача обучения может быть естественным образом сформулирована как задача управления. В этом случае ученик выступает в качестве объекта управления, а учитель или обучающее устройство — в качестве источника управления. Очевидно, что такого рода объект управления является сложным объектом и к его управлению применимы все известные принципы управления сложным объектом [172] (см. § 2.1). Поэтому рассмотрим снова блок-схему управления сложным объектом и интерпретируем ее с позиций проблемы обучения.

4.2.1. Обучение как управление сложным объектом

Блок-схема взаимодействия объекта, среды и управляющего устройства (УУ) в процессе управления показана на рис. 4.2.1. Здесь X — состояние среды, влияющей на состояние Y объекта. Информация об этих состояниях измеряется датчиками D_X и D_Y, которые информируют УУ. Очевидно, что $X' \neq X$ и $Y' \neq Y$ в силу того, что датчики измеряют только то, что используется в процессе управления. Однако $X' \in X$ и $Y' \in Y$, т. е. получаемая информация в какой-то мере (но далеко не полностью) отражает действительное состояние объекта и среды. Заметим, что ресурсы, выделяемые на создание системы управления, в значительной степени определяют и объем получаемой информации X' и Y'.

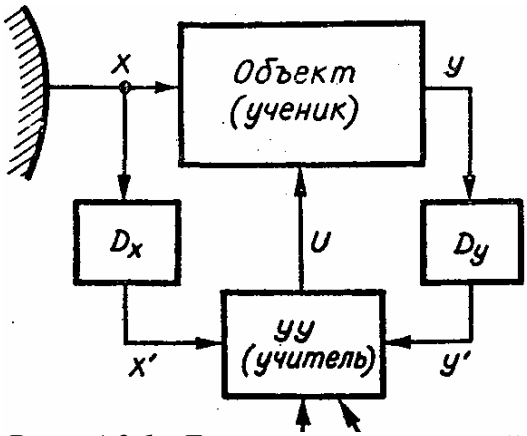


Рис. 4.2,1. Блок-схема взаимодействия ученика и учителя в процессе обучения как управления сложным объектом.

УУ, получая на входе информацию о среде X, объекте Y, цели Z^* и ресурсах R, должно выдать на выходе информацию об управлении U, с помощью которого возможно достичь цели Z^*, т. е. перевести объект в искомое состояние Y^*, соответствующее выполнению заданной цели Z^* в рамках ресурсов R, т. е.

$$\langle X', Y', Z^*, R \rangle \rightarrow U^* \rightarrow Y^*. \quad (4.2.1)$$

Алгоритм управления A призван решить эту задачу:

$$U^* = A\,(X',\ Y',\ Z^*,\ R) \qquad (4.2.2)$$

и является оператором, перерабатывающим исходную информацию в управление.

Задачу синтеза оператора A управления обычно декомпозируют на две.

1. Синтез модели F объекта, связывающей его наблюдаемые входы:

$$Y' = F\,(X',\ U). \qquad (4.2.3)$$

2. Синтез управления с помощью этой модели. В простейшем случае этого можно достичь, решая задачу минимизации, на пример:

$$\|Y^* - F\,(X',\ U)\| \to \min_{U \in R}, \qquad (4.2.4)$$

где множество допустимых управлений определяется выделенными для этого ресурсами R, а искомое состояние Y^* — заданной целью Z^*.

Следовательно, процесс синтеза модели F объекта управления является необходимым элементом управления сложным объектом. Такого рода управление обычно называют управлением с моделью.

Рассмотрим с этих позиций процесс обучения. Изображенная на рис. 4.2.1 система управления может рассматриваться и как система обучения. Здесь объект управления является объектом обучения (назовем его условно «ученик»), а УУ — обучающим устройством (назовем его условно «учитель»). X — состояние среды, влияющей на процесс обучения ученика (учитель информируется о состоянии среды X с помощью датчика D_X); X' — информация о среде X, получаемая учителем. Y — состояние ученика, измеряемое датчиком D_Y, на выходе которого имеется Y' — информация об этом состоянии, получаемая учителем в виде ответов на вопросы U, задаваемые ученику.

Учителю сообщаются цели обучения Z^* и ресурсы R, которыми он располагает для обучения. Задача состоит в том, чтобы организовать обучение U, изменяющее состояние У ученика таким образом, чтобы выполнялись поставленные цели обучения Z^*.

Как видно, в процессе обучения управление U имеет двоякую форму — обучающей информации и вопросов, ответы Y' на которые дают возможность оценить степень обученности ученика.

Используем опыт, накопленный при управлении сложными объектами для организации эффективного обучения. Для этого рассмотрим этапы такого управления и интерпретируем их с позиций обучения [171].

4.2.2. Этапы обучения

В процессе управления сложным объектом таких этапов восемь (см. § 2.1). Рассмотрим каждый из них отдельно, но применительно к процессу обучения.

4.2.2.1. Формулировка целей обучения

Формулировка целей управления заключается в определении критериев и тех требований к ним, выполнение которых решает задачу управления (см. подраздел 1.1.2). Цели Z^* всякого управления представимы в виде

$$Z^*: \begin{cases} \varphi_i \geqslant a_i \quad (i=1, \ldots, k_1); \\ \psi_j = b_j \quad (j=1, \ldots, k_2); \\ \eta_l \to \text{extr} \quad (l=1, \ldots, k_3). \end{cases} \tag{4.2.5}$$

Здесь φ_i, ψ_j и η_l — критерии-функционалы, определяемые на состояниях объекта управления.

Как видно, цели, т. е. требования, предъявляемые к этим функционалам, имеют тройственный характер: цели-неравенства, цели-равенства и экстремальные цели. Они выражают потребности субъекта, взаимодействующего с объектом управления. Выполнение этих целей Z^* удовлетворяет потребности субъекта, ради которых он и создает систему управления.

Теперь интерпретируем сказанное для случая обучения. Здесь субъектом выступает заказчик системы обучения, которому предстоит использовать кадры, подготовленные системой обучения. Цели (4.2.5) обучения в данном случае представляют собой требования к обученному ученику.

Цели-неравенства определяют тот минимум знаний и навыков ученика, нарушение которого недопустимо. Например, функционал φ_i может выражать в виде балльной оценки уровень знаний по i-му предмету. Тогда $a_i = 3$.

Цели-равенства связаны с теми знаниями и навыками ученика, отсутствие которых недопустимо. Например, в школьной системе обучения ψ_j может выражать знание j-го закона природы. Тогда $\psi_j = 1$ определяет знание этого закона, а $\psi_j = 0$ — незнание, что недопустимо, так как $b_j = 1$.

Экстремальные цели связаны с теми качествами ученика, которые целесообразно экстремизировать при обязательном выполнении неэкстремальных целей (равенств и неравенств). Они фактически определяют качество процесса обучения. Например: если η_l — средний балл по l-му предмету, то естественно требовать,

чтобы $\eta_l \to \max$. Если η_l — время обучения l-му предмету, то $\eta_l \to \max$.

Очевидно, что формулировка экстремальных целей Z^* обучения является сугубо индивидуальным процессом и не может быть унифицирована. Все зависит от потребностей субъекта и свойств ученика. Действительно, специфика ученика и его склонностей заставляет по-разному формулировать экстремальные цели управления — например, ускорять или углублять процесс обучения. Индивидуализация процесса обучения ориентирована именно на достижение экстремальных целей.

4.2.2.2. Выделение объекта обучения из среды

Выделение объекта обучения из среды сводится к определению «границы», отделяющей объект от среды, его окружающей. Эта граница целиком и полностью зависит от целей управления Z^* и ресурсов R, выделенных на управление. Ресурсы могут быть временными, стоимостными, энергетическими и т. п.; их вид и объем определяют «размеры» объекта, причем необязательно «большие» ресурсы нужны «большему» объекту — все зависит от целей управления Z^*.

При решении задач обучения процесс выделения объекта из среды вовсе не тривиален, как может показаться с первого взгляда. Только в очень простых системах обучения объектом является сам ученик. Действительно, социальные связи ученика естественным образом включаются в процесс обучения, что расширяет объект обучения на микро-, а иногда и макроокружение обучаемого. Это дает возможность в процессе обучения воздействовать на обучаемого не только непосредственно, но и через его среду, которая становится в данном случае объектом управления. С другой стороны, целевые функционалы, входящие в цели Z^* (4.2.5), могут быть определены на поведении ученика, т. е. проявляться в процессе его общения со средой. Это заставляет вводить «среду общения» ученика в объект управления.

Таким образом, цели Z^* обучения и ресурсы R, выделенные для достижения этих целей, определяют объект обучения, включающий вместе с учеником его среду. «Размеры» этой среды зависят от целей и ресурсов обучения.

4.2.2.3. Структурный синтез модели объекта обучения

Данный этап связан с определением структуры модели (4.2.3) объекта. Эта модель необходима для синтеза эффективного управления объектом. При управлении сложным объектом

эффективность управления прямо зависит от адекватности синтезируемой модели. Именно поэтому такое большое внимание уделяется процессу синтеза модели сложного объекта. Под структурой модели F будем подразумевать:

1) структуру входа модели X, т. е. содержательное описание его компонент, доступных наблюдению (измерению с помощью датчика D_X):

$$X' = (x_1, ..., x_n); \tag{4.2.6}$$

2) структуру управления U, т. е. содержательное описание его составляющих:

$$U = (u_1, ..., u_q); \tag{4.2.7}$$

3) структуру выхода модели, т. е. содержательное описание наблюдаемых выходов объекта:

$$Y' = (y_1, ..., y_m); \tag{4.2.8}$$

4) структуру оператора F, т. е. его функциональное описание с точностью до параметров:

$$C = (c_1, ..., c_q). \tag{4.2.9}$$

Это означает, что оператор модели F представляется в виде пары

$$F = \langle S, C \rangle, \tag{4.2.10}$$

где S — структура модели F, а C — ее параметры [4.2.9].

Таким образом, на стадии структурного синтеза модели строится ее описание в виде

$$Y' = f(X', U, C), \tag{4.2.11}$$

где f — выбранный оператор, параметры которого C предстоит определить на следующих этапах. При этом, естественно, должна быть определена структура входов X' (4.2.6), U (4.2.7) и выхода Y' (4.2.8).

В задачах обучения модель ученика F представляет собой оператор, определяющий его реакцию Y' в контролируемой обстановке X', в которой происходит обучение, на обучение U. Это обучение обычно имеет две составляющие

$$U = \langle U_1, U_2 \rangle, \tag{4.2.12}$$

где U_1 — порция обучающей информации, которую следует усвоить ученику, а U_2 — контрольные вопросы, на которые ученик отвечает в форме Y'. Заметим, что датчик информации D_Y о

состоянии Y ученика в данном случае кодирует его ответы, а D_X кодирует обстановку и специфику процесса обучения.

Простейшая модель ученика, например, определяется вероятностью незнания этим учеником соответствующих элементов обучающей информации:

$$P_N = (p_i^N, ..., p_l^N), \qquad (4.2.13)$$

где p_i^N — вероятность незнания i-го элемента информации (всего их l) в N-й момент времени. В этом случае Y' — его ответы по поводу знания того или иного элемента информации, которые представляют собой l-мерный вектор, состоящий из нулей («не знает»), единиц («знает») и прочерков («вопрос не задавался»). Сама модель представляет собой правило изменения вектора вероятностей (4.2.13) при определенном наборе порций обучающей информации. Состояние среды X' эта модель не учитывает.

4.2.2.4. Идентификация параметров *C* модели объекта обучения

На данном этапе определяются параметры (4.2.9) по наблюдениям поведения объекта в режиме нормальной эксплуатации, т. е. при отсутствии управления. Наблюдения

$$\langle X', Y' \rangle \quad (i = 1, ..., L) \qquad (4.2.14)$$

являются источником необходимой информации для идентификации параметров C. Очевидно, что при этом поведение среды X должно быть достаточно разнообразным, чтобы проявились свойства объекта, необходимые для идентификации C.

В задаче обучения этапу идентификации соответствует оценка свойств ученика во внеучебной обстановке, т. е. во время его общения со средой. Наблюдения такого рода дают очень ценную информацию о специфике ученика. Однако здесь большие трудности возникают при фиксации наблюдений, т. е. в организации системы сбора информации с помощью датчиков D_X и D_Y (см. рис. 4.2.1), поставляющих необходимую информацию (4.2.14). Именно поэтому идентификация как этап формального процесса обучения обычно не используется, хотя в неформализованных системах обучения ей уделяется большая роль, так как здесь поведение Y ученика не «зашумлено» системой обучения, часто заставляющей его вольно или невольно искажать информацию о себе, сообщаемую учителю: Опытные педагоги знают этот эффект и предпочитают при выяснении сложных обстоятельств «пассивное» наблюдение активному «допросу».

Однако далеко не все интересующие управление параметры C могут быть определены в режиме идентификации. Именно в этом случае необходимо обратиться к экспериментам с объектом.

4.2.2.5. Планирование экспериментов с объектом

Определяются параметры C модели в режиме специально организованных экспериментов. При этом определяются область Ω допустимых изменений варьируемого входа:

$$U \in \Omega \quad (4.2.15)$$

и критерий эффективности эксперимента, который задается на дисперсионной матрице D искомых коэффициентов $C = (c_1, \ldots, c_k)$. Под планом $\mathcal{U}$ эксперимента подразумевается набор входов объекта

$$\mathcal{U} = \{U_1, \ldots, U_N\}, \quad (4.2.16)$$

которые следует реализовать в эксперименте, чтобы по полученным наблюдениям выхода объекта оценить искомые параметры C. Для определения оптимального плана $\mathcal{U}^*$ необходимо решить задачу оптимизации

$$K(\mathcal{U}, S) \to \underset{\mathcal{U} \in \Omega}{\text{extr}} \Rightarrow \mathcal{U}^*, \quad (4.2.17)$$

где K — критерий эффективности плана, заданный на плане $\mathcal{U}$ и структуре модели S. Например, при минимизации объема эллипсоида, образованного рассеянием ошибок параметров С, имеет место так называемый D-оптимальный план (см. также § 4.4).

Теория оптимальных планов таких экспериментов разработана для объектов сравнительно простой, чаще всего полиномиальной структуры. Для более сложных объектов, к которым принадлежат объекты обучения, теории оптимального планирования экспериментов еще не существует.

В задачах обучения функцию планирования эксперимента с объектом выполняет процедура синтеза тестов (контрольных заданий, вопросов, задач и т. д.), с помощью которых определяются параметры С модели ученика. Естественно, что тесты должны быть построены таким оптимальным образом, чтобы при минимальном числе и минимальной сложности тестов ответы на них несли максимальную информацию о параметрах — а иногда и структуре — модели ученика.

Простейшим и типичным примером такого теста являются экзаменационные вопросы и задачи, позволяющие оценить уровень знаний ученика. В этом случае тест определяет один пара-

метр ученика, называемый его «оценкой», измеренной в порядковой шкале. Такая грубая, даже примитивная оценка свойств ученика, естественно, не может быть положена в основу формализованной системы обучения и служит очень слабой обратной связью в процессе обучения.

Итак, три последних этапа управления связаны с синтезом модели F объекта управления (обучения). Такая модель позволяет построить оптимальное обучение.

4.2.2.6. Синтез оптимального обучения

Суть этапа синтеза заключается в определении управления U^*, реализация которого в объекте дает возможность добиться заданной цели управления, — если, разумеется, модель объекта верна. Делается это следующим образом.

Целевые функционалы в (4.2.1) определены на наблюдаемых состояниях Y' объекта, т. е. цель Z^* имеет вид

$$Z^*: \begin{cases} \varphi_i(Y') \geqslant a_i \quad (i=1, \ldots, k_1); \\ \psi_j(Y') = b_j \quad (j=1, \ldots, k_2); \\ \eta_l(Y') \to \text{extr} \quad (l=1, \ldots, k_3). \end{cases} \tag{4.2.18}$$

Подставляя в выражение (4.2.18) полученную модель F, приходим к многокритериальной задаче оптимизации

$$\eta_l(F(X', U)) \underset{U \in \Xi}{\to} \text{extr} \qquad (l=1, \ldots, k_3), \tag{4.2.19}$$

где Ξ — множество допустимых управлений, определяемое следующими соотношениями:

$$\Xi: \begin{cases} \varphi_i(F(X', U)) \geqslant a_i \qquad (i=1, \ldots, k_1); \\ \psi_j(F(X' \; U)) = b_j \qquad (j=1, \ldots, k_2); \\ U \in R. \end{cases} \tag{4.2.20}$$

При решении этой задачи прежде всего следует осуществить свертку экстремальных целей (4.2.19), для чего необходимо иметь дополнительные сведения о весомости целей. В результате получаем стандартную вариационную задачу:

$$Q(X', U) \underset{U \in \Xi}{\to} \min \Rightarrow U^*, \tag{4.2.21}$$

где $Q(\cdot, \cdot)$ — скалярная функция свертки экстремальных критериев, а U^* — оптимальное управление. Если целевые критерии являются функциями, то задача (4.2.21) является задачей математического программирования.

Решение задачи (4.2.21) при известном состоянии среды X' не представляет принципиальных трудностей. Однако следует помнить, что эта задача для достаточно сложных объектов управления обычно является многоэкстремальной и овражной, что заставляет использовать для ее решения специальные методы поисковой оптимизации — например, методы случайного поиска [161, 166].

В процессах обучения синтез оптимального обучения производится аналогично путем решения оптимизационной задачи (4.2.21), причем здесь U^* — обучающая информация, оптимальная для данного ученика. В этом проявляется индивидуальность процесса обучения по описанной схеме.

4.2.2.7. Реализация обучения

Здесь происходит «введение» полученного на предыдущем этапе управления в объект. Эта процедура, не представляющая принципиальных трудностей при управлении техническими системами, в процессах обучения становится трудноразрешимой. Действительно, данный этап соответствует процессу заучивания и усвоения учеником заданного учебного материала. Обычно трудно рассчитывать на добросовестность ученика и все «педагогические» средства в основном направлены на то, чтобы этот этап обучения был выполнен в максимальной мере. Иначе говоря, решается следующая оптимизационная задача:

$$\| U - U^* \| \to \min, \qquad (4.2.22)$$

где U — информация, воспринятая учеником, т. е. заученная и осмысленная им.

По окончании процесса реализации управления объект переходит в новое состояние Y, которое, как правило, отличается от искомого Y^*. Дело здесь в том, что ввиду сложности объекта управления (см. § 2.1) действуют мешающие факторы, а именно:

1. Приближенность модели F, т. е.

$$F \neq F^0, \qquad (4.2.23)$$

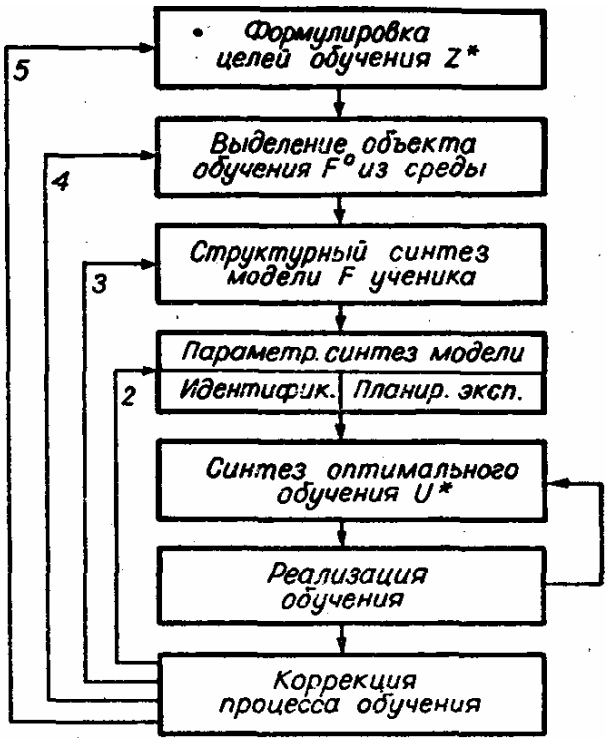
где F^0 — оператор объекта.

2. «Зашумленность» информации о состоянии среды X' и объекта Y', получаемой УУ, приводит к искажению синтезированной модели даже в том случае, когда ее структура в точности соответствует структуре объекта F^0.

3. Дрейф характеристик объекта приводит к тому, что модель всегда «отстает» от объекта F^0 и даже в принципе не может быть адекватной ему (разве что случайно). Этот дрейф может

быть вызван предыдущими шагами управления, что особенно характерно для систем обучения.

Легко видеть, что ученик является именно таким объектом управления; его модель всегда приближенна, ответы «зашумлены» различными посторонними и второстепенными факторами, которые в обилии сопровождают всякий процесс обучения, и, наконец, его свойства интенсивно изменяются в процессе обучения. Все эти обстоятельства, а также ограниченность ресурсов R обучения заставляют прежде всего повторять процесс синтеза обучения (стрелка *1* на рис. 4.2.2). Очевидно, что при этом учитывается новое состояние среды и ученика.



Рис, 4.2.2. Этапы обучения как процесса управления сложным объектом.

Это простейший цикл управления, который применяют для простых объектов. Сложные объекты управления, изменяющиеся во времени, требуют введения этапа коррекции или адаптации (см. рис. 4.2.2).

4.2.2.8. Коррекция (адаптация)

На данном этапе необходимо откорректировать (адаптировать) систему управления (обучения) в связи с изменением свойств объекта. Эта коррекция может затронуть параметры модели (стрелка *2* на рис. 4.2.2). В этом случае можно воспользоваться обоими рассмотренными подходами.

Прежде всего используется идентификация, т. е. коррекция параметров модели на основе информации, полученной в процессе управления (обучения ученика). Производится адаптация параметров моделей объекта с целью добиться максимального соответствия модели и изменяющегося объекта. Именно поэтому управление с коррекцией такого рода обычно называют управлением с адаптируемой моделью.

В процессах обучения подобная коррекция представляет собой, например, поправку в скорости изменения вероятности (4.2.13).

незнания определенного материала при его повторении или при отсутствии обучения. В первом случае (при повторении) эта скорость, естественно, уменьшится, а во втором (при необучении) — останется неизменной, причем степень такого изменения скорости очевидным образом зависит от результатов экзамена (теста).

Планированием эксперимента на этапе корреляции пользуются тогда, когда указанной информации недостаточно и для коррекции модели в процессе управления следует поставить дополнительные эксперименты с объектом. В этом случае управление (обучение) имеет двойственный (дуальный) характер: с одной стороны, оно изменяет состояние объекта для достижения целей управления Z^*, а с другой — специальным образом возмущает объект, чтобы полученная при этом информация позволила откорректировать параметры модели. Естественно, что эти дополнительные возмущения должны быть минимальны, так как они неизбежно нарушают процесс достижения поставленных целей управления. Управление и обучение с коррекцией такого рода обычно называют дуальным.

Типичным примером приемов такого рода дуального обучения являются так называемые «контрольные работы» в школьной системе образования. Они позволяют выявить специфику учащихся, но к процессу обучения имеют лишь косвенное отношение, поскольку последующей программы обучения практически не изменяют.

Однако изменение ученика в процессе обучения может затронуть и его структуру. В этом случае описанная выше параметрическая коррекция не будет эффективна и следует обращаться к коррекции структуры модели ученика (стрелка *3* на рис. 4.2.2). Примером такой коррекции структуры модели является переход от одной модели объекта обучения (ученика) к другой (например, от линейной к нелинейной). В этом случае естественно говорить об адаптивной структуре модели объекта обучения.

Если и эта мера не позволяет эффективно управлять, то обращаются к коррекции объекта F^0, т. е. к ревизии границы, отделяющей объект от окружающей его среды (стрелка *4* на рис. 4.2.2). В случае обучения это означает, что следует включить в систему обучения ближайшую среду ученика, его служебные и семейные связи и т. д. Примером такого рода коррекции является установление связи микросоциальной среды учащегося с процессом обучения. Использование обучающих примеров, построенных на информации о среде ученика, и есть простейшая коррекция границ объекта обучения.

И наконец, если и это не позволяет управлять объектом так, как необходимо, приходится обращаться к коррекции целей обучения Z* (стрелка 5 на рис. 4.'2.2). В процессе такой коррекции

определяется новое множество целей, которое достижимо данной системой обучения для имеющегося конкретного ученика F^0. Эти. новые цели могут не удовлетворять потребности субъекта, который тогда должен либо создавать новую систему обучения либо изменять свои потребности. В действительности используются оба указанных пути. Например, часто довольствуются уже созданной системой обучения, несмотря на то что давно ясна ее неудовлетворительность с точки зрения потребностей общества. Как видно, процесс управления сложным объектом и процесс обучения имеют в своей основе много общего. Эта общность настолько глубока, что позволяет достаточно формально использовать методы управления сложными объектами в процессах обучения. Проиллюстрируем это на примере заучивания странной лексики при обучении языку.

4.2.3. Система обучения с адаптивной моделью

Начнем с конкретной задачи. Большинство научных работников, начинающих изучать иностранный язык, имеют потребность понимать литературу по своей специальности для получения нужной информации с минимальными затратами на обращение к словарю. Понимание читаемого текста невозможно без определенного словарного запаса. Известно [19], что для свободного понимания текста большой протяженности достаточно знания. Не более 95% слов текста. Накопление этого лексического запаса и есть главная и наиболее трудоемкая задача в изучении языка. Ощущение «легкости» при чтении иностранного текста дает именно переход через этот порог. Отсюда следует весьма определенная постановка задачи обучения: заучивать слова и выражения иностранного языка так, чтобы затратить минимальное время для достижения «95%-ного порога» понимания лексики в интересующем ученика тексте.

Обобщим сказанное на более широкий класс задач обучения. Представим процесс обучения в виде последовательности шагов или уроков, происходящих соответственно в моменты времени $t_1,...,t_N,...$ На N-м уроке учитель сообщает ученику некоторую порцию обучающей информации (ОИ). Ученик изучает (заучивает) ее, а на следующий день происходит проверка (экзамен), результаты которой представляются в виде Y_N. Используя ответы ученика, или результаты экзамена Y_N, учитель согласно алгоритму обучения определяет новую порцию обучающей информации, которую и сообщает ученику на очередном (N+1)-м уроке. Таким образом, процесс заучивания представляет собой обмен информацией между учеником и учителем, причем Y_N -реакция ученика на обучающее воздействие U_N учителя. Пра-

вило синтеза U на основе ответов ученика Y, цели Z^* и ресурсов обучения R и является алгоритмом обучения A:

$$U = A\ (Y, Z^*, R) \qquad (4.2.24)$$

(состояние среды для простоты не учитываем).

Будем рассматривать такие процессы обучения, в которых обучающая информация (ОИ) разбита на пронумерованные порции. В этом случае UN представляет собой множество номеров

$$U_N = \{i_1, ..., i_{Mn}\},\ i_j \neq i_l \text{ при } j \neq l,\ i_j \in \{1, \ldots, n\}, \qquad (4.2.25)$$

где n — объем всей обучающей информации, M_N — объем порции ОИ, сообщаемой на уровне U_N ($1 \leq M_N \leq n$). Элементом ОИ может быть, например, правило, определение, задача и т. д. В задаче обучения иностранному языку элементами ОИ могут быть иностранные слова, словосочетания, грамматические правила, упражнения и т. д.

Целью обучения является минимизация времени обучения до заданного уровня обученности. Пусть T — время достижения этого уровня. Тогда целью обучения Z^* является

$$Z^*: \begin{cases} L \geqslant 0{,}95; \\ T \to \min, \end{cases} \qquad (4.2.26)$$

где L — уровень обученности, т. е. относительное число неизвестных ученику лексических единиц в определенном тексте.

Для описания динамики запоминания учеником ОИ воспользуемся естественными свойствами человеческой памяти: запоминанием и забыванием. При этом используем данные психологии, которая рассматривает поведение, и в частности обучение, как стохастический процесс [19].

4.2.4. Модель ученика

Знания ученика после N-го урока определяются состоянием его памяти, которое будем описывать вектором вероятностей незнания всех элементов ОИ:

$$P_N = (p_1\ (t_1^N), ..., p_n\ (t_n^N)\ ), \qquad (4.2.27)$$

где $p_i\ (t_i^N)$ — вероятность незнания i-го элемента ОИ в момент t_i^N, который отсчитывается от момента последнего заучивания i-го элемента. Полагаем, что в момент запоминания учеником заданной порции ОИ он помнит элементы этой порции ОИ с единичной вероятностью, т. е. $p_i(t_N) = 0$, $i \in U_N$. С течением времени происходит забывание, т. е. вероятность незнания возрастает.

Проверка (экзамен) запоминания порции U_N представляется в виде двоичного набора:

$$Y_N = \{y_{i_1}^N, \ldots, y_{i_{M_N}}^N\}, \quad (4.2.28)$$

где $i_j \in U_N$ ($j = 1, ..., M_N$),

$$y_{i_j}^N = \begin{cases} 0, & \text{если ученик вспомнил } i_j\text{-й элемент ОИ при проверке,} \\ 1, & \text{если ученик не вспомнил } i_j\text{-й элемент ОИ при проверке.} \end{cases}$$

Эти ответы используются для преобразования вектора P_N:

$$P_{N+1} = F(P_N, U_N, Y_N, C_N), \quad (4.2.29)$$

.

где C_N — вектор параметров ученика, характеризующих индивидуальные особенности его памяти на N-м шаге. Правило преобразования F естественно назвать моделью ученика.

Для задачи обучения запоминанию иностранных слов [196]. а затем и для задачи обучения пониманию текстов на иностранном языке модель F определялась в соответствии с формулой [19]:

$$p_i(t_i^N) = 1 - \exp(-\alpha_i^N t_i^N), \quad (4.2.30)$$

где a_i^N — скорость забывания i-го элемента ОИ на N-м шаге; t_i^N — время с момента последнего заучивания i-го элемента ОИ. Параметры $\alpha_i (i = 1, ..., n)$ образуют вектор C параметров ученика.

Скорость забывания α_i каждого элемента ОИ, очевидно, уменьшается, если этот элемент заучивается, и не изменяется а противном случае, т. е. имеет место следующая зависимость:

$$\alpha_i^{N+1} = \begin{cases} f_i(\alpha_i^N), & \text{если } i \in U_N; \\ \alpha_i^N, & \text{если } i \notin U_N, \end{cases} \quad (4.2.31)$$

где относительно функции f_i известно лишь то, что $\alpha < f_i(\alpha)$. Поэтому приходится воспользоваться линейной аппроксимацией f. Эта аппроксимация опирается на следующие очевидные соображения: скорость забывания α_i уменьшается при заучивании, но в разной мере: если ученик вспомнил i-е слово на экзамене, то естественно считать, что уменьшение скорости забывания этого слова больше, чем в противоположном случае. Это предположение определяет следующую рекуррентную формулу адаптации параметров C ученика:

$$\hat{\alpha}_i^{N+1} = \begin{cases} \gamma''' \hat{\alpha}_i^N, & \text{если } i \notin U_N; \\ \gamma' \hat{\alpha}_i^N, & \text{если } i \in U_N \text{ и } y_i^N = 0; \\ \gamma'' \hat{\alpha}_i^N, & \text{если } i \in U_N \text{ и } y_i^N = 1, \end{cases} \quad (4.2.32)$$

где γ', γ'' и γ''' — параметры $0<\gamma'<\gamma''<1\leq\gamma'''$, а $\hat{\alpha}_i^0>0$ — оценка начальной скорости забывания i-го элемента ОИ.

Так как на каждом N-м уроке i-й элемент ОИ либо выдается для запоминания ($i\in U_N$), либо нет ($i\notin U_N$), а память ученика обладает свойством забывания, то в модели (4.2.30) необходимо учитывать время забывания t_i^N информации после ее заучивания в момент $t_i^m=0$ ($0\leq m\leq N$). При этом

$$t_i^{N+1}=\begin{cases}\Delta t_N, & \text{если } i\in U_N;\\ t_i^N+\Delta t_N, & \text{если } i\notin U_N,\end{cases}$$

где $\Delta t_N = t_{N+1} - t_N$ — интервал времени между уроками U_N и U_{N+1}; t_1, t_2, ..., t_N, ... — моменты получения и заучивания ОИ. Доведение вероятности незнания i-го элемента ОИ показано на рис. 4.2.3 (в моменты t_2 и t_5 i-й элемент не выдавался на заучивание).

Введем критерий качества обучения, который содержательно соответствует вероятности незнания лексической единицы, наугад выбранной из заданного текста:

$$Q_N=\sum_{i=1}^{n} p_i(t_i^N)\,q_i, \qquad (4.2.33)$$

где q_i ($0<q_i<1$, $\sum_{i=1}^{n} q_i=1$) — встречаемость (частота) i-го элемента ОИ в данном тексте.

Задачей обучения Z^* является минимизация критерия качества обучения (4.2.33). Процесс обучения целесообразно заканчивать, когда $Q_N\leq\delta$, где в соответствии с формулой (4.2.26)

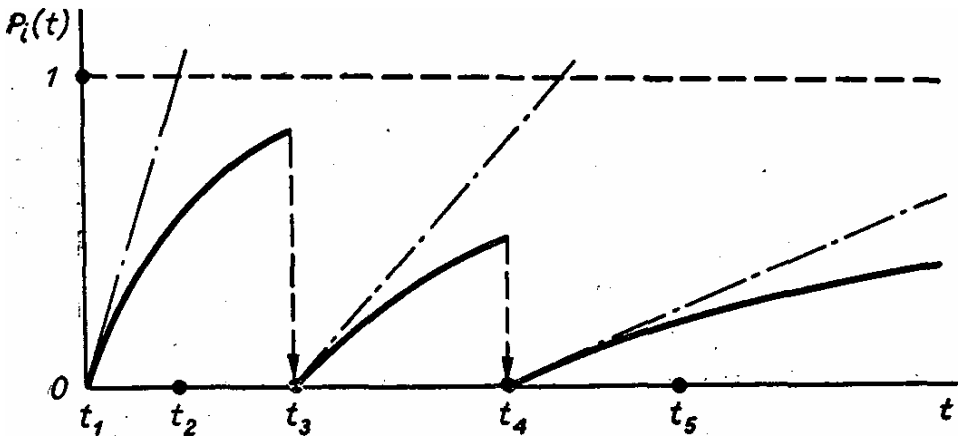


Рис. 4.2.3. Динамика забывания i-й порции ОИ в процессе обучения.

δ=0,05. Поэтому в первую очередь необходимо выдавать на заучивание ту информацию, которую ученик хуже помнит, с учетом ее встречаемости q_i.

Алгоритм минимизации Q на каждом шаге состоит в следующем очевидном правиле выбора:

$$\max_{1\leqslant i\leqslant n} p_i(t_i^N)q_i = p_{j_1}(t_{j_1}^N)q_{j_1};$$

$$\max_{\substack{1\leqslant i\leqslant n,\\ i\neq j_1}} p_i(t_i^N)q_i = p_{j_2}(t_{j_2}^N)q_{j_2};$$

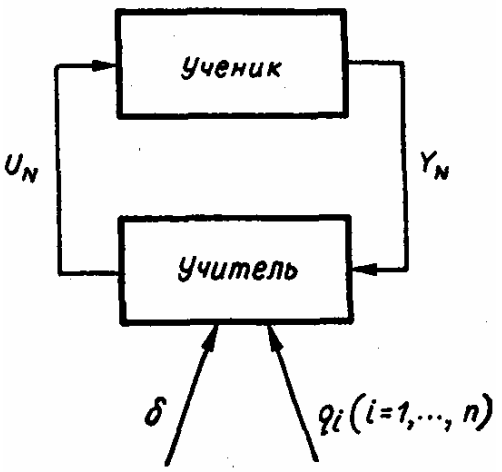


Рис. 4.2.4. Блок-схема обучения.

(4.2.34)

$$\cdots\cdots\cdots\cdots$$

$$\max_{\substack{1\leqslant i\leqslant n,\\ i\neq j_1,\dots,i\neq j_{M_N-1}}} p_i(t_i^N)q_i = p_{j_{M_N}}(t_{j_{M_N}}^N)q_{j_{M_N}}.$$

Множество номеров $\{j_1, \dots, j_{Mn}\}$ и есть U_{N+1} — порция ОИ, которую необходимо выдать для заучивания на (N+1)-м уроке. Она учитывает индивидуальные свойства ученика и текста.

Таким образом, алгоритм обучения состоит в следующем:

1. В результате проверки знания учеником порции UN обра зуется множество Y_N (4.2.28) (рис. 4.2.4).
2. По формуле (4.2.32) происходит адаптация параметров па мяти ученика.
3. По правилу (4.2.30) изменяются вероятности незнания элементов ОИ и образуется P_{N+1}.
4. По формуле (4.2.33) вычисляется критерий качества об учения Q_{N+1}.
5. Если $Q_{N+1}\leq\delta$, то обучение заканчивается. В противном случае по правилу (4.2.34) определяется очередная порция ОИ U_{N+1}, которая и выдается ученику для заучивания. Далее про цедура повторяется с пункта 1.

Исследование и доказательство сходимости этого процесса приведено в работе [216].

Рассмотрим поведение процесса обучения на модели.

4.2.5. Модельный анализ процесса обучения

Для моделирования описанного выше процесса обучения необходимо иметь модель ученика и уметь моделировать его ответы на

экзамене.

Моделью ученика в данном случае является описание F (4.2.29), в котором используется зависимость (4.2.31) в виде

$$\alpha_i(t_{j+1}) = \begin{cases} \gamma\alpha_i(t_j) \text{ при заучивании } i\text{-го слова;} \\ \alpha_i(t_j) \text{ в противном случае,} \end{cases} \quad (4.2.35)$$

где $0<\gamma<1$ — параметр ученика.

В качестве модели экзамена выбираем простое соотношение для вероятности «провала» на экзамене по i-му слову в момент t_j:

$$P_i^{\partial}(t_j) = lp_i(t_{j-1}). \quad (4.2.36)$$

Это вероятность того, что ученик не будет знать i-е слово на экзамене в момент t_j, $0<l<1$ — величина, показывающая, во сколько раз уменьшается вероятность незнания на экзамене после заучивания для данного ученика.

Таким образом, ученик характеризуется двумя параметрами — γ и l, а также начальными значениями скоростей забывания $\alpha_i(0)$ (i=1,...,n). Введем критерии эффективности предложенного процесса адаптации. Ее можно оценивать различным образом —

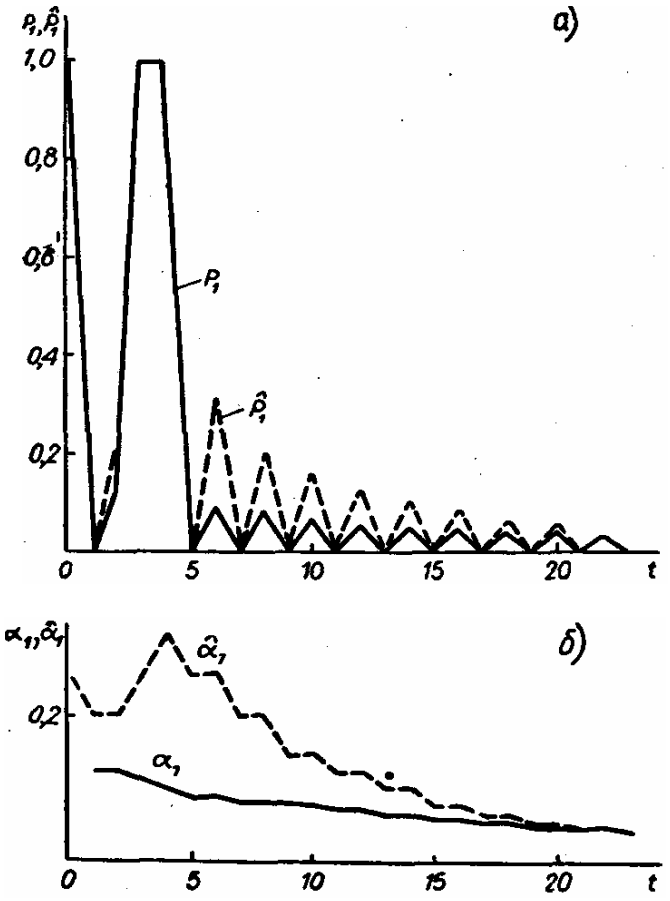


Рис. 4.2.5. Динамика поведения параметров первого слова (i = = 1) в процессе обучения: *а* — изменение вероятности забывания, *б* — изменение параметра скорости забывания.

например, по средней относительной близости скоростей забывания и их оценок, т. е. по формуле

$$\Phi_1(t) = \sum_{i=1}^{n} \frac{|\alpha_i(t) - \hat{\alpha}_i(t)|}{\alpha_i(t)} q_i, \quad (4.2.37)$$

где введен вес q_i. Очевидно, чем меньше эта величина, тем эффективнее адаптация.

Другая оценка эффективности связана с определением относительного числа слов, которые должны быть выданы на обучение. Пусть на основе имеющихся оценок $\hat{p}_i(t)$ выдано $m(t) = = m_1(t)+m_2(t)$ слов на обучение. Здесь m_1 — необходимые слова, входящие в число рекомендуемых к заучиванию, при условии, что были известны точные значения $p_i(t)$. Имея модель ученика, мы можем точно определить, что ему нужно учить. Сравнивая «предложения» алгоритма и действительные «потребности» ученика, можно оценить эффективность алгоритма отношением

$$\Phi_2(t) = \frac{m_1(t)}{m(t)}. \quad (4.2.38)$$

Очевидно, чем эффективнее работает программа, тем ближе к единице эта величина. В идеале $\Phi_2(t) = 1$.

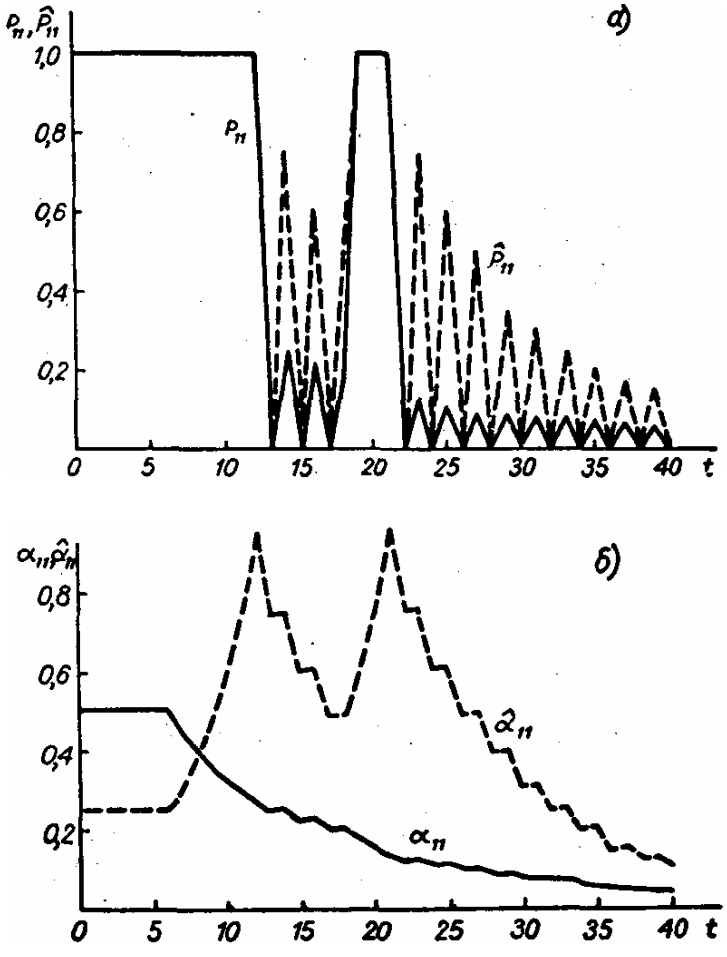


Рис. 4.2.6. Динамика поведения параметров одиннадцатого слова (*i*=11) в процессе обучения. Обозначения те же, что и на рис. 4.2.5.

Теперь рассмотрим модельный эксперимент [196], иллюстрирующий процесс заучивания иностранных слов, на котором показана эффективность адаптации.

Эксперимент был проведен для случая и =12 и гиперболического закона убывания частотности слов (закон Цыпфа):

$$q_i = \frac{c}{i}, \quad (4.2.39)$$

где $c = \left(\sum_{j=1}^{n} \frac{1}{j} \right)^{-1}$. Каждая порция обучения состояла из пяти слов.

Параметры ученика: l=0,5; γ=0,9. Начальные значения скоростей забывания $\alpha_i(0)$ выбирались случайно по равномерному закону в пределах $0<\alpha_i<1/2$. Параметры алгоритма адаптации (4.2.32) были выбраны так: $\gamma'=\gamma''=0{,}8$; $\gamma''' = 1{,}25$.

Динамика процесса адаптации представлена на рис. 4.2.5 и 4.2.6. Хорошо видно, что оценки стремятся к точным значениям, что доказывает эффективность процедуры адаптации.

Характер поведения величины $Q(t)$ и ее оценки $\hat{Q}(t)$ (рис. 4.2.7) показывает, что $\hat{Q}(t) \to Q(t)$, т. е. алгоритм адаптируется.

Далее было исследовано влияние параметра γ' на эффективность процесса адаптации. В качестве меры адаптивности был выбран критерий (4.2.37) на двадцатом шаге обучения, т. е. Φ_1 (20). Результаты моделирования процесса обучения для различных γ' ($\gamma''' = 1/\gamma'$) дают оптимальное значение γ'^*, равное 0,25, что существенно отличается от $\gamma' = 0{,}8$, для которого $\Phi_1(20)=22$. Они показаны на рис. 4.2.8, где $\gamma''=\gamma'$.

Для сравнения эффективности адаптивного ($\gamma' = 0{,}25$) и неадаптивного ($\gamma' = 1$) алгоритмов обучения были произведены рас-

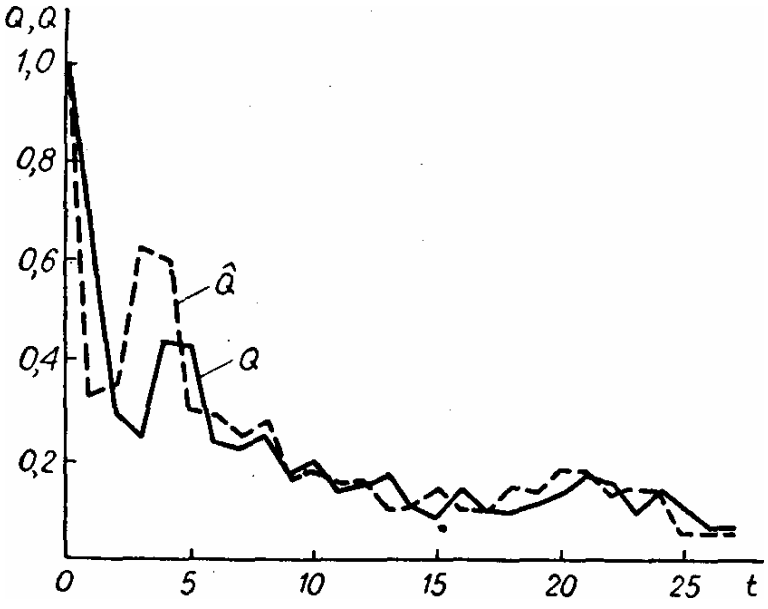


Рис. 4.2.7. Поведение показателей Q и $\hat{Q}$ в процессе обучения.

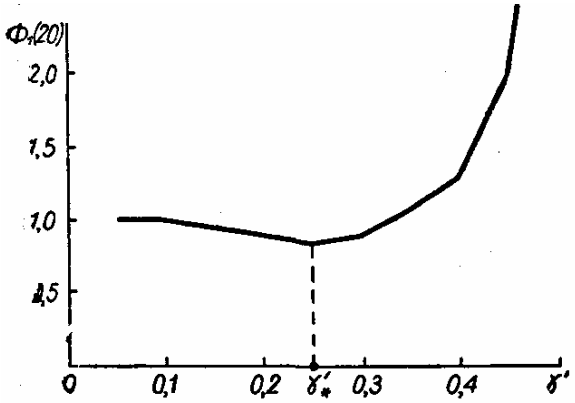


Рис. 4.2.8. Зависимость эффективности $\Phi_1(20)$ от γ'.

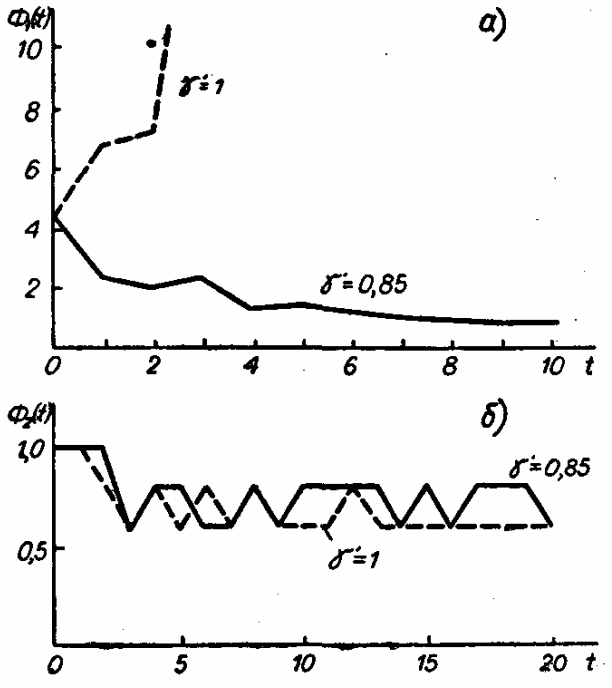


Рис. 4.2.9. Поведение критериев Φ_1 (*а*) и Φ_2 (*б*) *в* процессе обучения.

четы для обоих критериев (4.2.37) и (4.2.38). Из результатов, приведенных на рис. 4.2.9, вытекает, что адаптивный алгоритм эффективнее неадаптивного по обоим критериям, что и следовало ожидать.

Далее рассмотрим другие модели обучения и сравним их с описанной выше (подраздел 4.2.4) для задачи заучивания иностранной лексики.

4.2.6. Экспериментальное сопоставление различных моделей обучения

Существует много моделей обучения. Для сравнения различных моделей был поставлен эксперимент по заучиванию иностранных слов [187]. Эксперимент состоял в следующем.

Испытуемым давались для заучивания сразу 40 слов (n=40) на языке, которого они не знали и никогда раньше не изучали. Слова были подобраны приблизительно одинаковые по трудности запоминания. Время на заучивание ограничивалось. На другой день после заучивания проводился экзамен. Результаты экзамена фиксировались в виде последовательности нулей и единиц (4.2.28). Незапомненные слова доучивались, и на другой день снова проводился экзамен по всем n словам. Такая процедура повторялась до тех пор, пока испытуемый не заучивал все 40 слов.

В результате была построена средняя по всем испытуемым кривая обучения:

$$\bar{T}(t) = \frac{1}{M} \sum_{j=1}^{M} T_j(t), \quad (4.2.40)$$

где t=0,1,... — моменты заучивания и контроля; $\bar{T}(t)$ — средняя доля незапомненных слов по всем испытуемым в t-м испытании; $T(0) = 1$; M — количество испытуемых ($M = 5$);

$$T_j(t) = \frac{1}{n} \sum_{i=1}^{n} y_{ij}^{t} \quad (4.2.41)$$

— доля незапомненных слов в t-м. испытании для j-го испытуемого; y_{ij}^{t} — результат проверки знания i-го слова j-м испытуемым в момент t. Доверительные интервалы для $\bar{T}(t)$ строились стандартным способом (доверительная вероятность $\beta = 0{,}90$). Экспериментальная кривая обучения изображена на рис. 4.2.10.

Опишем теперь модели обучения, построим для каждой теоретическую кривую обучения и определим, какая из моделей лучше всего описывает процесс запоминания слов.

Модель Буша—Мостеллера [39]. Буш и Мостеллер для описа-

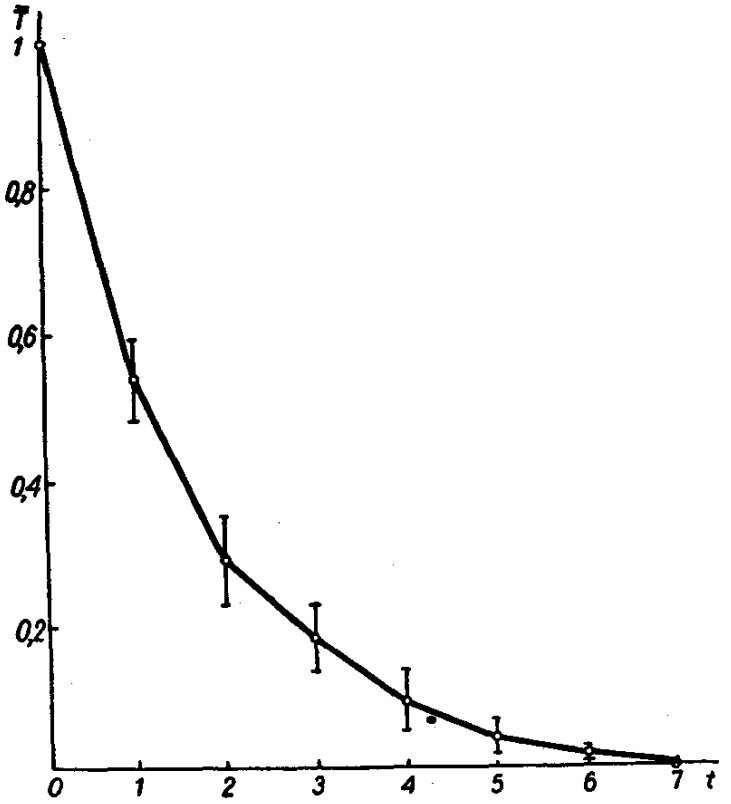


Рис. 4.2.10. Осредненная экспериментальная кривая обучения.

ния процесса обучения вводят оператор изменения вероятности правильного ответа в виде

$$q_i(t+1) = \begin{cases} \gamma_1 q_i(t) + (1-\gamma_1)\lambda_1, & \text{если } y_i^t=0; \\ \gamma_2 q_i(t) + (1-\gamma_2)\lambda_2, & \text{если } y_i^t=1 \end{cases} \quad (i = 1, \ldots, n;\ t = 0, 1, \ldots, ;\ 0<\gamma_1<\gamma_2<1), \quad (4.2.42)$$

где $q_i(t) = 1 - p_i(t)$. Параметры γ_1 и γ_2 характеризуют скорость обучения. Легко проверить, что λ_1 и λ_2 являются неподвижными точками. Если $q_i(t)=\lambda_j$ (j=1,2), то при этом $q_i(t+1)=\lambda_j$, т. е. если вероятность правильного ответа станет равной λ_j, то в последующих испытаниях она больше изменяться не будет.

Рассмотрим теперь два важных частных случая этой модели.

Случай 1. $\lambda_1=\lambda_2=1$. Тогда из выражения (4.2.42) получаем

$$q_i(t+1) = \begin{cases} \gamma_1 q_i(t) + 1 - \gamma_1, & \text{если } y_i^t = 0; \\ \gamma_2 q_i(t) + 1 - \gamma_2, & \text{если } y_i^t = 1. \end{cases} \quad (4.2.43)$$

Предполагается, что $p_i(0) = 1$ для любого i=1, ..., n. Таким образом, данная модель зависит только от двух параметров — γ_1 и γ_2. Здесь γ_1 учитывает запоминание, а γ_2 — забывание ученика. Эти параметры оцениваются по опытным данным: γ_2 оценивается с помощью метода максимума правдоподобия [39]:

$$\hat{\gamma}_2 \approx 1 - \sum_{t=0}^{7} x_t \Big/ \sum_{t=0}^{7} t n_t, \quad (4.2.44).$$

где n_t — число слов, забытых до t-го испытания; x_t — число слов из n запомненных в t-м испытании; для оценки γ_1 используется следующее соотношение [39] для среднего числа неправильных ответов $\bar{T}$ (это интеграл от функции $\bar{T}(t)$):

$$\bar{T} \approx \sum_{t=0}^{7} \bar{T}(t) = \frac{-\ln \dfrac{1-\gamma_2}{1-\gamma_1}}{\gamma_2 - \gamma_1} \quad (4.2.45)$$

По данным описанного выше эксперимента были получены оценки $\hat{\gamma}_1$=0,514; $\hat{\gamma}_2$=0,666 и построена теоретическая кривая обучения, обозначенная темными треугольничками на рис. 4.2.11 (кривая *1*).

Случай 2. $\lambda_1 = 1$; λ_2=0; γ_2=1. Это так называемая *модель Миллера—Мак-Гилла:*

$$q_i(t+1) = \begin{cases} \gamma_1 q_i(t) + 1 - \gamma_1, & \text{если } y_i^t = 0; \\ q_i(t), & \text{если } y_i^t = 1. \end{cases} \quad (4.2.46)$$

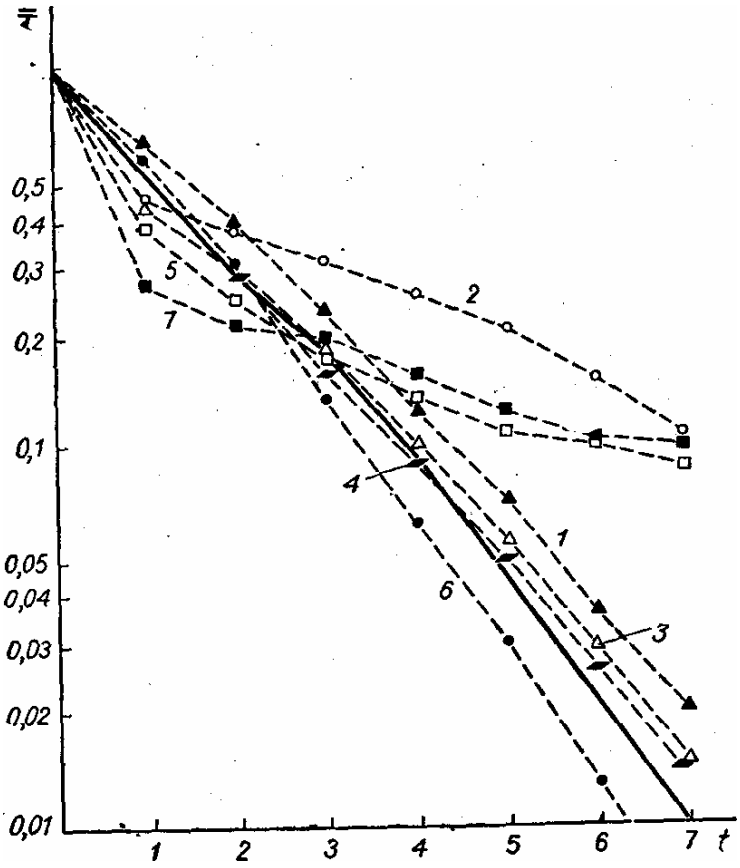


Рис. 4.2.11. Теоретические кривые обучения по разным моделям. Цифры на кривых соответствуют номерам моделей в тексте.

В этом случае вероятность запоминания никогда не уменьшается, а при удачном экзамене (y=0) — увеличивается.

Данная модель имеет неизвестными параметрами $p_i(0)$ и γ_1. Оценку параметра $p_i(0)$ легко получить с помощью метода максимума правдоподобия [39]:

$$1-p_i(0)=\frac{h}{m}, \quad (4.2.47)$$

где

$$m=\sum_{i=1}^{n}\sum_{t=0}^{t_i^*} y_i^t;$$

t_i^* — число неправильных ответов по i-му слову, следующих подряд с начала обучения.

Оценку параметра γ_1 получают из соотношения

$$\bar{T}\approx\frac{-\ln p(0)}{1-\gamma_1}. \quad (4.2.48)$$

Из опытных данных были получены оценки параметров $p(0)$ = = 0,456; γ_1=0,716 и построена теоретическая кривая обучения (рис. 4.2.11, светлые кружочки).

Последующие модели отличаются тем, что в них изменение вероятности ошибки не зависит от реакции ученика, т. е.

$$P_{t+1} = F(P_t, U_t, \gamma). \qquad (4.2.49)$$

Модель Халла [40]. Эта модель основана на предположении, что вероятности незнания в любом случае связаны следующим простым соотношением:

$$p_i(t+1) = \gamma\, p_i(t), \qquad (4.2.50)$$

где параметр γ характеризует степень уменьшения вероятности незнания в результате заучивания i-го слова в t-м испытании. Сравнив формулу (4.2.50) с (4.2.42), заметим, что эта модель является частным случаем модели Буша—Мостеллера. Неизвестным в модели Халла является один параметр γ, который оценивается следующим соотношением [40]:

$$\bar{T} = \sum_{t=0}^{7} p(t) = \frac{1-\gamma^8}{1-\gamma}, \qquad (4.2.51)$$

где $p(0) = 1$. Отсюда получаем значение $\gamma = 0{,}54$. Теоретическая кривая обучения для этой модели обозначена параллелограммами на рис. 4.2.11 (кривая *4*).

Модель Терстоуна [40]. Терстоун предложил следующую нерекуррентную модель обучения:

$$p(t) = \frac{\gamma}{t+\gamma}, \qquad (4.2.52)$$

где $p(t)$ — по-прежнему вероятность незнания в t-м испытании. Эта модель определяется одним параметром γ. который может быть получен из соотношения [40]

$$\bar{T} = \sum_{t=0}^{7} p(t) \approx \gamma \ln \frac{\gamma+7{,}5}{\gamma-0{,}5}. \qquad (4.2.53)$$

Из эксперимента была получена оценка γ=0,66. Теоретическая кривая обучения по этой модели обозначена на рис. 4.2.11 светлыми квадратиками (кривая *5*).

Модель Рестла [40]. Эта модель определяет вероятность неправильного ответа следующим образом:

$$p(t) = \frac{(1-\gamma)^t}{\gamma+(1-\gamma)^t}, \qquad (4.2.54)$$

где γ — неизвестный параметр, оценку которого получаем [40] из соотношения

$$\bar{T} = \sum_{t=0}^{7} p(t) = \frac{\ln \gamma - \ln (\gamma + \sqrt{1-\gamma})}{(1-\gamma) \ln (1-\gamma)}. \qquad (4.2.55)$$

Полученное в эксперименте значение этой оценки $y = 0{,}561$. Теоретическая кривая обучения для этой модели обозначена на рис. 4.2.11 темными кружочками (кривая *б*).

Модель Кричевского [40]. Кричевский выдвинул гипотезу, что после начального периода обучения возникает «внезапная» обученность. Это следует из так называемой теории «скачков». Применительно к процессу запоминания слов эта теория состоит в следующем. Предположим, что в начале опыта ученик не знает *i*-е слово (будем говорить, что он находится в состоянии A_1). Затем он заучивает *i*-е слово и в некотором испытании дает правильный перевод, т. е. переходит в состояние A_2. После этого он дает только правильные ответы, т. е. не выходит из состояния A_2. Тогда вероятность правильного ответа имеет вид

$$q_i(t) = \begin{cases} \gamma_1, & \text{если в } t\text{-м испытании ученик находится в состоянии } A_1. \\ 1, & \text{если в } t\text{-м испытании ученик находится в состоянии } A_2. \end{cases}$$

Далее вводится переходная вероятность:

$$\text{Вер}\,(A_2 \text{ при } t \mid A_1 \text{ при } t\text{-}1) = \gamma_2, \qquad (4.2.56)$$

определяющая вероятность перехода в состояние A_2 в *t*-м испытании при условии, что в (*t*—1)-м испытании ученик находится в состоянии A_1. Для построения теоретической кривой обучения, соответствующей этой модели, используется математическое ожидание величины $\bar{T}_j(t)$— доли незапомненных слов в *t*-м испытании для *j*-го ученика. Из аксиом данной модели следует:

$$M_j \bar{T}_j(t) = \gamma_1 (1-\gamma_2)^t, \qquad (4.2.57)$$

где M_j — оператор осреднения по *j*.

Таким образом, для построения теоретической кривой обучения необходимо знать параметры γ_1 и γ_2. Для оценки этих параметров находят математическое ожидание числа ошибок *T* [40]:

$$\bar{T} = (1-\gamma_1) \frac{1-\gamma_2}{\gamma_2} \qquad (4.2.58)$$

и математическое ожидание величины t — номера испытания, в котором была сделана последняя ошибка:

$$M(t \mid q_i(t+1)=1)=\frac{1-\gamma_2}{\gamma_2}. \quad (4.2.59)$$

Приравнивая математические ожидания и наблюдаемые средние значения, получаем следующие оценки для параметров модели: γ=0,374; γ_2=0,152. Теоретическая кривая обучения для этой модели обозначена на рис. 4.2.11 темными квадратиками (кривая *7*).

И, наконец, описанная в подразделе 4.2.4 модель по экспери ментальным данным имела γ_1=0,514 и γ_2=0,666. Ее теоретиче ская кривая обозначена светлыми треугольничками на рис. 4.2.11 (кривая *3*).

С целью сравнения рассмотренных моделей для каждой из них вычислялась невязка с экспериментальной кривой, изображенной на рис. 4.2.10:

$$\rho_i=\sum_{t=0}^{k-1}|\bar{T}(t)-\bar{T}_i(t)|, \quad (4.2.60)$$

где i — номер модели, i = 1,2, ..., 7; k = 8; $\bar{T}(t)$ — экспериментальная кривая обучения; $\bar{T}_i(t)$ — теоретическая кривая обучения, полученная с помощью i-й модели. Приводим результаты вычислений.

№ модели	Название модели	ρ
1	Буша — Мостеллера	0,406
2	Миллера — Мак-Гилла	0,893
3	Обучения иностранному языку	0,135
4	Халла	0,064
5	Терстоуна	0,458
6	Рестла	0,160
7	Кричевского	0,697

На рис. 4.2.11 сплошной линией изображена экспериментальная кривая; каждая из теоретических кривых имеет номер, соответствующий номеру модели в таблице.

Из приведенных результатов и рис. 4.2.11 видно, что лучше всего описывает исходные данные модель Халла, далее следует предложенная модель обучения иностранному языку. Основное различие моделей состоит в том, что согласно первой из них вероятности незнания в i-м испытании для всех слов одинаковы, а согласно предложенной модели — различны. Очевидно, в этом состоит недостаток модели Халла, так как не все слова запомина-

ются одинаково. Положительный эффект применения модели Халла в рассмотренном выше эксперименте, по-видимому, тем и объясняется, что вероятности незнания слов, предложенных для запоминания, были в среднем примерно равны.

Таким образом, эксперимент показал, что предложенная модель обучения иностранному языку (см. подраздел 4.2.4) пригодна для обучения запоминанию иностранных слов.

4.2.7. Обучение с использованием предложенной адаптивной модели

Описанная выше (в подразделе 4.2.4) процедура была применена для задачи обучения пониманию текстов на иностранном (английском) языке. Была создана система обучения, абонентами которой стали сотрудники Института электроники и вычислительной техники (ИЭВТ) АН ЛатвССР.

В качестве источника информации были выбраны тексты на английском языке по элементарной математике [68]. В результате их обработки с помощью ЭВМ был получен частотный словарь слов и словосочетаний объемом $n = 962$ лексические единицы (ЛЕ). Словарь содержал слова и словосочетания с их транскрипцией, переводом и частотой $q1, ..., q_n$.

Кроме самостоятельных в словарь на правах отдельных ЛЕ включались:

1) словосочетания и фразы, значение которых нельзя опре делить, исходя из значений составных частей данных сочетаний (например, *in the same way*, *hold forth*, *in question* и др.);

2) словосочетания, имеющие терминологический характер (например, *slide rule*, *bar graph*, *extraction of roots* и др.);

3) аналитические формы глаголов (например, *was missing, have advanced*, *would know* и др.).

Включение аналитических форм глаголов в словарь на правах отдельных ЛЕ дает возможность ученику освоить основные грамматические конструкции при помощи часто повторяющихся слов.

Перевод слов и словосочетаний давался по возможности однозначным, с учетом конкретного значения слова в определенном контексте. Некоторые трудности представлял перевод слов с широкой семантикой — часто используемых наречий, предлогов, частиц. В этих случаях в словаре при переводе был дан ряд значений, строго соответствующих контекстуальным.

Перед началом обучения каждого абонента системы выяснялось, каким словарным запасом он обладает. Для этого каждому абоненту предлагалось просмотреть словарь и указать

незнакомые слова и словосочетания. По ответам определялся исходный уровень незнакомой лексики:

$$Q_0 = \sum_{i=1}^{n} p_i^0 q_i, \quad (4.2.61)$$

где q_i — частота i-й ЛЕ в словаре;

$$p_i^0 = \begin{cases} 0, & \text{если абонент знает } i\text{-ю ЛЕ;} \\ 1, & \text{если } i\text{-я ЛЕ ему незнакома.} \end{cases}$$

В результате такого тестирования для каждого j-го абонента определялся объем всей обучающей информации

$$U^j = (i_1^j, \ldots, i_{nj}^j),$$

где i_k^j — номер незнакомого j-му абоненту слова или словосочетания в словаре; n_j — общее число незнакомых для этого абонента ЛЕ в словаре.

Роль «учителя» в системе выполняла программа, реализующая описанный в подразделе 4.2.2 алгоритм обучения. Цель обучения состояла в том, чтобы снизить процент незнакомой лексики Q до 5%, поскольку именно при таком словарном запасе возможно свободное чтение и понимание текста на иностранном языке [108].

На каждом такте обучения с помощью ЭВМ согласно (4.2.34) определялась порция ОИ $U_N = \{i_1, \ldots, i_{Mn}\}$. Слова и словосочетания с этими номерами выдавались для запоминания. При этом были предусмотрены две формы обучения, которые выбирались самим абонентом. В первой ЛЕ выдаются списком с транскрипцией и переводом, во-второй — в виде отдельных карточек, на одной стороне которых была написана ЛЕ, а на другой — ее перевод с транскрипцией русскими буквами.

Абонент заучивал слова и словосочетания, указанные в бланке задания, и на другой день сам проверял их запоминание. При этом в бланке задания, который был одновременно и отчетом, абонент указывал затраченное на заучивание время $T_3(N)$ и время, планируемое им для следующего занятия $T_{\text{ПЛ}}(N+1)$. Эта информация о ресурсе R, которым располагает абонент, используется для определения объема порции ОИ M_{N+1} исходя из следующего приближенного равенства:

$$T_{\text{пл}}(N+1) \approx k_{N+1} \sum_{i \in U_{N+1}} p_i(t_i^{N+1}), \quad (4.2.62)$$

где U_{N+1} — множество элементов порции ОИ, определяемых по правилу (4.2.34); $p_i(t_i^{N+1})$ — вероятность незнания i-й ЛЕ через

время t_i^{N+1} после ее заучивания; k_{N+1} — коэффициент, определяемый адаптивно:

$$k_{N+1} = k_N + \beta (T_з(N) - T_{пл}(N)) \frac{1}{T_з(N)}, \quad (4.2.63)$$

т. е. на каждом шаге осуществлялась коррекция этого коэффициента, с тем чтобы время, необходимое для заучивания данным абонентом получаемой им порции ОИ, соответствовало планируемому им времени. Выражение (4.2.62) опирается на очевидное соображение, что время заучивания одной ЛЕ пропорционально вероятности ее незнания.

На следующем N-м шаге обучения указанные в бланке задания ответы абонента — номера незапомненных им ЛЕ, время $T_З$ и $T_{ПЛ}$, а также дата выполнения задания — вводились в ЭВМ. Дата выполнения задания необходима для определения промежутка времени между уроками, т. е. величины Δt_N, так как темп обучения задавался самим учеником.

Ответы ученика Y_N и интервал времени Δt_N используются для преобразования вектора вероятностей незнания ЛЕ P_N согласна модели данного ученика (4.2.30), (4.2.32). Затем вычислялся критерий качества обучения Q_N (4.2.33), который являлся количественной характеристикой, определяющей процент незнакомой для данного ученика лексики. Если $Q_N \leq 0{,}05$, то обучение заканчивалось. При невыполнении этого неравенства по (4.2.34) определялась очередная порция ОИ.

В модели ученика имелись параметры γ', γ'' и α_i^0 ($\gamma'''=1$), характеризующие индивидуальные особенности памяти абонента. Эти параметры неизвестны и должны быть определены в процессе обучения. Начальные скорости забывания α_i^0 оценивались методом максимального правдоподобия. Оценка неизвестных параметров модели γ' и γ'' в процессе обучения является достаточно трудоемкой процедурой. Поэтому в поставленном эксперименте в качестве γ' и γ'' были взяты оценки, полученные ранее при запоминании иностранных слов в эксперименте, описанном в подразделе 4.2.4.

Для реализации описанной выше процедуры обучения в ИЭВТ АН ЛатвССР был разработан пакет программ, написанных на алгоритмическом языке Фортран-4 для операционной системы ЕС ЭВМ. На основе этого пакета была реализована экспериментальная система обучения языку (ЭСОЯ), опытная эксплуатация которой дала возможность оценить эффективность предложенного способа обучения с моделью [273].

Основными критериями эффективности системы были выбраны среднее время, затраченное абонентом на заучивание

одной ЛЕ, и уровень понимания текста, на основе которого производилось обучение.

Первый критерий (среднее время, затраченное на одну ЛЕ) вычислялся на основе отчетов абонентов о знании словаря, полученных при входе и выходе из системы. Из этих отчетов было определено число новых ЛЕ, которые запомнил абонент. Время, затраченное на их заучивание, было определено по ежедневным отчетам абонента. В результате вычислялось среднее время, затрачиваемое на заучивание одной ЛЕ.

Эксперимент показал, что это время неодинаково для различных абонентов и зависит от того, учил или не учил абонент язык ранее. Так, в группе абонентов, изучавших язык ранее (12 человек), на одну ЛЕ было затрачено среднее время $\bar{t}$= = 0,95 мин, которое изменялось в пределах от 0,3 до 1,56 мин. В группе незнакомых ранее с английским языком (9 человек)

$\bar{t}$ = 1,7 мин, Причем t_{max}= 4 мин.

Для сравнения напомним, что по программе средней школы для заучивания одной ЛЕ предусматривается 30 мин.

Вторым критерием оценки эффективности предложенного способа обучения было понимание английского текста. С этой целью на различных этапах обучения при разных значениях Q из текста случайно выбирались предложения для перевода, качество которого оценивалось экспертным путем по трехбалльной системе: .

«3» — абонент понял смысл и содержание предложения. Перевод правильный (допускаются некоторые несущественные в смысловом отношении ошибки при переводе);

«2» — абонент правильно перевел все отдельные слова предложения, но имеются ошибки, изменяющие смысл предложения. Перевод неточен;

«1» — абонент не понял предложение. Отдельные слова предложения переведены неправильно или вообще не переведены. Перевод предложения отсутствует.

Результаты оценки качества перевода для разных абонентов при различных значениях Q приведены на рис. 4.2.12. Здесь же дугами показана динамика улучшения качества перевода (уменьшение доли незнакомой лексики) в процессе обучения для различных абонентов. На этом рисунке четко просматривается зависимость между уровнями знания лексики Q и понимания текста.

Проведенные эксперименты показали, что для уверенного понимания специального (математического) текста на иностранном языке вполне достаточно знания порядка 75% слов текста (а не 95%, как рекомендовано в работе [108]).

При изложенном способе обучения пониманию текстов на иностранном языке естественно предположить, что для

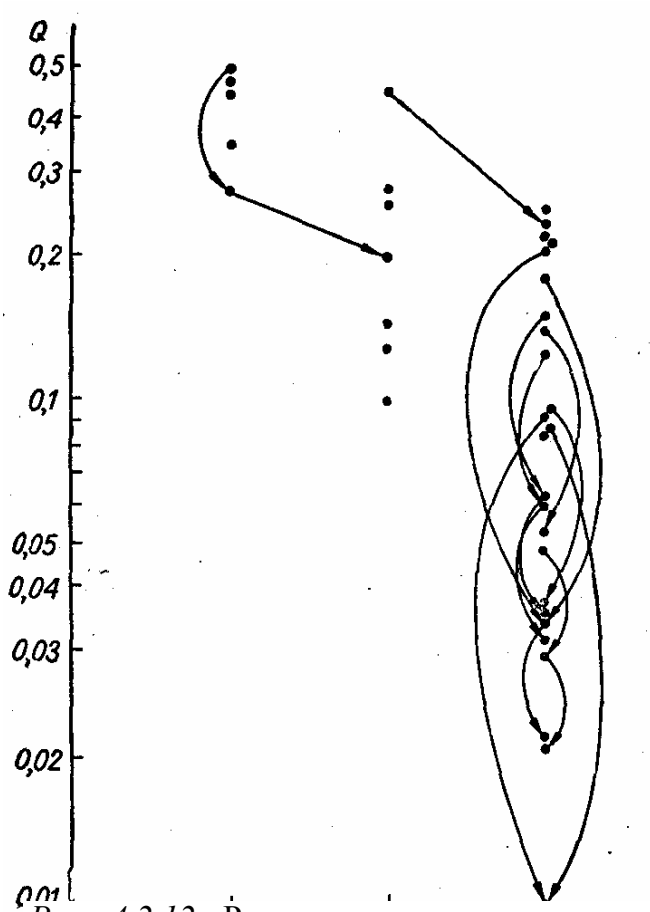


Рис. 4.2.12. Результаты экспериментального обучения.

понимания текста необязательно знание нормативной грамматики — достаточно знание ЛЕ, т. е. слов и словосочетаний, содержащих основные грамматические конструкции. Для проверки этой гипотезы по экспериментальным данным была вычислена корреляция между процентом незнакомой лексики в тексте Q и оценкой качества перевода $b \in \{1, 2, 3\}$

Так как Q изменяется в метрической шкале, а b — в шкале порядка, то для вычисления коэффициента ранговой корреляции необходимо прежде всего преобразовать шкалу Q в шкалу порядка, т. е. ввести ранги Q. Для этого интервал [0; 1} был разбит на 4 части и каждой присвоен свой ранг:

интервал [0; 0,25] соответствовал первому рангу (a = = 1);

интервал [0,25; 0,5] соответствовал второму рангу (a = 2) и т. д. до четвертого (a = 4) ранга.

В результате по данным для абонентов, ранее не изучавших язык, был подсчитан коэффициент ранговой корреляции ρ (по Спирмену) [104]. Он оказался равным 0,45, т. е. очень высоким. Это означает, что существует явная зависимость между знанием лексики и пониманием текста абонентами системы.

В заключение отметим свойства описанной системы обучения, которые выгодно отличают ее от других:

1. Адаптивность, которая выражается в том, что система строит модель абонента, с помощью которой обучает его наи лучшим образом, приспосабливаясь к индивидуальным особен ностям памяти абонента.

2. Анонимность: абонент фигурирует в системе под своим номером.

3. Полное отсутствие в системе преподавателя. (Описанная выше процедура экспертной оценки правильности перевода по надобилась лишь для того, чтобы показать достаточность за-

учивания лексики, и в серийном варианте системы обучения не предусматривается.)

4. Темп обучения задается абонентом в виде указания вре мени, выделяемого им для заучивания очередной порции ОИ.

5. Процесс заучивания может производиться в любых усло виях, удобных абоненту, и никак не связан с системой. Общение с системой происходит лишь в момент получения очередного за дания и может осуществляться с помощью телетайпа, дисплея и других средств общения с ЭВМ.

§ 4.3. Адаптивный синтез многопороговых логических элементов методом случайного поиска

Пороговая логика является сравнительно новой перспективной областью кибернетики, связанной с теорией адаптивных и самоорганизующихся систем, распознаванием образов, искусственным интеллектом и робототехникой, конструированием универсальных модулей ЭВМ и принципиально новых методов обработки информации [257].

Большие функциональные возможности пороговых схем известны давно [77, 242, 257], однако трудности физической реализации надежных в работе пороговых устройств на уровне технологии 60-х годов вызвали пессимистические настроения у инженеров-практиков. В последние годы теоретические исследования и создание технических схем (в том числе многозначных) показывают, что этот временный кризис постепенно преодолевается благодаря успехам техники и более глубокому пониманию природы «ненадежности», компенсируемой простым механизмом многофункциональной перенастройки — структурной и параметрической. Это позволяет переоценить и конкретизировать как сферу применения пороговых устройств (например, в адаптивных системах управления), так и способы их реализации. Многопороговые схемы в интегральном исполнении могут найти широкое применение в логических, арифметических и запоминающих узлах ЭВМ [77, 85, 134, 146]. Разработка данных узлов на основе порогового базиса является перспективной также вследствие его однородности, функциональной полноты, единства задач анализа, синтеза, диагностики и настройки, удобства их решения на ЭВМ

Кроме того, фактические результаты и методология современной пороговой логики могут оказаться эффективным инструментом для решения фундаментальных проблем нейрокибернетики (анализ и моделирование нейронных ансамблей,

целенаправленный синтез нейронных структур с заданными свойствами), психофизики (анализ пороговой зоны, прямое многомерное шкалирование чувствительности сенсорных систем) и других научных направлений, исследующих функционально-поведенческие и нейрофизиологические процессы.

Формальная теория пороговых структур еще недостаточно развита ввиду сложности и новизны проблематики. Ниже рассматриваются методы анализа и синтеза пороговых структур методами случайного поиска, их специфические свойства и проявление этих свойств в задачах случайного поиска.

4.3.1. Пороговый логический элемент (ПЛЭ)

Математическая модель ПЛЭ, впервые рассмотренная в [128], схематически показана на рис. 4.3.1. Здесь двоичные входы $x_1, ..., x_n$ образуют вектор входа

$$X = (x_1, ..., x_n), \tag{4.3.1}$$

где $x_i \in \{0; 1\}$. Вектор весов

$$U = (u_1, ..., u_n) \tag{4.3.2}$$

и порог τ однозначно определяют выходную логическую (булеву) функцию $f(X)$ порогового элемента:

$$y = f(X) = \begin{cases} 0, & \text{если } l(X) < \tau, \\ 1, & \text{если } l(X) \geqslant \tau, \end{cases} \tag{4.3.3}$$

где

$$l(X) = \sum_{i=1}^{n} u_i x_i. \tag{4.3.4}$$

Таким образом, пара

$$\langle U, \tau \rangle \tag{4.3.5}$$

определяет структуру некоторой логической функции. Образование этой функции удобно проиллюстрировать в n-мерном прост-

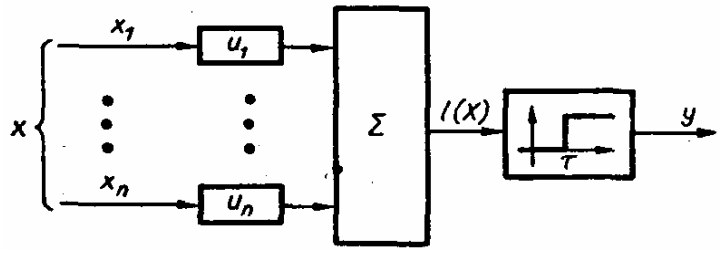


Рис. 4.3.1. Блок-схема порогового логического элемента.

ранстве возможных входов $\{X\}$. Оно представляет собой вершины n-мерного гиперкуба, который гиперплоскостью

$$\sum_{i=1}^{n} u_i x_i = \tau \quad (4.3.6)$$

разделяется на две части. Вершинам одной из них в соответствии с (4.3.3) присваивается «0», а вершинам других — «1».

Очевидно, что произвольную логическую функцию, т. е. произвольное присвоение вершинам 0 или 1, так образовать невозможно. Именно это обстоятельство ограничивает применение ПЛЭ такого рода. Произвольную логическую функцию можно реализовать лишь с помощью сети из ПЛЭ [254].

4.3.2. Многопороговый логический элемент

Под многопороговым логическим элементом (МПЛЭ) понимается $(n\times 1)$-полюсник с n входами x_1, ..., x_n и одним выходом y, определяемыми следующим соотношением:

$$y = f(X) = \begin{cases} \bar{\alpha}, & \text{если } \tau_{2j-1} < l(X) < \tau_{2j}; \\ \alpha & \text{в противном случае,} \end{cases} \quad (4.3.7)$$

где $\alpha \in \{0; 1\}$, а функция $l(X)$ определена выражением (4.3.4). Как видно, МПЛЭ отличается от ПЛЭ пороговым элементом ПЭ (см. рис. 4.3.1), который в данном случае становится многопороговым элементом (МПЭ), реализующим функцию (4.3.7). На рис. 4.3.2 приведены два варианта МПЭ при различных значениях α.

Таким образом, МПЛЭ определяется тройкой

$$\langle U, T, \alpha \rangle \quad (4.3.8)$$

где U — вектор весов (4.3.2), определяющий вход МПЭ (4.3.4), T — вектор порогов:

$$T = (\tau_1, ..., \tau_k) \quad (4.3.9)$$

и α — двоичный параметр. Тройка (4.3.8) определяет k-пороговый логический элемент.

МПЛЭ имеет наглядную геометрическую интерпретацию в пространстве его

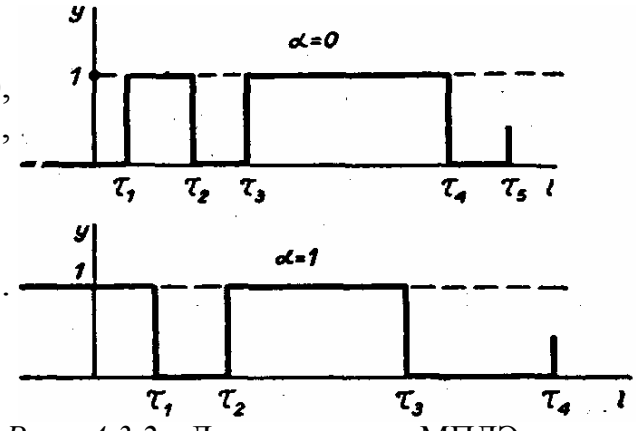


Рис. 4.3.2. Два варианта МПЛЭ, отличающиеся значениями параметра а и порогов.

входов $\{X\}$. Гиперкуб $0 \le x_i \le 1$ ($i=1,...,n$) рассекается параллельными гиперплоскостями

$$\sum_{i=1}^{n} u_i x_i = \tau_j \quad (j=1,\ldots,k) \qquad (4.3.10)$$

на $k+1$ «слоев». Вершины этого гиперкуба, попавшие в один из «слоев», принимают одно и то же значение. В двух соседних «слоях» вершины имеют противоположное значение. Веса вектора U определяют наклон этих гиперплоскостей.

Задача синтеза МПЛЭ для произвольной логической функции сводится к определению тройки (4.3.8). В простейшем случае, когда нет ограничений на число k порогов, любую логическую функцию можно реализовать с помощью так называемой канонической реализации [242]:

$$U = (2^{i_1-1},\ 2^{i_2-1},\ \ldots,\ 2^{i_n-1}), \qquad (4.3.11)$$

где $i_p \in \{1,\ldots,n\}$ и $i_p \neq i_q$ при $p \neq q$. В этом случае легко показать, что значения $l(X_j)$ не могут совпадать для любых j, т. е.

$$l(X_j) \neq l(X_i), \text{ если } j \neq i, \qquad (4.3.12)$$

и число порогов k такого МПЛЭ может изменяться в пределах

$$0 \le k \le 2^n - 1. \qquad (4.3.13)$$

(Случай $k = 0$ вырожденный, соответствующий $f(X)$=const, т. е. постоянной.)

Очевидно, что такой большой диапазон (4.3.13) возможного числа порогов при канонической реализации весов (4.3.11) является существенным недостатком, так как именно пороги определяют сложность технической реализации МПЛЭ. Чем меньше порогов, тем проще и дешевле элемент.

Поэтому следует изыскать такой метод синтеза МПЛЭ, который минимизировал бы число порогов k. Легко заметить, что количество порогов однозначно определяется весами U, т. е. $k = k(U)$ (параметр а при этом однозначно определится вектором порогов U и заданной логической функцией $F(X)$, которую следует реализовать с помощью МПЛЭ).

Заметим, что имеется много задач, где для нормального функционирования вполне достаточно задать несколько точек $N \ll 2^n$. Именно эта ситуация выгодно отличает процесс синтеза МПЛЭ от стандартных методов синтеза конечных автоматов.

Рассмотрим процесс синтеза МПЛЭ, т. е. определение вектора весов U по функции $F(X)$, заданной в точках $X_1,...,X_N$.

1 . Определяются $l_i = l(X_i)$ $(i=1,\ldots, N)$ и проверяется условие (4.3.12). Если оно нарушается, то вектор U следует изменить и повторить п. 1.

2. Производится ранжирование чисел li в ряд $li_1 < ... < li_N$

3. Определяется число изменений в ряде значений функции
$F(Xi_1)$, $F(Xi_2)$, ..., $F(Xi_N)$. Это и есть число порогов МПЛЭ. Сами значения порогов определяются из условия $li_j < \tau_l < li_{j+1}$, где i_j вычисляется из условия $F(X_i) \neq F(Xi_{j+1})$, а параметр

В процессе синтеза МПЛЭ следует решить задачу:

$$k(U) \to \min_{U \in S} \Rightarrow U^*, \tag{4.3.14}$$

где S — допустимые значения весов U, удовлетворяющих заданной булевой функции $F(X)$, т. е.

$$S : f(X) = F(X), \tag{4.3.15}$$

где $f(X)$ — логическая функция МПЛЭ (4.3.7). Решение задачи (4.3.14) в виде U^* и определяет МПЛЭ с минимальным числом порогов

$$k_{\min} = k(U^*) = \min_{U \in S} k(U). \tag{4.3.16}$$

МПЛЭ с минимальным числом порогов естественно назвать оптимальным МПЛЭ. Перейдем к синтезу таких МПЛЭ.

4.3.3. Анализ задачи синтеза оптимальных многопороговых логических элементов

Естественно задачу оптимального синтеза (4.3.14) решать поисковым методом. Исследуем ее с этих позиций.

Прежде всего рассмотрим характер поведения минимизируемой функции $k(U)$. Она кусочно-постоянна, так как k — целое число. Далее:

$$k(U) = k(cU) \tag{4.3.17}$$

Для любого c. При этом пороги изменяются очевидным образом:

$$T \to cT. \tag{4.3.18}$$

Поэтому можно рассматривать не все пространство весов, а лишь поверхность единичной гиперсферы:

$$| U | = 1 . \tag{4.3.19}$$

Необходимое условие (4.3.12) также накладывает на множество допустимых весов U ограничение, которое сводится к выполнению довольно очевидного требования линейной независимости:

$$u_i \neq \sum_{j \neq i}^{n} a_j u_j \quad (i = 1, \ldots, n), \tag{4.3.20}$$

где $a_j \in \{-1; 0; 1\}$. Это означает, что веса могут быть лишь слабо линейно-зависимыми. («Слабость» здесь выражается в указанных ограничениях на коэффициенты линейной зависимости.)

Рассмотрим, как изменяется функция $k(U)$ при незначительной случайной вариации U. Элементарный анализ показывает, что вероятность ее изменения — скачка не более чем на 4 — значительно превышает вероятности других изменений. Это означает, что существует корреляция между числом порогов k и расположением U на поверхности гиперсферы $|U| = 1$.

Для проверки был поставлен следующий эксперимент [25]. Рассматривалась логическая функция пяти (k=5) переменных:

$$F(X) = x_1x_2 \vee x_2x_3 \vee x_2x_5 \vee x_1x_3x_4 \vee x_1x_3x_5, \tag{4.3.21}$$

исходная каноническая реализация которой (4.3.11) имеет $k = 15$ порогов. Априори было известно, что эта функция реализуется однопороговым (k=1) элементом (4.3.8):

$$\langle (3, 5, 3, 1, 2); 6; 0 \rangle, \tag{4.3.22}$$

которому на гиперсфере $|U| = 1$ соответствует точка

$$U^* = \left(\frac{\sqrt{3}}{4}, \frac{5}{4\sqrt{3}}, \frac{\sqrt{3}}{4}, \frac{1}{4\sqrt{3}}, \frac{1}{2\sqrt{3}} \right). \tag{4.3.23}$$

Эксперимент состоял в следующем. На участке поверхности гиперсферы $|U| = 1$, лежащем в первом гипероктанте, выбирались случайные точки U_ξ в соответствии с равномерным законом распределения по поверхности гиперсферы. Затем строилась реализация МПЛЭ, т. е. определялось число порогов k по формуле (4.3.7), где $\alpha = F(0, 0, ..., 0) = 0$.

Расстояние ρ между точками U^* (4.3.23) и U_ξ для простоты находилось по хорде:

$$\rho = 2 \sin \frac{\varphi}{2}, \tag{4.3.24}$$

Здесь φ — угол между векторами:

$$\varphi = \arccos \langle U^*, U_\xi \rangle, \quad (4.3.25)$$

где угловыми скобками обозначена операция скалярного произведения векторов:

$$\langle A, B \rangle = \sum_{i=1}^{n} a_i b_i. \quad (4.3.26)$$

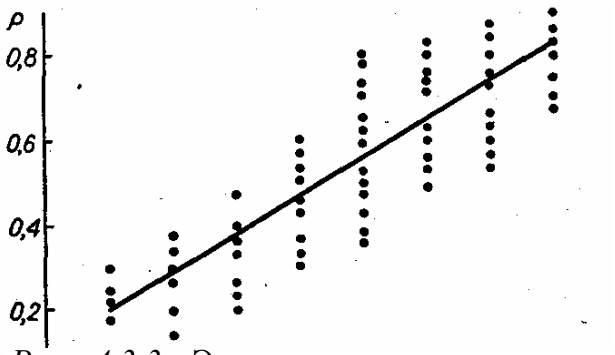


Рис. 4.3.3. Экспериментальная зависимость между расстоянием ρ и числом порогов *k*.

Результаты эксперимента в виде зависимости расстояния от числа порогов показаны на рис. 4.3.3. Отчетливо видна зависимость между этими величинами. Интересно, что уже на 117-м шаге такого «слепого» случайного поиска была найдена однопороговая реализация МПЛЭ [25].

Таким образом, можно считать, что при переходе от одной точки сферы к ближайшей другой с достаточно большой вероятностью не происходит значительного изменения числа порогов МПЛЭ. Это означает, что синтез оптимального МПЛЭ можно производить адаптивными параметрическими методами типа случайного поиска (см. § 4.1).

4.3.4. Индексные зоны

Пусть $u_1, \ldots, u_n$ — произвольные веса, удовлетворяющие условию (4.3.20); $\{X\}$ — множество всех двоичных векторов $X_j = (x_1^j, \ldots, x_n^j)$, количество которых равно $N=2^n$ ($X_1 = (0, 0, \ldots)$, $X_2 = (1, 0, \ldots, 0)$, $\ldots$, $X_N = (1, 1, \ldots, 1)$). Определим значения $l_j = \sum_{i=1}^{n} u_i x_i^j$ ($j = 1, \ldots, 2^n$) и упорядочим их по возрастанию:

$$l_{j_1} < l_{j_2} < \ldots < l_{j_2 n}, \quad (4.3.27)$$

где $j_p \in \{1, \ldots, 2^n\}$ и $j_p \neq j_q$ при $p \neq q$. Индексом точки U назовем 2^n-разрядный символ

$$Z(U) = (j_1, \ldots, j_{2n}). \quad (4.3.28)$$

Множество точек U, имеющих одинаковые индексы $Z(U)$, назовем индексной зоной [26]. Эти зоны примечательны тем, что внутри каждой из них любой МПЛЭ не изменяет, числа порогов.

Действительно, число порогов любого МПЛЭ равно числу изменений ряда:

$$F(X_{j_1}), \ldots, F(X_{j_{2^n}}), \quad (4.3.29)$$

которые могут происходить лишь при, переходе из одной индексной зоны в другую. Поэтому области пространства весов U с одним и тем же числом порогов k для заданной логической функции $F(X)$ образуются из индексных зон, целиком входящих в ту или иную область.

Легко показать, что индексные зоны выпуклы [26].

Для иллюстрации построим индексные зоны в первом октанте для n=3. Пусть $U = (u_1, u_2, u_3)$ — произвольная точка ($u_i \geq 0$, i = 1, 2, 3) в индексной зоне точки U = (1, 2, 4), являющейся канонической реализацией (4.3.11). Определим эту зону.

Ряд (4.3.27) в этом случае принимает вид

$$0 < u_1 < u_2 < u_1 + u_2 < u_1 + u_3 < u_2 + u_3 < u_1 + u_2 + u_3. \quad (4.3.30)$$

Исключая из системы неравенств (4.3.30) тривиальные, получаем

$$\begin{aligned} &0 < u_1 < u_2; \\ &u_1 + u_2 < u_1. \end{aligned} \quad (4.3.31)$$

Отсюда находим индекс искомой зоны, соответствующей канонической форме:

Z (l, 2, 4) = (1, 2, 3, 4, 5, 6, 7, 8).

Построим эту зону в сферических координатах (φ, ψ):

$$\begin{aligned} u_1 &= \rho \cos\varphi \sin\psi; \\ u_2 &= \rho \sin\varphi \cos\psi; \\ u_3 &= \rho \cos\psi. \end{aligned} \quad (4.3.32)$$

Система неравенств (4.3.31) определяет первую зону:

$$\text{«1»:} \quad \begin{cases} 0 < \cos\varphi < \sin\varphi; \\ \sin\varphi + \cos\varphi < \operatorname{ctg}\psi. \end{cases} \quad (4.3.33)$$

Аналогично можно построить индексы остальных 11 зон: Z_2= (1, 3, 2, 4, 5, 7, 6, 8); Z_3 = (1, 3, 2, 5, 4, 7, 6, 8); Z_4 = (1, 3, 5, 2, 7, 4, 6, 8); Z_5 = (1, 3, 5, 7, 2, 4, 6, 8); Z_6 = (1, 5, 3, 7, 2, 6, 4, 8);

$Z_7 =$ (1, 5, 3, 2, 7, 6, 4, 8);
$Z_8 =$ (1, 5, 2, 3, 6, 7, 4, 8); (4.3.34)
$Z_9 =$ (1, 5, 2, 6, 3, 7, 4, 8);
$Z_{10} =$ (l, 2, 5, 6, 3, 4, 7, 8);
$Z_{11} =$ (1, 2, 5, 3, 6, 4, 7, 8);
$Z_{12} =$ (l, 2, 3, 5, 4, 6, 7, 8)

и неравенства, определяющие эти зоны. На рис. 4.3.4 показаны зоны на плоскости углов φ и ψ. Номера зон соответствуют (4.3.34).

Пусть теперь задана булева функция

(4.3.35)

$$F(X) = x_1 \vee x_2 x_3,$$

для которой легко определить число порогов k_i в i-зоне:

$k_4 = k_5 = k_6 = k_7 = 1;$
$k_2 = k_3 = k_8 = k_9 = k_{11} = k_{12} = 3;$ (4.3.36)
$k_1 = k_{10} = 5.$

В результате получаем рельеф функции $k(\varphi, \psi)$, изображенный на рис. 4.3.5. Хорошо видно, что скачки функции не превышают 2, т. е. минимального изменения значения функции числа порогов.

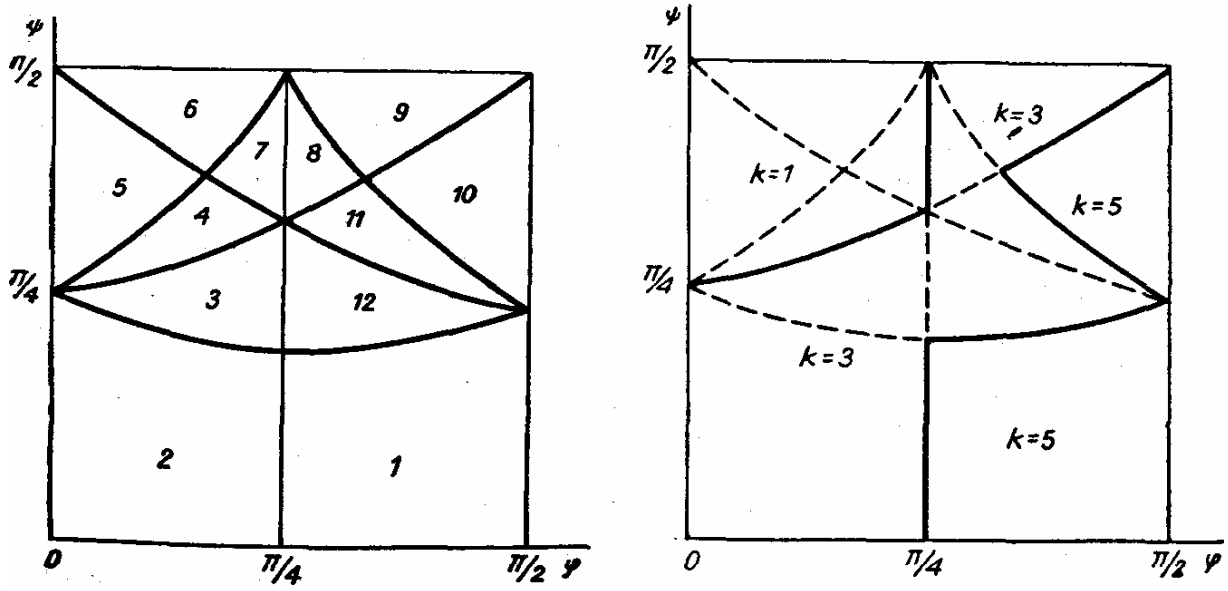


Рис. 4.3.4. Индексные зоны для положительного квадранта (n=3).

Рис. 4.3.5. Рельеф функции числа порогов для примера (4.3.35), Пунктиром показаны границы индексных зон.

Таким образом, индексные зоны в задаче синтеза оптимального МПЛЭ являются теми «квантами», из которых состоят области равного уровня функции числа порогов $k(U)$.

4.3.5. Экспериментальный синтез многопороговых логических элементов

Поиск оптимальной точки осуществляется в n-мерном параллелепипеде, содержащем начало координат:

$$a_j \le u_j \le b_j \quad (j = 1, ..., n). \tag{4.3.37}$$

Компоненты начальной точки поиска U_0 вычислялись по формуле

$$u_j^0 = a_j + \xi_j (b_j - a_j). \tag{4.3.38}$$

Компоненты последующих (текущих) точек поиска U определялись следующим образом:

$$u_j = u^*_j + \mu \left(\xi_i - \frac{1}{2} \right), \tag{4.3.39}$$

где u^*_j — компоненты оптимальной точки U^*, полученной на предыдущих шагах поиска (на первом шаге оптимальной считается точка U_0); μ=const — шаг поиска; $\xi_j \in [0; 1]$ — случайные числа с равномерным законом распределения.

Значение критерия качества $k(U)$ в текущей точке U сравнивается со значением $k^* = k(U^*)$. Точка U считается удачной, если $k<k^*$, т. е. полагается $U^*=U$, и дальнейший поиск идет от этой последней точки, что соответствует случайному поиску по наилучшей пробе.

Критерий окончания поиска связан с тем, что в течение m шагов подряд не найдена новая удачная точка.

В процессе поиска возможен выход за пределы области поиска. Во избежание этого достаточно, например, полагать $u_i = a_i$ ($u_i = b_i$), если имел место выход j-й компоненты за левую (правую) границу интервала $[a_i, b_i]$.

Описанный алгоритм поиска был экспериментально исследован на ЭВМ. Составлена программа поиска оптимальных независимых реализаций булевых функций с числом переменных $n \le 5$.

В частности, был осуществлен поиск оптимальных реализаций рассмотренной выше функции пяти переменных (4.3.21), который проводился в параллелепипеде:

$$0 \le u_j \le 10 \quad (j = 1,..., 5). \tag{4.3.40}$$

Параметры алгоритма: шаг поиска μ = 2, параметр критерия останова $m = 30$.

В процессе поиска (из разных начальных точек U_0) было получено несколько оптимальных однопороговых реализаций, причем среднее число шагов поиска равнялось 15—20. Для сравнения заметим, что при случайном сканировании (см. (4.3.3)) оптимальная реализация была найдена лишь на 117-м шаге.

Одна из найденных точек U^* имела следующие компоненты: $u^*_1 = 1{,}9$; $u^*_2 = 6$; $u^*_3 = 4{,}2$; $u^*_4 = 1{,}2$; $u^*_5 = 1{,}36$. Соответствующая оптимальная зона определяется, согласно формуле (4.3.27), неравенствами:

$$
\begin{aligned}
&u_1 < u_4 + u_5;\\
&0 < u_4 < u_5 < u_1;\\
&u_3 + u_5 < u_2 < u_1 + u_3;\\
&u_1 + u_3 + u_4 < u_2 + u_5;\\
&u_1 + u_5 < u_3 < u_1 + u_4 + u_5 < u_2 + u_3.
\end{aligned}
\qquad (4.3.41)
$$

4.3.6. Вероятностные характеристики поиска

Эффективность описанного алгоритма поиска существенно зависит от вида заданной логической функции $F(X)$, а также от параметров μ и m алгоритма поиска. Действительно, с уве личением n количество индексных зон быстро возрастает и ве роятность нахождения оптимальной (для заданной функции) зоны уменьшается. Если в подобной ситуации осуществлять поиск с большим шагом μ при малом значении m, то функция $k(U)$ ведет себя как многоэкстремальная, хотя в действитель ности она может и не обладать этим свойством. С другой сто роны, при малом шаге μ и большом m поиск становится неэффективным по затратам времени, хотя принципиально позво ляет решить задачу.

Таким образом, параметры алгоритма должны быть в опре-деленном смысле согласованы с характером функции $F(X)$. В этом отношении можно дать лишь качественные рекомендации.

Для случая функций трех переменных получены количественные оценки эффективности поиска, которые приводятся ниже.

Пусть в пространстве $\{U\}$ задана сфера с центром в начале координат. На участке сферы, лежащем в первом октанте, возьмем случайную точку U_ξ в соответствии с равномерной плотностью распределения по поверхности этого участка. Оценим вероятности попадания этой точки в индексные зоны (см. подраздел 4.3.4 настоящей книги и работу [26]).

Найдем плотности распределения углов φ и ψ, являющихся сферическими координатами точки U_ξ. Пусть плотность распределения угла $\varphi \in [0; \pi/2]$ равномерная:

$$\rho(\varphi) = \frac{2}{\pi}. \quad (4.3.42)$$

Тогда плотность распределения угла $\psi \in [0; \pi/2]$, согласно [161],

$$\rho(\psi) = \sin\psi. \quad (4.3.43)$$

Поскольку φ и ψ — независимые случайные величины, то

$$\rho(\varphi, \psi) = \rho(\varphi)\,\rho(\psi). \quad (4.3.44)$$

Следовательно, вероятность попадания точки U в индексную зону Z_i ($i = 1, 2, ..., 12$) равна

$$P_i = \frac{2}{\pi} \int_{\varphi_1}^{\varphi_2} \int_{\psi_1(\varphi)}^{\psi_1(\varphi)} \sin\psi\, d\psi\, d\varphi, \quad (4.3.45)$$

где φ_1, φ_2, $\psi_1(\varphi)$ и $\psi_2(\varphi)$ определяются границами этой зоны. Вычислим, например, вероятность P_1 попадания в первую индексную зону (см. рис. 4.3.4). Пределы интегрирования:

$$\varphi_1 = 0;\ \varphi_2 = \pi/2;\ \psi_1(\varphi) = 0;\ \psi_2(\varphi) = \operatorname{arc\,ctg}(\sin\varphi + \cos\varphi). \quad (4.3.46)$$

Получаем

$$P_1 = \frac{1}{\pi} \operatorname{arc\,sin} \frac{1}{3} \approx 0{,}108. \quad (4.3.47)$$

Аналогично можно найти остальные вероятности:

$$\begin{aligned} P_2 &= P_{11} \approx 0{,}062; \\ P_3 &= P_{10} \approx 0{,}060; \\ P_4 &= P_9 \approx 0{,}100; \\ P_5 &= P_8 \approx 0{,}116; \\ P_6 &= P_7 \approx 0{,}054. \end{aligned} \quad (4.3.48)$$

Для рассмотренной выше функции $F(X)$ (4.3.35) вероятность случайного попадания в оптимальную область равна

$$P_3 + P_4 + P_5 + P_6 \approx 0{,}33, \quad (4.3.49)$$

т. е. потери на поиск достаточно малы (в среднем каждый третий случайный шаг поиска оказывается удачным).

4.3.7. Синтез надёжного многопорогового логического элемента

При проектировании реальных МПЛЭ необходимо учитывать нестабильность их физических характеристик и сигналов, которые влияют на правильность вычисления логической функции.

Следуя работе [77], будем считать, что эта нестабильность сводится к случайным отклонениям весов входов и порогов относительно их номинальных значений, а модули отклонений пропорциональны соответствующим номинальным значениям.

Пусть задан однопороговый ПЛЭ, реализующий некоторую булеву функцию с пороговым интервалом (l_α, l_β):

$$l_\alpha = \max\,(l(X) \mid f(X) = \alpha); \qquad (4.3.50)$$

$$l_\beta = \min\,(l(X) \mid f(X) = \bar{\alpha}).$$

Обозначим через 6 относительное отклонение параметров данного элемента. Тогда условия надежной работы МПЛЭ (с учетом указанных выше функций) будут следующими (рис. 4.3.6):

$$l'_\alpha \le \tau;$$
$$\tau'' \le l_\beta, \qquad (4.3.51)$$

где

$$l'_\alpha = l_\alpha\,(1 + \delta);$$
$$l'_\beta = l_\beta\,(1 - \delta); \qquad (4.3.52)$$
$$\tau' = \tau\,(1 - \delta);$$
$$\tau'' = \tau\,(1 + \delta).$$

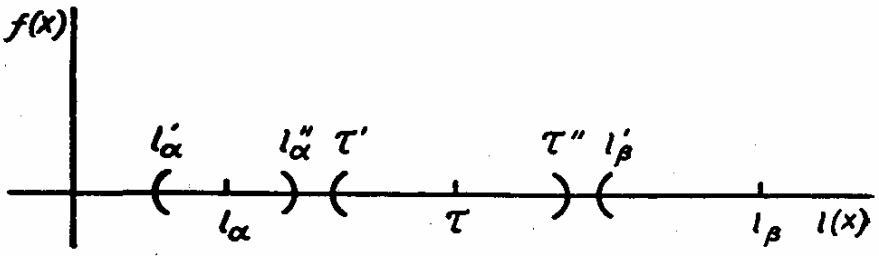


Рис. 4.3.6. Пороговый интервал однопорогового логического элемента.

Из (4.3.51), переходя к равенствам, легко получить выражение для максимально допустимого отклонения δ^*, которое является характеристикой надежности данного ПЛЭ:

$$\delta^* = \frac{1-a}{1+a}, \quad (4.3.53)$$

где $a = \left(\frac{l_\alpha}{l_\beta}\right)^{1/2}$. Величина порога τ определяется из выражения:

$$\tau = \sqrt{l_\alpha l_\beta}. \quad (4.3.54)$$

Таким образом, данный ПЛЭ правильно вычисляет заданную булеву функцию, если выполнено условие

$$\delta \leq \delta^*. \quad (4.3.55)$$

В случае МПЛЭ ‹*U, T,* α› имеем:

$$\delta^* = \min \{\delta_1^*, \ldots, \delta_k^*\},$$

где δ^*_i (i = 1, ..., k) — характеристики надежности соответствующих пороговых интервалов, вычисленные по формуле (4.3.53). Величина δ^* соответствует минимально допустимому отклонению параметров данного МПЛЭ от их номинальных значений, при котором этот МПЛЭ надежно функционирует. Естественно, что МПЛЭ максимальной надежности должен иметь минимально возможное (для заданной булевой функции) число порогов, поскольку диапазон изменения взвешенного сигнала $l(X)$ ограничен (действительно, из условия $|U|$=1 следует, что $\sum_{j=1}^{n} |u_j| < \sqrt{n}$). Эксперименты на ЭВМ подтвердили это соображение. На рис. 4.3.7 показана характерная экспериментальная зависимость величины δ для МПЛЭ с разным числом порогов, реализующего одну и ту же функцию.

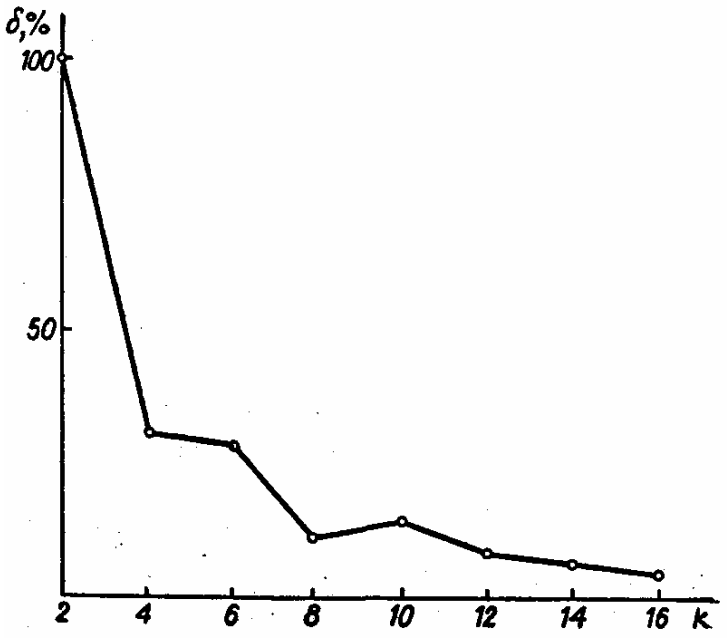
Отсюда следует трудность технической реализации МПЛЭ с большим числом порогов ввиду жестких требований, предъявляемых к его компонентам. Это обстоятельство заставляет реализовать сложные функции сетью из МПЛЭ.

Здесь следует отметить, что для максимизации надежности МПЛЭ можно незначительно варьировать значения порогов T. Для этого, однако, нужно знать статистические свойства вариаций весов δU, и сигналов δX.

Пусть $P_U(\delta U)$ — плотность распределения случайных отклонений

весов U, а $p_X(\delta X)$ — плотность распределения случайных

Рис. 4.3.7. Зависимость характеристики надежности МПЛЭ от числа порогов.



отклонений входных сигналов X. Обозначим буквой P вероятность ошибки при реализации всех входных сигналов $\{X\}$:

$$P = \underset{\{X\}}{\text{Вер}} \,(f(X, U, T) \neq F(X)), \tag{4.3.56}$$

где $F(X)$ — заданная логическая функция, а $f(X,U,T)$ — функция, реализованная данным МПЛЭ с весами U и порогами T.

Вероятность P зависит от порогов T, и ее можно вычислить с помощью интеграла:

$$P(T) = \sum_{X_i \in \{X\}} \iint [f(X_i + \delta X, U + \delta U, T) - F(X_i)]^2 \times$$

$$\times p_X(\delta X)\, p_U(\delta U)\, d\delta X d\delta U. \tag{4.3.57}$$

Аналитическая оценка этого интеграла, очевидно, будет представлять большие трудности. Поэтому удобно воспользоваться методом Монте-Карло:

$$\hat{P}(T) = \frac{1}{N} \sum_{X_i \in \{X\}} \sum_{j=1}^{N} [f(X_i + \delta X_j, U + \delta U_j, T) - F(X_i)]^2, \tag{4.3.58}$$

где N — база оценки, а δX_j и δU_j — реализация случайных отклонений сигналов и весов в соответствии с заданными плотностями распределений $p_X(\delta X)$ и $p_U(\delta U)$.

Теперь задача синтеза надежного МПЛЭ формулируется следующим образом:

$$\hat{P}(T) \to \min_{T} \Rightarrow T^*. \qquad (4.3.59)$$

Как видно, это задача оптимизации в обстановке случай-дых помех, генерируемых монте-карловской оценкой. Как отмечено в § 3.2, дисперсия этой помехи уменьшается с ростом базы оценки N как $1/N$, что позволяет управлять уровнем помехи в процессе оптимизации.

Решать эту k-мерную оптимизационную задачу следует одяим из методов параметрической адаптации, рассмотренных в § 4.1, что дает возможность определить оптимальные пороги T^*.

Таким образом, применение методов параметрической адаптации позволяет синтезировать оптимальные МПЛЭ по различным критериям: числу порогов (4.3.14) и надежности (4.3.59). В обоих случаях случайный поиск является эффективным инструментом синтеза: минимизируемые функции всегда достаточно сложны, что создает наиболее благоприятную ситуацию для использования случайного поиска.

4.3.8. Многозначные многопороговые логические элементы*

Многозначные логические элементы отличаются более чем двумя выходными уровнями, т. е. являются небулевыми элементами. На рис. 4.3.8 показана пятиэлементная модель многозначного МПЛЭ с функционально неполными операторами l и F_m (смысл их будет раскрыт ниже) и многозначным структурным алфавитом [22]. Эта модель имеет:

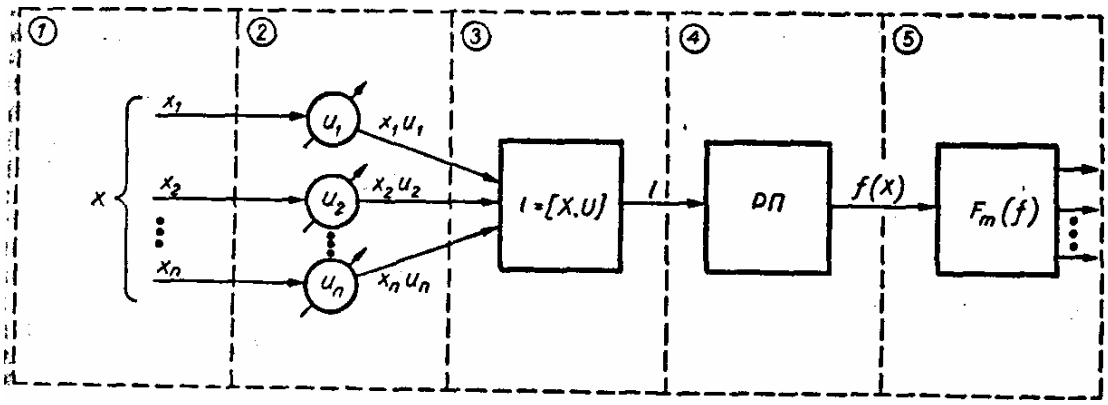


Рис. 4.3.8. Блок-схема многозначного МПЛЭ.

* Этот подраздел по просьбе автора написан А. Т. Бахаревым.

1. Входной элемент из n входов, вообще говоря, разной значности:

$$x_i \in \{0, 1, \ldots, q_i - 1\} \quad (i = \overline{1, n}), \tag{4.3.60}$$

образующих входной вектор

$$X = (x1, \ldots, x_n). \tag{4.3.61}$$

2. Элемент усиления входных сигналов x_i с коэффициентами u_i, образующими вектор весов входов

$$U = (u_1, \ldots, u_n), \tag{4.3.62}$$

где в общем случае u_i — действительные числа.

3. Элемент формирования входной композиции — свертки

$$l = l(X) = \sum_{i=1}^{n} u_i x_i. \tag{4.3.63}$$

4. Элемент решающего правила характеризуется настраиваемым вектором порогов

$$T = (\tau_1, \ldots, \tau_k). \tag{4.3.64}$$

Здесь $\tau_1 < \ldots < \tau_k$. Этот вектор реализует логическое преобразование

$$l \to f = f(l) \tag{4.3.65}$$

в виде

$$f(l) = \begin{cases} \alpha_1, & \text{если } l < \tau_1; \\ \alpha_2, & \text{если } \tau_1 \leqslant l \leqslant \tau_2; \\ \ldots\ldots\ldots\ldots\ldots \\ \alpha_{k+1}, & \text{если } \tau_k \leqslant l, \end{cases} \tag{4.3.66}$$

где

$$\alpha_j \in \{0, 1, \ldots, q_f - 1\}. \tag{4.3.67}$$

На рис. 4.3.9 показан характерный вид функции f.

5. Выходной элемент формирует настраиваемые выходные сигналы

$$F_m = F_m(f), \quad m < q_f(q_f - 1)^k, \tag{4.3.68}$$

т. е. функции F являются производными от f (задающей функции) и могут отличаться от нее лишь значениями α_j на

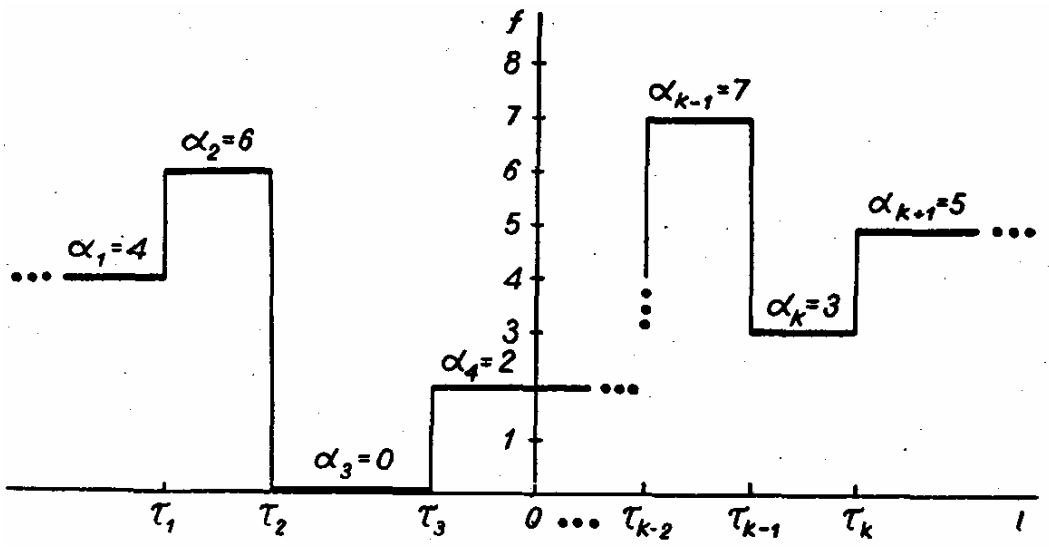


Рис. 4.3.9. Пример функции *f* многозначного МПЛЭ.

соответствующих пороговых интервалах. Эта модель и ее частные случаи рассмотрены в работах [14, 15, 22—24]. Модель является функционально полной и обладает свойствами многоустойчивых и многофункциональных моделей. Алгоритм синтеза такой модели для реализации произвольной логической функции почти полностью аналогичен рассмотренному выше для двузначных (булевых) функций, достаточно лишь в качестве компонент канонического вектора U взять следующие числа:

$$\begin{aligned} &u_{\beta_1}=1;\\ &u_{\beta_2}=u_{\beta_1}q_{\beta_1}=q_{\beta_1};\\ &\cdots\cdots\cdots\\ &u_{\beta_n}=u_{\beta_{n-1}}q_{\beta_{n-1}}=\prod_{i=1}^{n-1}q_{\beta_i}, \end{aligned} \qquad (4.3.69)$$

где $\beta_i \in \overline{\{1, n\}}$, т. е. нумерация компонент произвольна (количество всех возможных перенумераций равно $n!$). В общем случае компоненты вектора U должны удовлетворять условию линейной независимости

$$u_i \neq \sum_{j\neq i}\varepsilon_j u_j \qquad (i=\overline{1, n}), \qquad (4.3.70)$$

где $\varepsilon_j \in \{0; \pm 1, \ldots, \pm(\hat{q}-1)\}$, $\hat{q} = \max(q_1, \ldots, q_n)$. Тогда $l_{p1} \neq l_{p2}$, если $X^{(p_1)} \not\equiv X^{(p_2)}$, т. е. свертка отображает множество всех возможных векторов $X^{(S)}$ на строго различные точки на числовой оси

$$l_S\ (S=\overline{0,\ Q};\ Q=\prod_{i=1}^{n} q_i-1)$$

Основное свойство этого алгоритма заключается в том, что он позволяет сразу решить задачу синтеза логического элемента для произвольной логической функции, хотя и не обеспечивает минимальность числа порогов, т. е. минимальную размерность вектора T (4.3.64). Кроме того, процедура синтеза стандартна (не зависит от специфики реализуемой функции), проста и удобна для программирования на ЭВМ. Минимизация числа порогов здесь производится аналогично изложенному в подразделе 4.3.2. Задача оптимального синтеза многозначных МПЛЭ . формулируется, как рассмотрено выше для МПЛЭ, в виде (4.3.14).

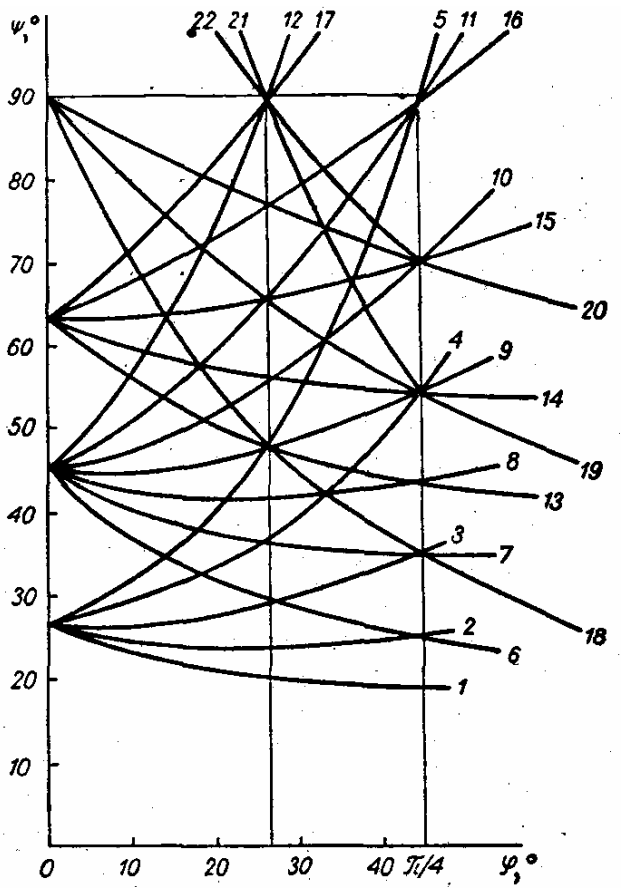


Рис. 4.3.10. Индексные зоны трехуровневого трехвходового элемента. Уравнения границ заданы формулами (4.3.71).

Проблема разбиения пространства весов входов на индексные зоны в многозначном случае значительно сложнее, нежели в двузначном: количество индексных зон с ростом n и q_i быстро возрастает (практически определить все индексные зоны для $n \geq 6$ невозможно даже на ЭВМ независимо от значности $q_i \geq 2$), a их геометрические размеры на единичной гиперсфере, естественно, уменьшаются. На рис. 4.3.10 показаны все 78 индексных зон для случая $q_1=q_2=q_3=3$ ($n=3$) в полосе $0 \leq \varphi \leq \pi/4$, $0 \leq \leq \psi \leq \pi/2$ при записи компонент вектора U в сферических координатах.

Уравнения некоторых разделяющих кривых имеют следующий вид [14]:

1. $2 (\sin \varphi + \cos \varphi) = \operatorname{ctg} \psi$;
2. $\sin \varphi + 2 \cos \varphi = \operatorname{ctg} \psi$;

. . . . . .

7. $\sin \varphi + \cos \varphi = \operatorname{ctg} \psi$;

. . . . . .

9. $\cos \varphi = \operatorname{ctg} \psi$; (4.3.71)

. . . . . .

11. $\cos \varphi - \sin \varphi = \operatorname{ctg} \psi$;

. . . . . .

19. $\sin \varphi = \operatorname{ctg} \psi$;

. . . . . .

21. $2 \sin \varphi - \cos \varphi = \operatorname{ctg} \psi$;

22. $2 \sin \varphi - \cos \varphi = 2 \operatorname{ctg} \psi$.

Отметим, что кривые *7, 9, 11* и *19* соответствуют рассмотренному ранее случаю $q_1 = q_2 = 2$, n=3 на рис. 4.3.4. В ситуации, когда с ростом размерности модели количество индексных зон быстро возрастает, возникает задача исключения полного перебора всех индексных зон с целью решения задачи оптимизации числа порогов модели. Поисковые методы вообще и метод случайного поиска в частности являются, вероятно, единственно возможным подходом к решению этой задачи. Заметим, что точное определение значения характеристики надежности δ, рассмотренной ранее, представляет большую трудность даже для моделей малой размерности и выполняется, как правило, только поисковыми методами.

Аппарат индексных зон позволяет интерпретировать специфические свойства пороговых структур — функциональную устойчивость и адаптивность (свойственные биологическим феноменам). Предположим, что на вектор U действует некоторый случайный дестабилизирующий фактор среды ΔU, изменяющий его, а флюктуации компонент вектора T несущественны либо сводятся к фактору среды. Если фактор ΔU не выводит вектор U за пределы фиксированной индексной зоны, то он .не может изменить выходную характеристику f модели, что соответствует функциональной устойчивости схемы. С другой стороны, если этот фактор выводит вектор U в другую индексную зону, то выходная характеристика логического элемента может существенно измениться.

Уменьшение функциональной устойчивости порогового элемента с ростом его размерности эквивалентно известному в пороговой логике факту малой надежности пороговых устройств с большим числом входов [77], отмеченному выше. Этим, в частности, можно объяснить многочисленные неудачи конструирования реальных пороговых устройств с большим коэффициентом надежности,

который, как правило, принципиально недостижим. С другой стороны, вероятностный подход к данной проблеме, заключающийся в учете частот ошибок на всех входных наборах и частот, с которыми эти наборы появляются, показал [125], что снижение надежности порогового элемента в резуль-

тате его усложнения (увеличения числа входов) не является неизбежным. Более того, при случайных входах надежность растет с увеличением числа входов! Это объясняется малым числом «рабочих» сверток (как и в случае слабоопределенных функций), что фактически эквивалентно снижению размерности элемента — эффект множественного объединения нескольких соседних индексных зон в одну.

В реальных условиях может допускаться функционирование порогового элемента с некоторой ошибкой σ, т. е. на σ произвольных входных наборах выходная функция может отличаться от заданной. Так возникает задача разбиения всех допустимых логических функций на непересекающиеся классы эквивалентности и определения их мощности [15]. В общем случае на каждом S-м входном наборе X^S может иметь независимую (индивидуальную!) значность $F_S \in \{0, 1, \ldots, q_{F_S}-1\}$. Тогда количество указанных классов равно Q+1, а их мощность

$$R = \binom{\sigma}{Q} \prod_{j=1}^{\sigma} (q_{F_{S_j}} - 1), \quad (4.3.72)$$

где S_j — подмножество индексов S длины σ, т. е. количество классов зависит только от числа входных наборов, а мощность каждого класса определяется еще и количеством допустимых значений выходной функции [15]. Частными случаями приведенной оценки мощности классов являются следующие:

а) если $F \in \{0, 1, \ldots, q_F-1\}$, т. е. выходная функция равно значна на любом наборе, то

$$R = \binom{\sigma}{Q} (q_F-1)^{\sigma}; \quad (4.3.73)$$

б) если $q_F=q_i=q$, т. е. выходная функция равнозначна по входам и выходу, то

$$R = \binom{\sigma}{q^n} (q-1)^{\sigma}; \quad (4.3.74)$$

в) если $q_F=q_i=q=2$, т. е. выходная функция является бу левой, то

$$R = \binom{\sigma}{2^n}. \quad (4.3.75)$$

Последняя формула показывает специфичность булева базиса (его вырожденность в классе многозначных логических функций). На основе приведенных оценок можно построить алгоритм

аппроксимации многозначных выходных функций в базисе случайных логических функций.

В заключение отметим, что аппарат индексных зон можно применить к решению актуальной проблемы классификации многозначных логических функций, сложность которой общеизвестна. Так, в работе [254, с. 36] класс троичных функций двух переменных ($q_1 = q_2 = q_F = 3$) до сих пор считается необозримым. Ранее было известно [258], что в этом классе имеется 1593 функции (из общего их числа 19683) с пороговым порядком $k \leq 2$, т. е. любая из них может быть реализована многопороговой моделью с числом порогов, не большим двух. В работе [23] полным перебором индексных зон показано, что весь этот класс функций имеет сложность $k \leq 7$, т. е. разбивается на семь подклассов.

§ 4.4. Адаптивный синтез оптимальных планов эксперимента для регрессионной модели

4.4.1. Постановка задачи

Одна из задач планирования эксперимента [137, 223] связана с нахождением параметров $c_1, \ldots, c_q$ модели, структура которой определена:

$$y = \sum_{i=1}^{q} c_i \varphi_i(X), \quad (4.4.1)$$

где $\varphi_i(X)$ ($i = 1, \ldots, q$) — заданная система линейно-независимых скалярных функций векторного аргумента:

$$X = (x_1, \ldots, x_n). \quad (4.4.2)$$

Например, для структуры (4.4.1), определяемой полной квадратичной формой, имеем систему функций $\varphi_i(X)$ вида

$$\varphi_1(X) = 1;\ \varphi_2(X) = x_1;\ \varphi_3(X) = x_2; \quad (4.4.3)$$
$$\varphi_4(X) = x_1^2;\ \varphi_5(X) = x_1^2;\ \varphi_6(X) = x_1 x_2,$$

где $q = 6$.

Задача определения параметров $C = (c_1, \ldots, c_q)$ по наблюдениям состояния объекта y_j в точках X_j ($j = 1, \ldots, N \geq n$), т. е. на основе информации

$$I = \langle X_j, y_j \ (j = 1, \ldots, N) \rangle, \quad (4.4.4)$$

сводится, как известно, к решению стандартной задачи минимизации функции суммарной невязки поведения модели (4.4.1) и объекта (4.4.4):

$$Q(C) = \sum_{j=1}^{N} \mu \left[\sum_{i=1}^{q} c_i \varphi_i(X_j) - y_j \right] \to \min_{C \in R^q}, \quad (4.4.5)$$

где $\mu(\cdot)$ — функция, обладающая естественными свойствами:

$$\mu(\cdot) \geq 0; \ \mu(z) = \mu(-z); \ \mu(0) = 0, \quad (4.4.6)$$

например,

$$\mu(\cdot) = (\cdot)^2; \quad \mu(\cdot) = |\cdot| \quad (4.4.7)$$

и т. д. В задачах планирования эксперимента принята квадратичная невязка, для которой, вычисляя частные производные и приравнивая их к нулю:

$$\frac{\partial Q(C)}{\partial c_l} = 0 \qquad (l = 1, \ldots, q), \quad (4.4.8)$$

получаем систему линейных алгебраических уравнений вида

$$\sum_{i=1}^{q} c_i \sum_{j=1}^{N} \varphi_l(X_j) \varphi_i(X_j) = \sum_{j=1}^{N} y_j \varphi_l(X_j) \qquad (l = 1, \ldots, q). \quad (4.4.9)$$

Матрицу этой системы

$$\Phi = \left\| \sum_{j=1}^{N} \varphi_l(X_j) \varphi_i(X_j) \right\|_{q \times q} \quad (4.4.10)$$

обычно называют матрицей Фишера или информационной матрицей [137].

Задача планирования эксперимента заключается в определении таких точек N экспериментов

$$\bar{X} = \{X_1, \ldots, X_N\}, \quad (4.4.11)$$

называемых планом эксперимента, чтобы их реализация (4.4.4) в объекте дала возможность наилучшим образом определить параметры модели (4.4.1).

Легко видеть, что выбор точек плана $\bar{X}$ далеко не произволен. Действительно, они могут быть расположены так, что ранг матрицы Ф станет меньше q и параметры из такого эксперимента невозможно

будет определить.

Для формирования критерия оптимальности плана будем рассматривать объекты, выход которых определяется следующим образом:

$$y_j = f(X_j) + \varepsilon_j, \qquad (4.4.12)$$

т. е. является суммой регулярной и случайной составляющих. Регулярная часть $f(X)$ зависит только от входа X объекта, а случайная ε не зависит от входа X и является независимой реализацией нормального закона распределения с нулевым математическим ожиданием и дисперсией σ^2, т. е.

$$M\varepsilon = 0; \; M\varepsilon^2 = \sigma^2, \qquad (4.4.13)$$

где M — знак математического ожидания. Значение дисперсии σ^2 может быть априори неизвестным.

Тогда результат каждого эксперимента является случайной величиной, что делает случайными и параметры C модели (4.4.1). Свойства случайного вектора C определяются его математическим ожиданием

$$MC = C^* = (c^*_1, ..., c^*_q) \qquad (4.4.14)$$

с дисперсионной матрицей:

$$DC = \| \sigma^2 \|_{q \times q}, \qquad (4.4.15)$$

где

$$\sigma_{ij} = M(c_i - c^*_i)(c_j - c^*_j) \quad (i, j = 1,..., q). \qquad (4.4.16)$$

Эта дисперсионная матрица и определяет все точностные свойства параметров. Действительно, их дисперсии расположены по диагонали матрицы и равны

$$Dc_i = \sigma_{ii}^2 \quad (i = 1, ..., q), \qquad (4.4.17)$$

а коэффициент корреляции i-го и j-го параметров модели равен σ_{ij}^2.

Очевидно, что все экстремальные требования к эксперименту, связанные с точностью определения искомых параметров, можно определить на дисперсионной матрице (4.4.15) в виде скалярной функции

$$K = K(DC), \qquad (4.4.18)$$

минимум которой соответствует выполнению требований эксперимента.

Например, функцией K может быть обобщенная дисперсия, т. е. определитель дисперсионной матрицы

$$K^{(1)} = |\, DC \,|, \qquad (4.4.19)$$

минимум которого часто интересует экспериментатора. Другой пример критерия — след дисперсионной матрицы:

$$K^{(2)} = \sum_{i=1}^{q} \sigma_{ii}^2. \qquad (4.4.201$$

Наконец, максимальная дисперсия

$$K^{(3)} = \max_{i=1,\ldots,q} \sigma_{ii}^2 \qquad (4.4.21)$$

тоже может быть примером такой функции, минимума которой добивается экспериментатор.

В общем случае вид функции (4.4.18) задается исходя из потребностей экспериментатора и может быть любым.

Связь этой функции с планом $\tilde{X}$ осуществляется через матрицу Фишера в виде [223]

$$DC = \sigma^2 \Phi^{-1}, \qquad (4.4.22>$$

т. е. дисперсионная матрица (4.4.15) с точностью до постоянной равна обратной матрице Фишера (4.4.10), что позволяет легко вычислять любой элемент дисперсионной матрицы по заданному плану $\tilde{X}$. Это означает, что план $\tilde{X}$ и система функций $\varphi_i(X)$ (i = 1, ..., q) модели (4.4.1) однозначно определяют дисперсионную матрицу, а вслед за ней и значение критерия

$$K = \psi(\tilde{X}), \qquad (4.4.23)$$

где ψ — алгоритм определения критерия по формулам (4.4.18), (4.4.22), (4.4.10) и (4.4.11). Это и позволяет назвать критерии K критерием эффективности плана $\tilde{X}$.

Теперь задача синтеза оптимального плана $\tilde{X}^*$ формулируется как задача минимизации:

$$\psi(\tilde{X}) \to \min_{\tilde{X} \in S} \Rightarrow \tilde{X}^*, \qquad (4.4.24)$$

где S — область планирования эксперимента, определяющая пределы изменения элементов плана $\tilde{X}$ в пространстве входов объекта $\{X\}$.

В зависимости от выбора критерия K полученный оптимальный план называют по-разному. При минимизации критерия

(4.4.19) получают D-оптимальный план $(\tilde{X}^*_D)$, для критерия (4.4.20) — A-оптимальный $(\tilde{X}^*_A)$ и для критерия (4.4.21) — E-оптимальный $(\tilde{X}^*_E)$. Заметим, что, согласно выражению (4.4.22), минимизация $|DC|$ эквивалентна максимизации определителя матрицы Фишера Ф. Это значительно упрощает задачу, так как отпадает необходимость в обращении матрицы. Указанным обстоятельством пользуются при синтезе D-оп-тимального плана:

$$|\Phi| \to \max_{\tilde{X} \in S} \Rightarrow \tilde{X}^*_D, \quad (4.4.25)$$

Задача (4.4.24) имеет в общем случае nN искомых переменных и является многоэкстремальной задачей нелинейного программирования. Воспользуемся для ее решения методами параметрической адаптации и, в частности, случайным поиском.

4.4.2. Последовательный синтез плана

Задача (4.4.24) обычно имеет очень большую размерность. Естественно упростить ее путем декомпозиции на более простые задачи, для чего используется прием сведения задачи (4.4.24) к последовательности задач:

$$\psi(X^*_1, \ldots, X^*_{q+z}, X) \to \min_{X \in S} \Rightarrow X^*_{k+z+1}. \quad (4.4.26)$$

Эти задачи имеют лишь n переменных, и результатом их решения является точка $(q+z+1)$-го эксперимента. Добавляя ее к предыдущему плану эксперимента (назовем его опорным) $\tilde{X}^*_{q+z} = \{X^*_1, \ldots, X^*_{q+z}\}$, получаем план $\tilde{X}^*_{q+z+1}$ и т. д. Однако исходный план $\tilde{X}^*_q$ должен быть задан или получен путем решения задачи (4.4.24) при $N = q$. Такое последовательное наращивание плана позволяет задачу оптимизации с nN переменными свести к последовательности задач с n переменными (при заданном исходном плане $\tilde{X}^*_q$).

Проанализируем задачи (4.4.26), получаемые таким последовательным синтезом [32]. Для этого построим рельеф минимизируемой функции (4.4.26) в пространстве варьируемой точки X плана

$$\tilde{X}_{q+z+1} = \{X^*_1, \ldots, X^*_{q+z}, X\}, \quad (4.4.27)$$

т. е. исследуем поведение функции

$$\psi = \psi(\tilde{X}^*_{q+z}, X). \quad (4.4.28)$$

Именно в этой ситуации решается задача минимизации

(4.4.26). .

Рассмотрим в качестве критерия оптимизации плана D-критерии (4.4.19), т. е. будем решать задачу максимизации определителя информационной матрицы (4.4.25):

$$\psi(\bar{X}^*_{q+z}, X) \to \max_{x \in S}, \quad (4.4.29)$$

где ψ в данном случае есть алгоритм вычисления определителя матрицы Фишера на плане (4.4.27).

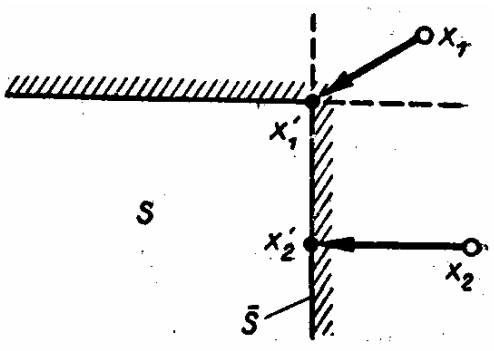


Рис. 4.4.1. Работа процедуры ортогонального проецирования для прямоугольной области планирования.

Область планирования S выберем простейшую — квадрат ($n = 2$):

$$S: \begin{cases} -1 \leqslant x_1 \leqslant 1; \\ -1 \leqslant x_2 \leqslant 1. \end{cases} \quad (4.4.30)$$

. Для решения задачи (4.4.29) естественно использовать локальный случайный поиск с оператором проецирования точек на область поиска при $X \notin S$. В этом случае применялась процедура ортогонального проецирования на границу $\bar{S}$ области S:

$$\mathrm{пр}_S(X \notin S) = X' \in \bar{S}, \quad (4.4.31)$$

работа которой проиллюстрирована на рис. 4.4.1.

Алгоритм поиска имел следующий вид (напомним, что рассматривалась задача максимизации):

$$X_N = \left\{ \begin{array}{l} X_{N-1} \text{ при } \Delta Q'_N \leqslant 0 \\ X_{N-1} + a_N \xi_N \text{ при } \Delta Q'_N > 0 \end{array} \right\} \text{при } X_{N-1} + a_N \xi_N \in S;$$
$$\mathrm{пр}_S(X_{N-1} + a_N \xi_N) \text{ при } (\Delta Q''_N > 0) \wedge (X_{N-1} + a_N \xi_N \notin S), \quad (4.4.32)$$

где

$$\Delta Q'_N = Q(X_{N-1} + a_N \xi_N) - Q(X_{N-1});$$
$$\Delta Q''_N = Q(\mathrm{пр}_S(X_{N-1} + a_N \xi_N)) - Q(X_{N-1}); \quad (4.4.33)$$

ξ_N — N-я реализация нормального случайного вектора с нулевым математическим ожиданием и единичной дисперсией по всем направлениям $\{X\}$.

Параметр величины случайного шага a_N адаптировался стандартным образом (см. § 3.5):

$$a_N = \begin{cases} \gamma_1 a_{N-1} & \text{при } \Delta Q_N > 0; \\ \gamma_2 a_{N-1} & \text{при } \Delta Q_N \leqslant 0, \end{cases} \quad (4.4.34).$$

где ΔQ вычислялась по одной из формул (4.4.33) в зависимости от сложившейся ситуации, параметр γ_2=1,2, а γ_1 был выбран в соответствии с указанными в § 3.5 рекомендациями. Рассмотрим сначала линии равного уровня максимизируемого критерия (4.4.29) для квадратичной формы (q = 5) вида

$$y = c_1 + c_2x_1 + c_3x_3 + c_4x_1^2 + c_5x_2^2 \qquad (4.4.35)$$

лри различном числе точек опорного плана $\tilde{X}^*_{q+z}$.

На рис. 4.4.2 изображены линии равного уровня значения определителя матрицы Фишера при z=3. Здесь точки опорного плана $\tilde{X}^*_{5+3}$ обозначены крестиками. Хорошо видно, что задача имеет многоэкстремальный характер. Локальные экстремумы расположены в точках i = 2, 4, 5, 6, 7, 8, а глобальный экстремум — в точке $X^{(2)}$. Здесь же показаны четыре траектории случайного поиска (4.4.32) со случайными начальными точками. Две из этих траекторий сошлись к глобальному экстремуму $X^{(2)}$.

Рельеф для той же модели (4.4.35), но при другом опорном плане (i = 2, 3, 4, 5, 6, 7, 8, 9) показан на рис. 4.4.3 (кружочки

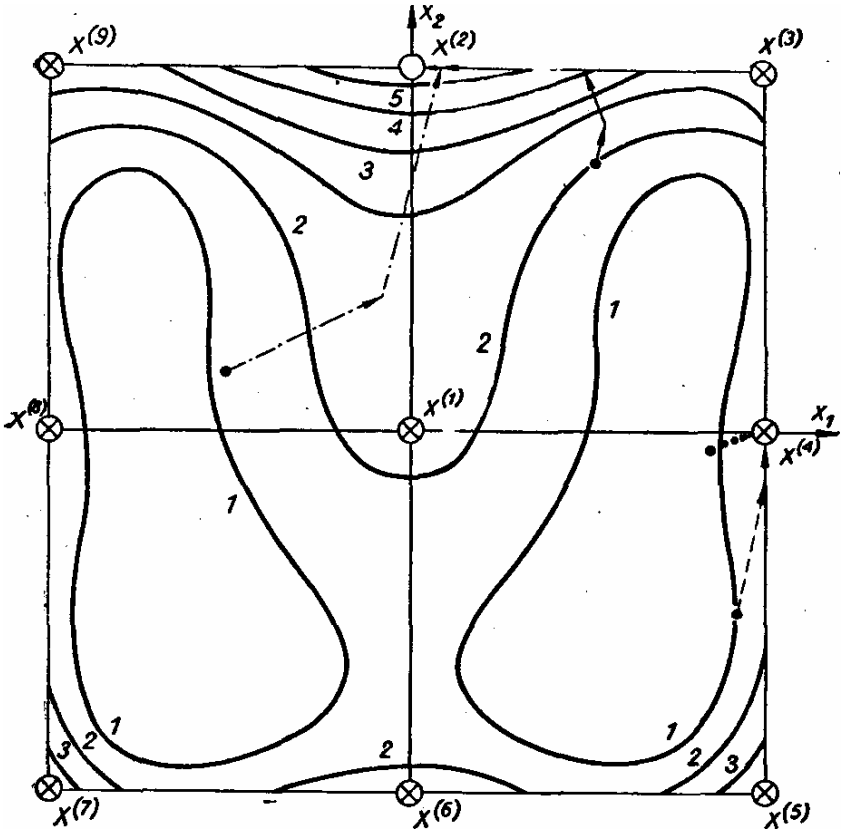


Рис. 4.4.2. Линии равного уровня определителя информационной матрицы для полинома (4.4.35), соответствующие следующим значениям: *1* — 3450, *2* — 3750, *3* — 4050, *4* — 4400, 5 — 4700. Максимум (5000) в точке $X^{(2)}$.

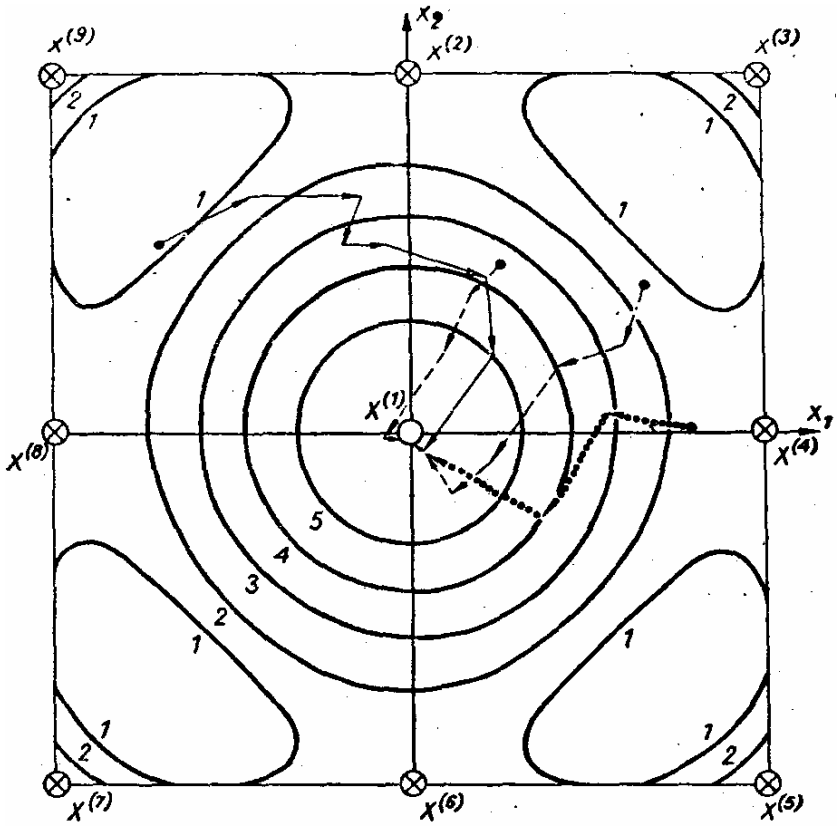


Рис. 4.4.3. Линии равного уровня при вакантной центральной точке $X^{(1)}$. Значения критерия на уровнях: *1* — 3700, *2* — 4000, *3* — 4300, *4* — 4600, *5* — 4900.

с крестиками). Симметрия плана обусловила и симметрию рельефа. Как видно, задача многоэкстремальна и имеет локальные экстремумы в точках i = 1, *3,* 5, 7, 9. Глобальный экстремум $X^{(1)}$ был достигнут при всех четырех реализациях поиска из случайных начальных точек (см. ломаные линии на рис. 4.4.3).

На рис. 4.4.4. представлены линии равного уровня максимизируемого критерия с той же моделью и опорным планом i = = 1, 2, 3, 4, 6, 7, 8, 9, точки которого обозначены крестиками. Локальные экстремумы этой многоэкстремальной задачи расположены в точках с номерами t = 2, 3, 5, 7, 8, 9 и точке *A,* а глобальный поиск (4.4.32) дважды привел к точке *A,* один раз к $X^{(7)}$ и один — в глобальный экстремум $X^{(5)}$.

Теперь рассмотрим случай шеститочечного опорного плана N = 5 + 1 (z = 1) для той же модели (4.4.35) и критерия (4.4.29).

На рис. 4.4.5 представлен рельеф, порождаемый планом $X^*_{5+1} = \{X^{(i)}\}$, где i = 1, 5, 6, 7, 8, 9 (крестики на рисунке). Рельеф имеет четыре локальных максимума (i = 3, 6, 7, 8) и глобальный — в точке $X^{(3)}$. Глобальный максимум уверенно находился выбранным алгоритмом поиска.

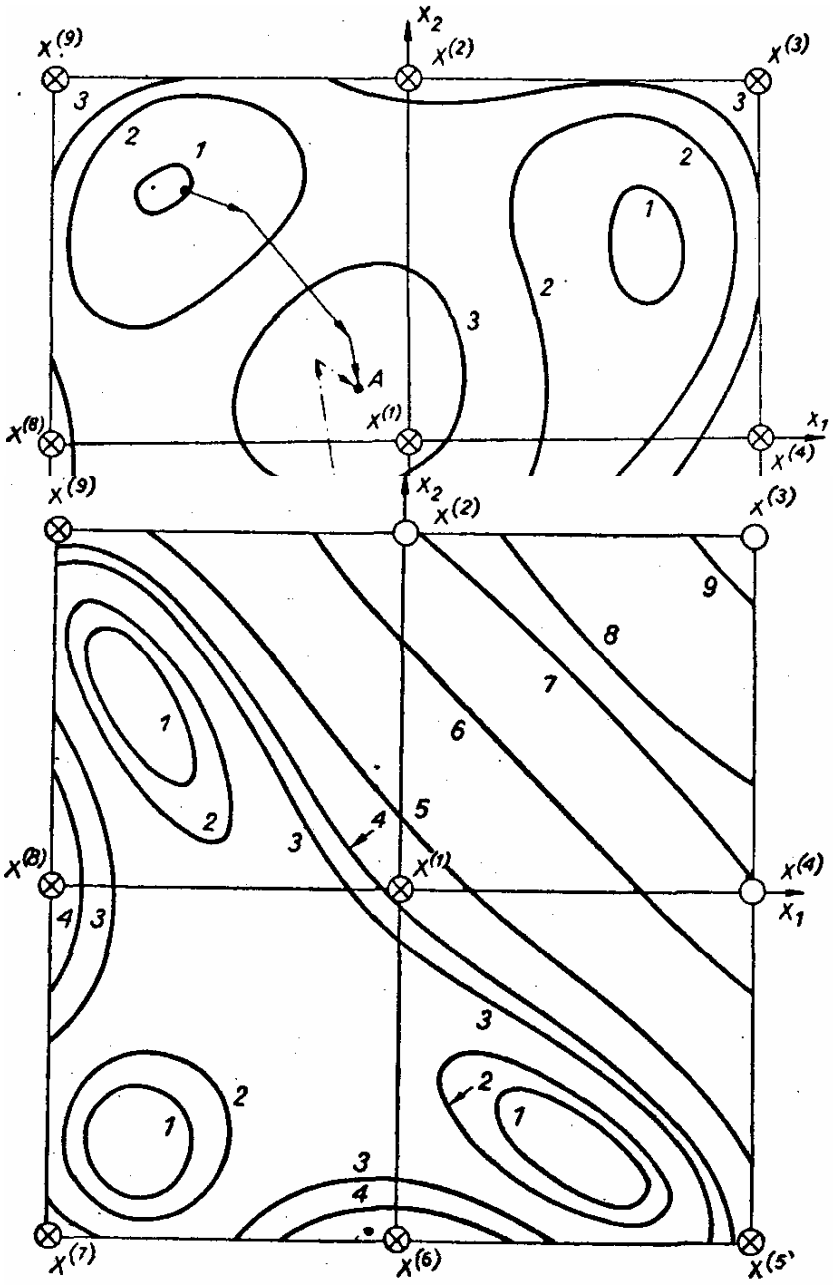


Рис. 4.4.4. Линии равного уровня при вакантной угловой точке $X^{(5)}$. Значения критерия на уровнях: *1* — 1400, *2* — 1500, *3* — 1600, *4* — 2100, 5 — 3100, *6* — 4200.

Рис. 4.4.5. Линии равного уровня для шеститочечного опорного плана. Значения критерия на уровнях: *1* — 24, *2* — 25, *3* — 28, *4* — 30, 5 — 40, *6* — 100, 7 — 200, S — 300, *9* — 700.

В случае трех вакантных точек на одной стороне области планирования S (рис. 4.4.6) любая точка, принадлежащая этой стороне, решает задачу.

Выше рассмотрены планы для объектов, описываемых полиномами второго порядка. Эти планы опирались на девять точек, равномерно покрывающих квадрат области планирования 5, и хорошо изучены теоретически [223]. Рассмотрим теперь D-опти-мальные планы для кубической регрессии [32], приведенные в работе [119]:

$$y = c_1 + c_2x_1 + c_3x_2 + c_4x_1^2 + c_5x_1x_2 + c_6x_1^2x_2 + c_7x_1^3. \qquad (4.4.36)$$

На рис. 4.4.7 приведены линии равного уровня для опорного плана из семи точек *A, B, D, E, F, G, H* (обозначены крестиками). Видно, что полученная задача оптимизации многоэкстремальна: небольшие локальные экстремумы расположены в точках *A, H, F* и *E,* а глубокий глобальный — в точке *C.* Последний легко отыскивается любым локальным методом поиска. Заметим, что полином (4.4.36) содержит x_2 лишь в первой степени,

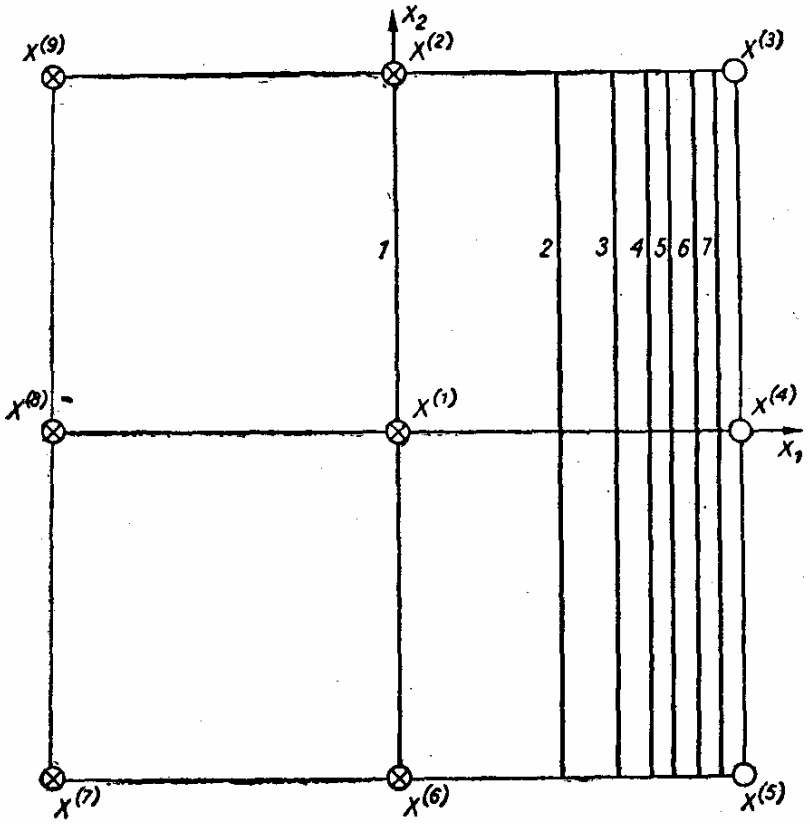


Рис. 4.4.6. Линии равного уровня при вакантных точках $X^{(3)}$, $X^{(4)}$ и $X^{(5)}$. Значения критерия на уровнях: *1* — 0, *2* — 27, *3* — 53, *4* — 80, *5* — 106, *6* — 133, *7* — 160.

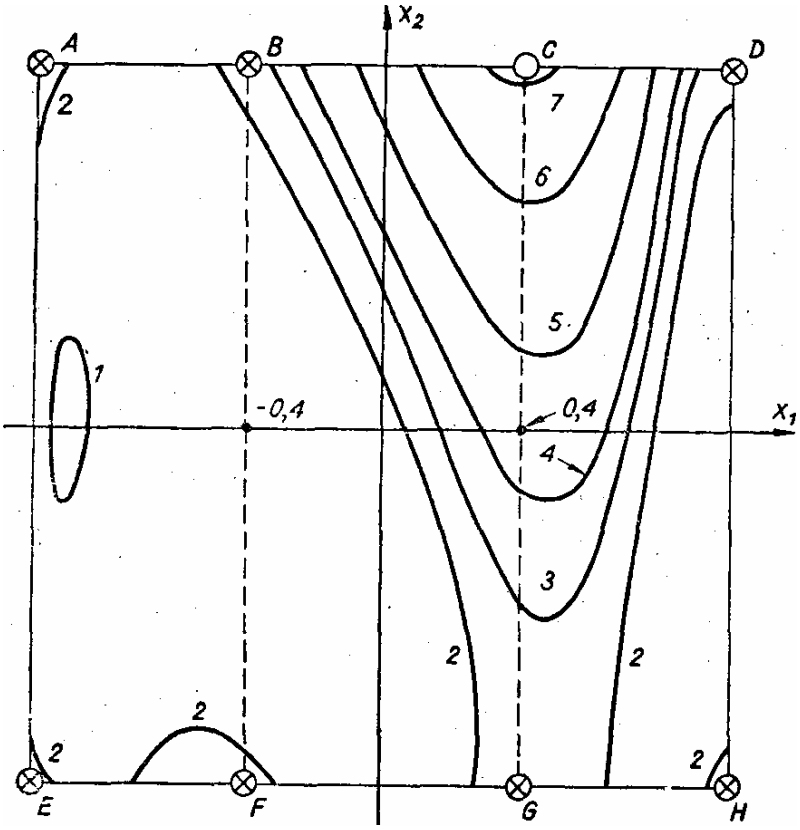


Рис. 4.4.7. Линии равного уровня для полинома третьей степени (4.4.36) при вакантной точке С, Значения критерия на уровнях: *1* — 320, *2* — 440, *3* — 500, *4* — 560, 5 — 680, *6* — 860, 7 — 1050.

что и определяет двухуровневость полученного оптимального плана. Рассмотрим полином

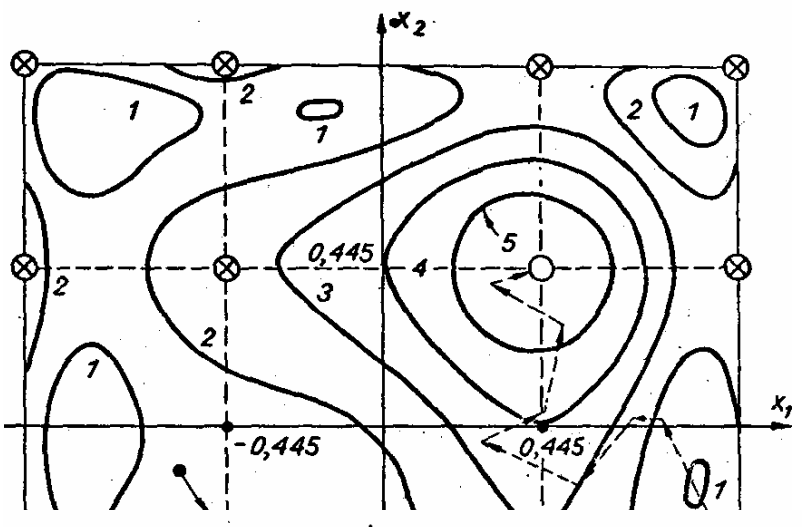
$$y = c_1 + c_2x_1 + c_3x_2 + c_4x_1^2 + c_5x_2^2 + c_6x_1^3 + c_7x_2^3 \qquad (4.4.37)$$

без перекрестных членов. Пусть опорные планы содержат 15 точек (на рис. 4.4.8 и 4.4.9 они обозначены крестиками и показан рельеф). Хорошо видны многоэкстремальность этих задач и большая зона глобального экстремума, вероятность попасть в которую достаточно велика. Траектории поиска показывают это. Полином

$$y = c_1x_1^2 + c_2x_2^2 + c_3x_1^3 + c_4x_2^3 \qquad (4.4.38)$$

и опорный план с угловыми точками порождают линии равного уровня, изображенные на рис. 4.4.10. Здесь четыре локальных экстремума, расположенных на пересечении осей координат с областью планирования, и все они являются глобальными.

Рис. 4.4.8. Линии равного уровня в случае полинома (4.4.37) при вакантной точке (0,445; 0,445). Значения критерия на уровнях: *1* — 5300, *2* — 5700, *3* — 6000, *4* — 6350, *5* — 6700.



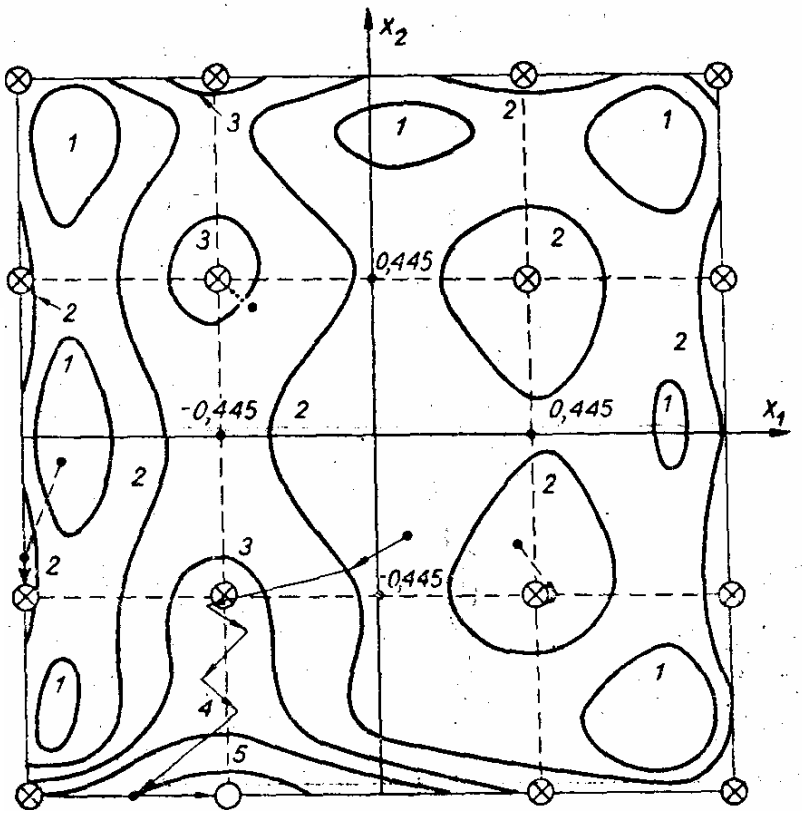


Рис. 4.4.9. Линии равного уровня для полинома (4,4.37) при вакантной точке (-0,445; -1). Значения критерия на уровнях: *1* — 5300, *2* — 5600, *3* — 5900, *4* — 6150, 5 — 6700.

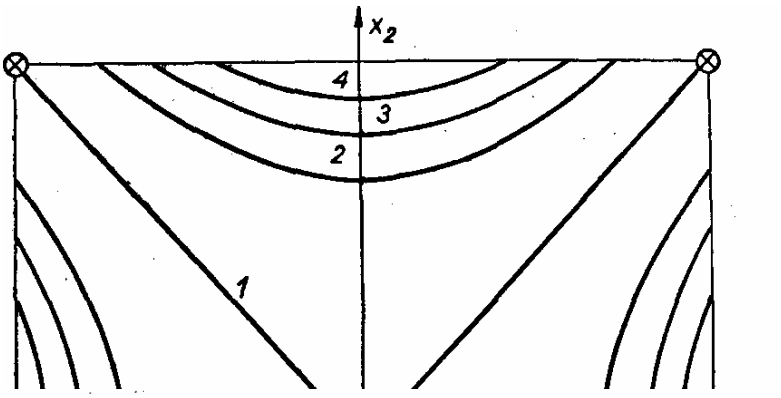


Рис. 4.4.10. Линии равного уровня в случае полинома (4.4.38). Значения критерия на уровнях: *1* — О, *2* — 15, *3* — 29, *4* — 44.

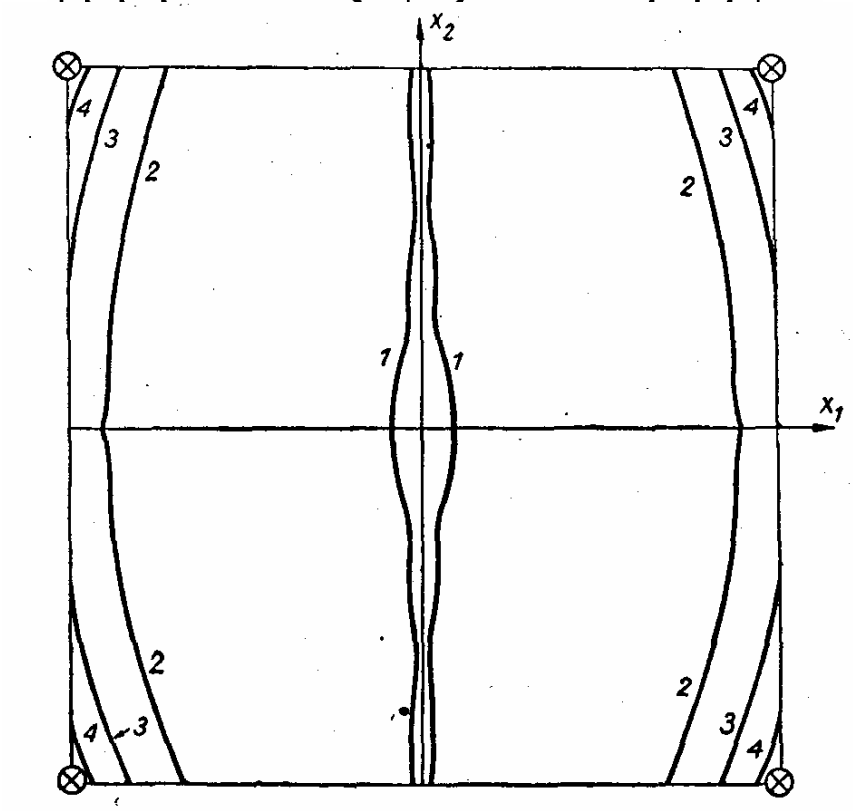


Рис. 4.4.11. Линии равного уровня в случае полинома (4.4.39). Значения критерия на уров- нях: *1* — 256, *2* — 332, *3* — 408, *4* — 484.

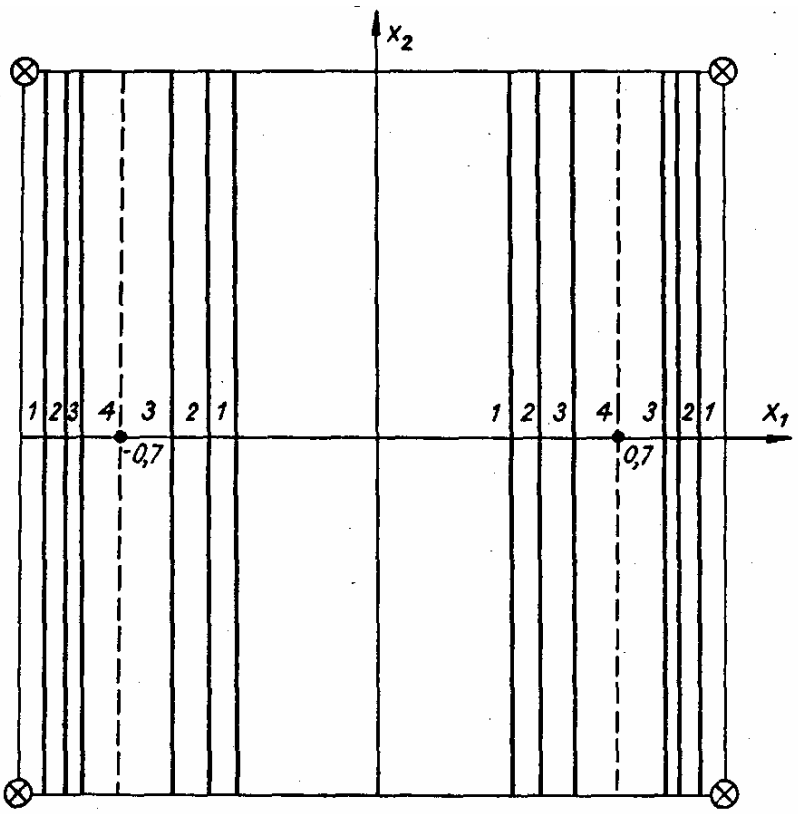


Рис. 4.4.12. Линии равного уровня в случае полинома (4.4.40). Значения критерия на уровнях: *1* — 0, *2* — 1, *3* — 2, *4* — 3. Пунктиром обозначены линии максимума.

Рельеф критерия для четырехточечного плана и полинома

$$y = c_1x_1x_2 + c_2x_1^2 + c_3x_1^2x_2 + c_4x_1^3 \qquad (4.4.39)$$

показан на рис. 4.4.11. Здесь также все четыре локальных экстремума глобальны и совпадают с точками опорного плана. Наконец, на рис. 4.4.12 представлен рельеф критерия четырехточечного плана и модели четвертого порядка вида

$$y = c_1x_2 + c_2x_1^2 + c_3x_1x_2^3 + c_4x_1^4.$$

Рельеф имеет линейчатый характер. Глобальные экстремумы расположены вдоль пунктирных линий и совпадают с локальными.

Из рассмотренных экспериментальных наблюдений вытекает, что задача синтеза оптимального плана является, как правило, многоэкстремальной. Она имеет глубокий глобальный экстремум, и ее решение может быть найдено методом случайного поиска (4.4.32) путем организации многократных спусков из случайно выбранных начальных точек. Так как зона притяжения глобального экстремума в этих задачах не мала, то вероятность отыскания глобального экстремума за несколько локальных спусков достаточно велика и этот алгоритм можно рекомендовать для решения задачи синтеза оптимальных планов.

4.4.3. Диалоговый синтез плана

Описанные в предыдущем подразделе алгоритмы последовательного синтеза оптимальных планов удобно реализуются в интерактивном режиме взаимодействия пользователя с ЭВМ. На рис. 4.4.13 представлена блок-схема такого взаимодействия на N-м шаге. Пользователь сообщает ЭВМ:

1. Область планирования S_N — например, в виде гиперпараллелепипеда:

$$a_i^{(N)} \le x_i \le b_i^N \quad (i = 1, ..., n), \text{.....} \quad (4.4.41)$$

который определяется векторами

$$A^{(N)} = (a_1^{(N)}, ..., a_n^{(N)});$$
$$B^{(N)} = (b_1^{(N)}, ..., b_n^{(N)}). \quad (4.4.42)$$

2. Систему функций Φ_N:

$$\Phi_N = (\varphi_1^{(N)}(X), ..., \varphi q_N^{(N)}(X)), \quad (4.4.43)$$

с помощью которых описывается модель на N-м шаге

$$y^{(N)} = \sum_{i=1}^{q_N} c_i \varphi_i^{(N)}(X), \quad (4.4.44)$$

имеющая q_N неизвестных параметров.

Проще всего эту систему функций представить конъюнкциями:

$$\varphi_i^{(N)}(X) = \prod x_i^{p_{ji}^{(N)}} \quad (4.4.45)$$

которые определяются показателями степени p_{ij}. Матрица

$$P^{(N)} = \| p_{ij}^{(N)} \|_{q_N \times n} \quad (4.4.46)$$

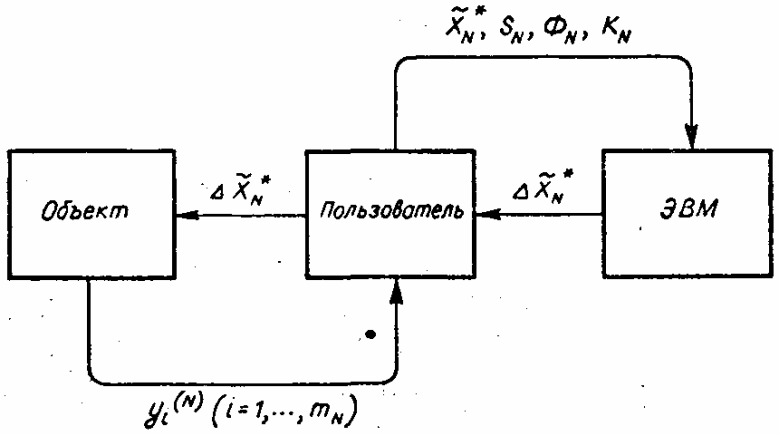


Рис. 4.4.13. Схема диалогового взаимодействия пользователя с ЭВМ в процессе планирования оптимальных экспериментов.

однозначно определяет систему функций Φ_N. Например, система функций (4.4.3) определяется матрицей (6×2) вида

$$P=\begin{Vmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 2 & 0 \\ 0 & 2 \\ 1 & 1 \end{Vmatrix} \qquad (4.4.47)$$

3. Опорный план $\tilde{X}^*_N$:

(4.4.48)

определяющий исходные (например, очевидные или уже реализованные) точки плана, которые, по мнению пользователя, наверняка входят в искомый на (N+1)-м этапе план:

$$\tilde{X}^*_{N+1}=\langle \tilde{X}^*_N,\ X_{l_N+1}, \ldots, X_{l_N+k_N} \rangle, \qquad (4.4.49)$$

образованный добавлением новых точек (экспериментов), число которых k_N должно быть также определено пользователем.

При этом все параметры задания должны удовлетворять естественному условию

$$l_N+ k_N \geq q_N, \qquad (4.4.50)$$
$$\tilde{X}^*_N=\langle X_1, \ldots, X_{l_N} \rangle,$$

т. е. число экспериментов не должно быть меньше числа неизвестных параметров модели.

4. Критерий оптимальности K_N плана на N-м этапе, который является функцией (4.4.18), заданной на дисперсионной матрице (4.4.15). Обычно различают D-оптимальный план, критерий оптимальности которого имеет вид (4.4.19), A-оптималь-ный — вида (4.4.20) и E-оптимальный — с критериями вида (4.4.21). Возможен и любой другой способ задания критерия K.

Таким образом, функция ψ перехода от (N—1)-го этапа к N-му

(4.4.51)

$$S_N, \Phi_N, \tilde{X}^*_N, k_N=\psi(S_{N-1}, \Phi_{N-1}, \tilde{X}^*_{N-1}, k_{N-1}, I_N)$$

реализуется на основе информации I_N, полученной после (N—1)-го этапа и имеющей двоякий характер — формальный и неформальный. С одной (формальной) стороны, — это результаты экспериментов, проведенных на (N—1)-м этапе, т. е. наблюдения

$$\langle X_i^{(N\text{-}1)}, y_i^{(N\text{-}1)} \rangle \quad (i = 1, ..., m_{N\text{-}1}), \quad (4.4.52)$$

где $X_i^{(N\text{-}1)}$ — точки экспериментов; $y_i^{(N\text{-}1)}$ — их результаты; $m_{N\text{-}1}$ — число таких экспериментов на (N—1)-м этапе. Заметим, что вовсе необязательно $m_{N\text{-}1}=k_{N\text{-}1}$ и $X_i^{(N\text{-}1)}$ совпадут с рекомендованными на предыдущем этапе, так как экспериментирование с объектом является слишком сложным процессом, в который неизбежно вносят искажения внешние неучитываег мые факторы.

С другой (неформальной) стороны, процесс общения пользователя с объектом неизбежно изменяет его представления о структуре модели объекта, области планирования и критериях. Это обстоятельство и заставляет пользователя реализовать функцию ψ в (4.4.51).

Однако подобные изменения обычно бывают незначительными и поэтому легко реализуются в интерактивном режиме. Для этого достаточно дать пользователю возможность корректировать указанную информацию, т. е. последовательно показывать ему информацию о предыдущем (N—1)-м этапе и выяснять, будет ли коррекция, и если будет, то какая. Структура такого диалога пользователя с ЭВМ описана подробно в работе [33].

Формальный этап синтеза состоит в решении задачи

$$K_N(\tilde{X}^*_N, \Delta\tilde{X}_N) \to \min_{\Delta\tilde{X}_N \in S_N} \Rightarrow \Delta\tilde{X}^*_N,$$

где

$$\Delta\tilde{X}_N = \langle X_1^{(N)}, \ldots, X_{k_N}^{(N)} \rangle \quad (4.4.54)$$

— дополнительные точки плана. Решение задачи (4.4.53) дает $\Delta\tilde{X}^*_N$ — оптимальную достройку плана $\tilde{X}_N$, т. е. план

$$\tilde{X}^*_{N+1} = \langle \tilde{X}^*_N, \Delta\tilde{X}^*_N \rangle, \quad (4.4.55)$$

который предлагается пользователю для реализации дополнительных точек $\Delta\tilde{X}^*_N$. Фактическая их реализация, как сказано выше, зависит от многих внешних факторов и априори не определена.

Процесс решения оптимизационной задачи (4.4.53) происходит методом глобального случайного поиска: если $l_N \geq q_N$, то к плану последовательно присоединяются точки достройки, т. е. каждый раз поиск идет в R^n. Если же $l_N < q_N$, то необходимо сначала расширить размерность пространства поиска до минимального k'_N, кратного n:

$$k'_N = \min_{l_N + k \geqslant q_N} k. \quad (4.4.56)$$

В этом случае будет определено k'_N/N точек достройки плана. Для последующих точек условие (4.4.50) будет выполняться и процедура достройки может иметь последовательный характер, описанный в подразделе 4.4.2.

Таким образом, описанный выше интерактивный режим синтеза оптимальных планов эксперимента позволяет пользователю эффективно и оптимально решать сложные задачи планирования эксперимента [33].

§ 4.5. Адаптация в процессах восстановления числовых таблиц [294, 295]

Рассматривается задача восстановления таблиц количественных показателей. Таблица имеет вид прямоугольной матрицы

$$A = || a_{ij} ||_{N\times M}, \qquad (4.5.1)$$

где $a_{ij} \in S_{ij}$ (i=1, ...,N; j=l, ..., M); $S = || S_{ij} ||_{N\times M}$ — заданная матрица интервалов изменений количественных показате лей этой таблицы. Предполагается, что матрица (4.5.1) обладает избыточностью, т. е. между элементами таблицы имеются связи, о характере которых априорная информация, вообще говоря, отсутствует. В матрице имеются m пропусков, которые образуют набор значений неизвестных элементов

$$Z = \{z_1, ..., z_m\}.$$

Для восстановления значений неизвестных элементов можно применить метод многомерной линейной экстраполяции [185, 186], который при отсутствии информации о характере связей между элементами таблицы использует естественную гипотезу о кусочной линейности этих связей. В дальнейшем будем широко использовать указанную гипотезу.

Рассмотрим сначала простейший случай, когда m=1, т. е. $Z=z$. Тогда в таблице (4.5.1) имеется один пропуск на месте элемента a_{kl}. Для восстановления значения Z элемента a_{kl} матрица A разделяется на три части: X_{kl}, Y_{kl} и V_{kl}, где X_{kl} — матрица размерности (N-1)×(M-1), полученная из матрицы (4.5.1) после вычеркивания k-й строки и l-го столбца:

$$X_{kl} = || a_{ij} ||_{(N-1)\times(M-1)} \quad \text{при } i\neq k \text{ и } j\neq l; \qquad (4.5.2)$$

Y_{kl} — l-й вектор-столбец матрицы (4.5.1) без элемента a_{kl}:

$$Y_{kl} = || a_{ij} ||_{(N-1)\times 1} \text{ при } i\neq k; \qquad (4.5.3)$$

V_{kl} — k-я вектор-строка матрицы (4.5.1) без элемента a_{kl}.

$$V_{kl} = \| a_{kj} \|_{1 \times (M-1)} \text{ при } j \neq l. \quad (4.5.4)$$

Значение z тогда будет равно оценке неизвестного элемента $\hat{a}_{kl}$:

$$z = \hat{a}_{kl} = \varphi(X_{kl}, Y_{kl}, V_{kl}), \quad (4.5.5)$$

где φ — решающее правило такой оценки.

В качестве решающего правила φ и предлагается применить метод многомерной линейной экстраполяции, позволяющей оценить элемент a_{kl}. Воспользуемся следующей модификацией этого метода. Опорная последовательность образуется матрицей X_{kl} в форме

$$\langle U_i, a_{il} \rangle \quad (i = 1, ..., k-1, k+1, ..., N), \quad (4.5.6)$$

где U_i — $(M—1)$-мерный вектор вида

$$U_i = (a_{i1}, ..., a_{i,l-1}, a_{i,l+1}, ..., a_{i,M}), \quad (4.5.7)$$

или в форме

$$\langle W_j, a_{kj} \rangle \quad (j = 1, ..., l-1, l+1, ..., M), \quad (4.5.8)$$

где W_j — $(N—1)$-мерный вектор вида

$$W_j = (a_{1j}, ..., a_{k-1,j}, a_{k+1,j}, a_{Nj}). \quad (4.5.9)$$

Искомая оценка a_{kl} определяется как линейная экстраполяция опорной последовательности по U_k (или W_l). Для этого образуем вектор

$$U = \sum_{i \in L_k} \lambda_i U_i, \quad (4.5.10)$$

где $\sum_{i \in L_k} \lambda_i = 1$, а множество L_k состоит из t номеров, подбираемых

по признаку близости U_i к U_k в соответствии с выбранной мерой $\rho(U_i, U_k)$ — например, евклидовой:

$$\rho(U_i, U_k) = | U_i - U_k |. \quad (4.5.11)$$

Как видно, задача отбора U_i в сумму (4.5.10) сводится к выбору t векторов U_i, ближайших к заданному по критерию выбранной меры ρ.

В зависимости от величины t возможны два режима работы алгоритма.

1. $t = M$. В этом случае вектор параметров

$$\Lambda = (\lambda_1, ..., \lambda_M) \quad (4.5.12)$$

определяется из системы линейных уравнений:

$$U_k = \sum_{i \in L_k} \lambda_i U_i; \quad \sum_{i \in L_k} \lambda_i = 1. \quad (4.5.13)$$

Это обычная линейная зависимость строк таблицы *A,* которая постулируется для удобства решения задачи. Однако условию решения задачи $t=M$ далеко не всегда удается удовлетворить, например при $N<M$. Поэтому особый интерес представляет следующий случай.

2. $t<M$. Здесь для определения параметров Λ необходимо ввести дополнительную информацию в виде гипотезы о близости U_k и (4.5.10), т. е. в виде той же меры $\rho(U, U_k)$, минимум которой дает возможность определить искомые параметры Λ.

Для этого следует решить задачу минимизации:

$$\rho\left(\sum_{i \in L_k} \lambda_i U_i, U_k \right) \to \min_{\Lambda \in \Omega} \Rightarrow \Lambda^*, \quad (4.5.14)$$

где

$$\Omega: \sum_{i \in L_k} \lambda_i = 1, \quad (4.5.15)$$

а Λ^* есть решение задачи минимизации.

Теперь оценка искомого экстраполируемого значения очевидна:

$$\hat{a}_{kl} = \sum_{i \in L_k} \lambda^*_i a_{il}, \quad (4.5.16)$$

где λ^*_i ($i \in L_k$) — решение системы (4.5.13) или задачи (4.5.15). В этом и состоит решающее правило φ в выражении (4.5.5). Далее рассмотрим общий случай: $m \geq 1$. Построим функцию невязки таблицы *A* и ее поэлементной экстраполяции в виде суммы

$$Q(Z) = \sum_{i,j} f(a_{ij} - \hat{a}_{ij}), \quad (4.5.17)$$

где $\hat{a}_{ij}$ — оценка экстраполированного значения a_{ij} изложенным способом (4.5.16); f — унимодальная неотрицательная функция, например $f(\bullet) = |\bullet|$, а в местах соответствующих пропусков таблицы *A* стоят параметры вектора *Z*.

Тогда задача восстановления таблицы представляется в виде задачи оптимизации: необходимо минимизировать суммарную невязку (4.5.17) экстраполяции всех значений элементов таблицы путем подбора значений неизвестных элементов (4.5.2), т. е. решить задачу:

$$Q(Z) = \rightarrow \min_{z \in S} \Rightarrow Z^*, \qquad (4.5.18)$$

где $Z^* = (z^*_1, \ldots, z^*_m)$ — решение этой задачи — оценки значений неизвестных элементов; S — заданная ранее матрица интервалов изменения элементной таблицы A. Выбор метода оптимизации для решения задачи (4.5.18) зависит, очевидно, от характера функции (4.5.17).

Для анализа поведения невязки (4.5.17) решались следующие модельные задачи. Модель представляла собой матрицу (4.5.1) с заданной избыточностью. Избыточность моделировалась простейшей линейной зависимостью строк матрицы (4.5.1) следующим образом: первые $t<N$ строк матрицы были независимы, они заполнялись целыми случайными числами с равномерным распределением в интервале [0; 9], а остальные $N—l$ строк представляли собой линейные комбинации независимых строк, т. е. $a_{ij} = \sum_{l} \alpha_{ik} a_{kj}$ для $i = l+1, \ldots, N$; $j = 1, \ldots, M$.

где a_{ik} — целые случайные числа с равномерным распределением в интервале [0; 9]. Ранг r такой матрицы, очевидно, равен числу линейно-независимых строк, т. е. $r = l$.

Пример 1. $N = 5$, $M = 4$, $l = 2$, $m = 1$. Функция невязки (4.5.17) для данного примера представлена на рис. 4.5.1. Хорошо видно, что она имеет экстремальный характер, причем ее минимум соответствует истинному значению восстанавливаемого элемента. Любой локальный метод однопараметрической оптимизации этой функции обеспечит точное восстановление утраченного элемента таблицы.

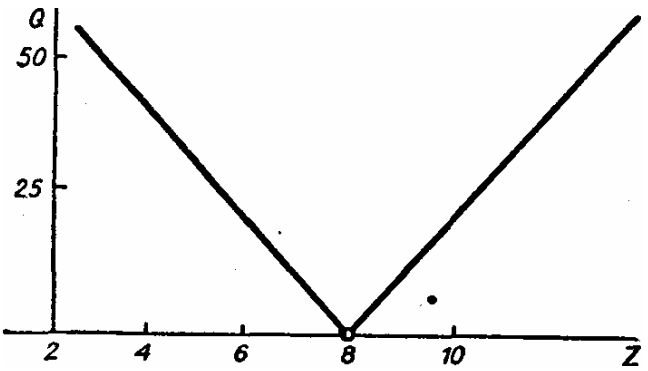


Рис. 4.5.1. Функция невязки линейно-зависимой таблицы (m= = 1). Минимум: $z^*=a_{kl}=8$.

Нетрудно заметить, что для точного восстановления одного элемента такой линейно-избыточной матрицы необходимо, чтобы

$r = \min (N\text{-}1; M\text{-}1)$.

Произвольный характер восстанавливаемой матрицы моделировался путем введения элемента стохастичности. Для этого ко всем значениям элементов a_{ij} линейно-избыточной матрицы добавлялась стохастическая составляющая

$$a'_{ij} = a_{ij} + \beta\varepsilon_k$$

где a'_{ij} — элемент моделируемой матрицы; a_{ij} — элемент линейно-избыточной матрицы; β — коэффициент стохастичности; ε_k — независимые случайные числа, равномерно распределенные в интервале [—1; 1], $k = NM$—1.

Вид функции (4.5.17) для различных значений коэффициента стохастичности β приведен на рис. 4.5.2, а, *б.* В отличие от первой модели функция невязки здесь многоэкстремальна, ее глобальный минимум и решает задачу восстановления. В табл. 4.5.1 приведены положения экстремумов функции невязки для различных значений параметра стохастичности β.

Таблица 4.5.1

β	0	0,1	0,3	1	2	3	4	5
*Z**	8,0	7,8	7,6	6,5	5,6	5,2	4,9	4,6

Из таблицы видно, что значительное случайное изменение таблицы (β = 3—5) незначительно изменяет положение глобального экстремума функции невязки. Это свидетельствует об эффективности предложенного способа восстановления пропусков в таблице.

Для решения задачи (4.5.8) в таких условиях необходимо применять глобальный поиск [116] — например, поиск со сглаживанием (см. § 3.6).

Теперь рассмотрим случай восстановления таблицы с двумя ($m \neq 2$) пропущенными элементами. На рис. 4.5.3 представлены линии равного уровня функции невязки для линейно-избыточной модели таблицы *A*, но при $m = 2$. Функция $Q(z_1, z_2)$ имеет нулевое минимальное значение при аргументах, совпадающих с точными значениями восстанавливаемых элементов.

Как видно, задача унимодальна и экстремум в данном случае находится любым локальным методом поисковой оптимизации.

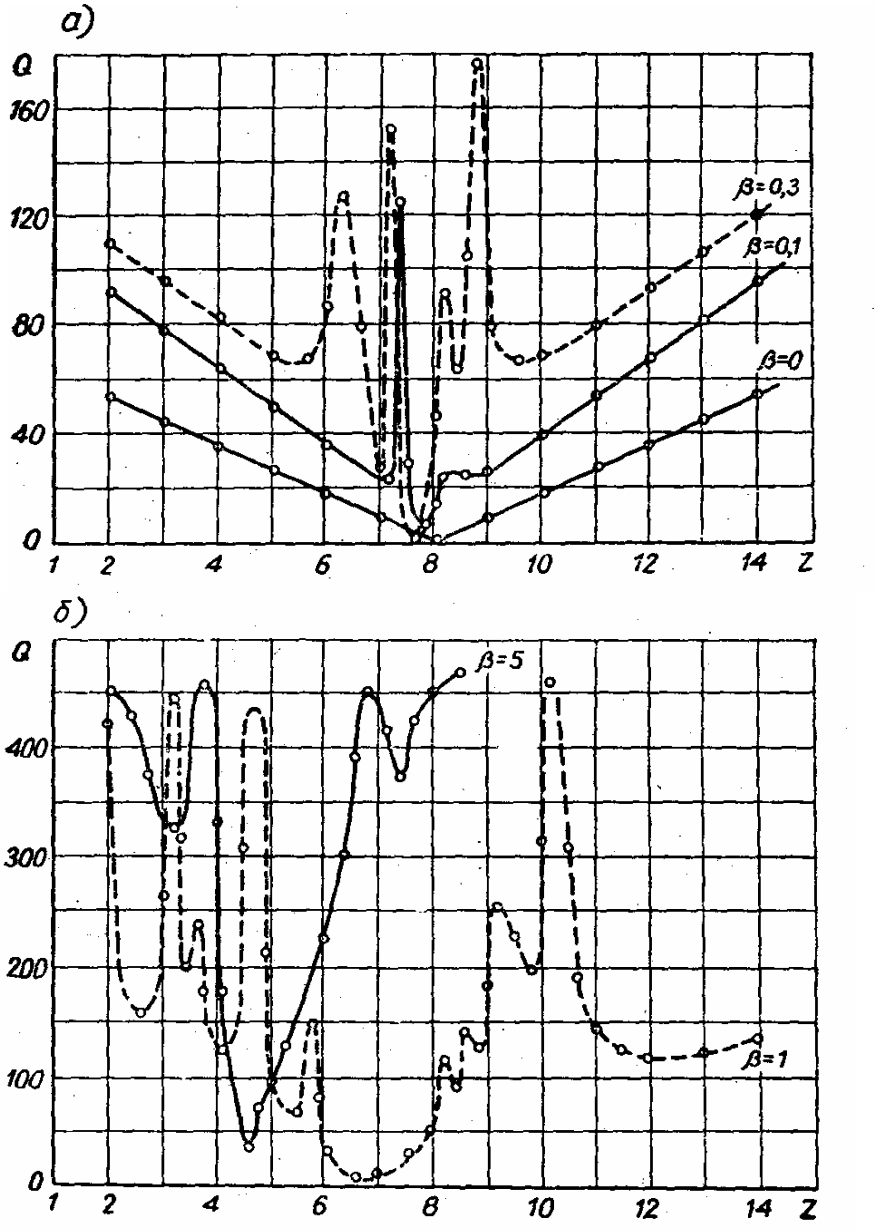


Рис. 4.5.2. Функция невязки линейно-зависимой таблицы со стохастическим возмущением: *а* — при малой стохастичности β, *б* — при большой стохастичности в.

Вид линий равного уровня функции невязки для той же таблицы, но «зашумленной» стохастической добавкой при (β = = 0,1 приведен на рис. 4.5.4. Здесь минимальное значение функции невязки не равно нулю, но ее оптимум близок к истинному. В этом случае задача также многоэкстремальна и для ее решения целесообразно применять глобальные методы случайного поиска (4.4.6).

Таким образом, применение метода многомерной линейной экстраполяции в сочетании с процедурой параметрической оп-

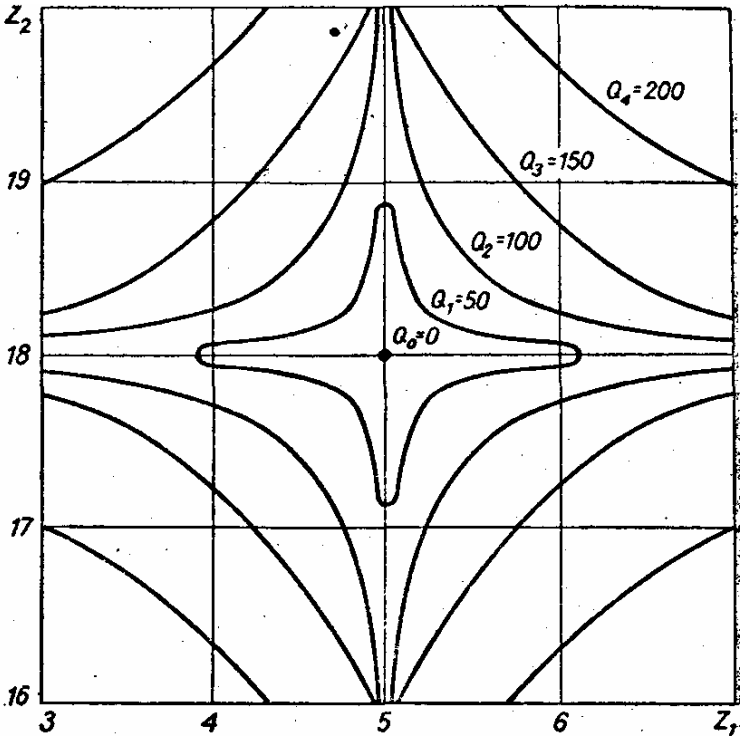


Рис. 4.5.3. Линии равного уровня функции невязки линейно-зависимой модельной таблицы при m=2.

тимизации дает возможность достаточно точно восстанавливать элементы избыточной матрицы.

Для иллюстрации процесса восстановления пропущенных значений таблицы были проведены два цикла экспериментального поиска со стохастическим сглаживанием, показанные на

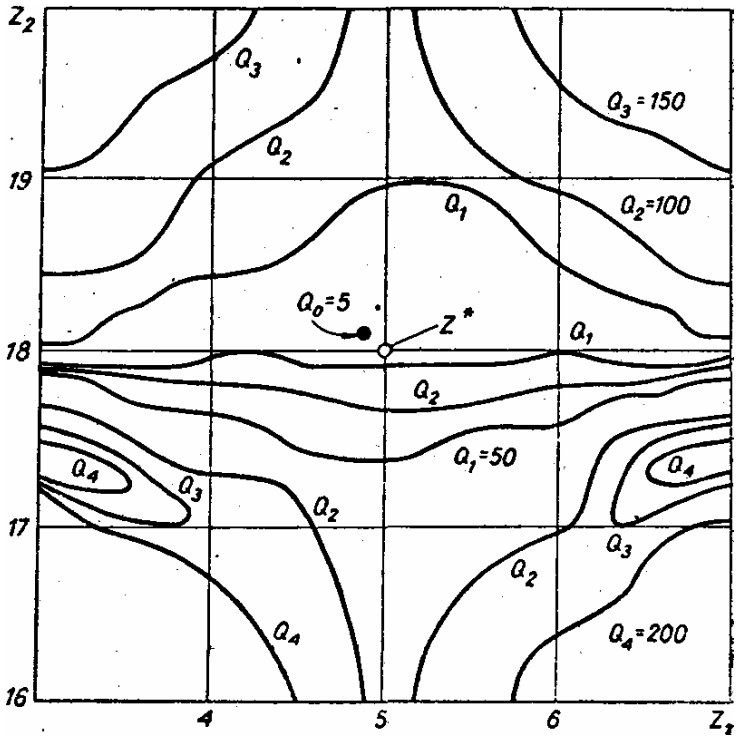


Рис. 4.5.4. Линии равного уровня функции невязки линейно-избыточной модельной таблицы с параметром стохастичности β=0,1.

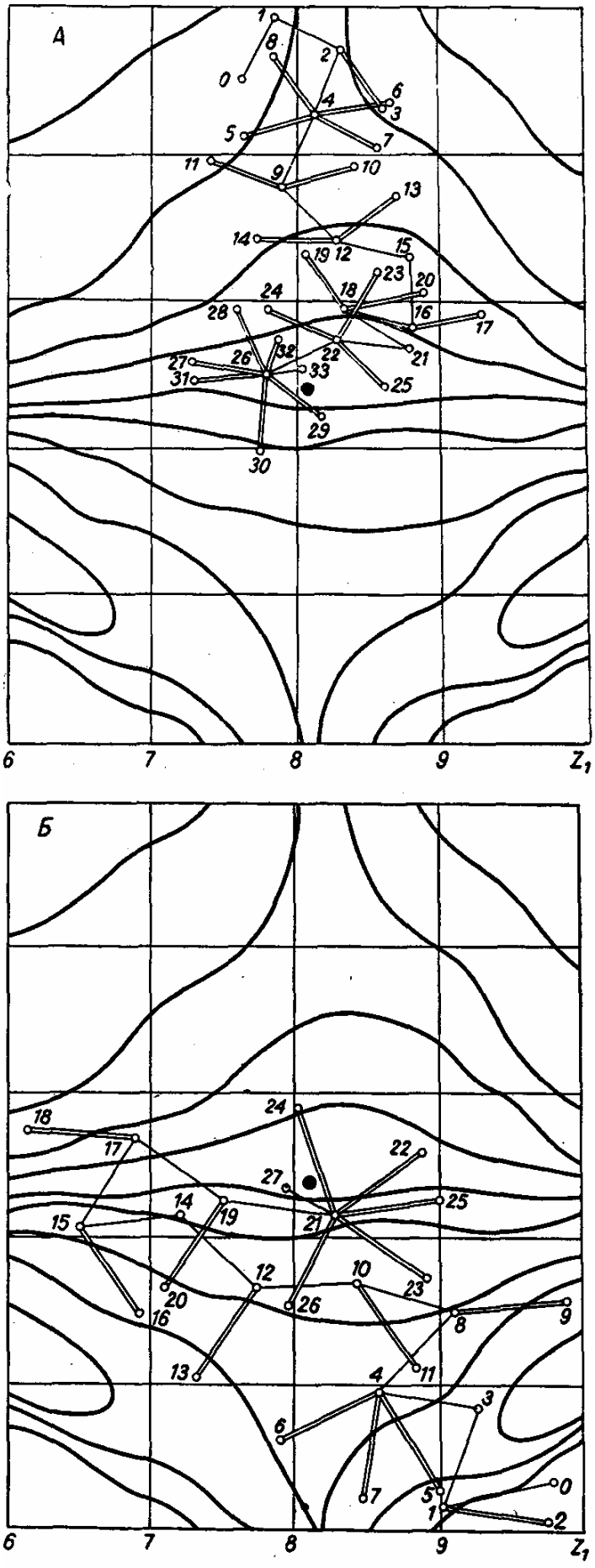


Рис. 4.5.5. Траектории случайного поиска: *А* — Z_0= = (7,6; 8,5); a=0,5; g=0,2; Z_{33}=($8_1$05; 6,52); *Б* — Z_0= = (9,7; 4,4); a=0,75; g=0,3; Z_{27}= (7,95; 6,32). Черный кружок — точное положение экстремума.

рис. 4.5.5, *А*, *Б*. Сглаживание вводилось на базе четырех случайных измерений:

$$\bar{Q}(Z)=\frac{1}{4}\sum_{i=1}^{4}Q(Z+g\xi_i); \qquad (4.5.19)$$

где g — величина области сглаживания; ξ_i — i-я реализация: единичного случайного вектора, равномерно распределенного а пространстве параметров $\{Z\}$. В качестве алгоритма поиска был использован адаптивный случайный поиск с «возвратом»:

$$\Delta Z_N=\begin{cases} a\xi_N \text{ при } \Delta\bar{Q}_{N-1}<0; \\ -\Delta Z_{N-1}+a\xi_N \text{ при } \Delta\bar{Q}_{N-1}\geqslant 0, \end{cases} \qquad (4.5.20)$$

где величина шага a уменьшалась в два раза после пяти неудачных шагов (и возвратов).

На рис. 4.5.5 показаны две траектории поиска минимума стохастически сглаженной функции (4.5.19); двойными линиями показаны неудачные шаги. Хорошо видно, что в обоих случаях поиск вывел в зону экстремума, причем именно здесь был уменьшен рабочий шаг.

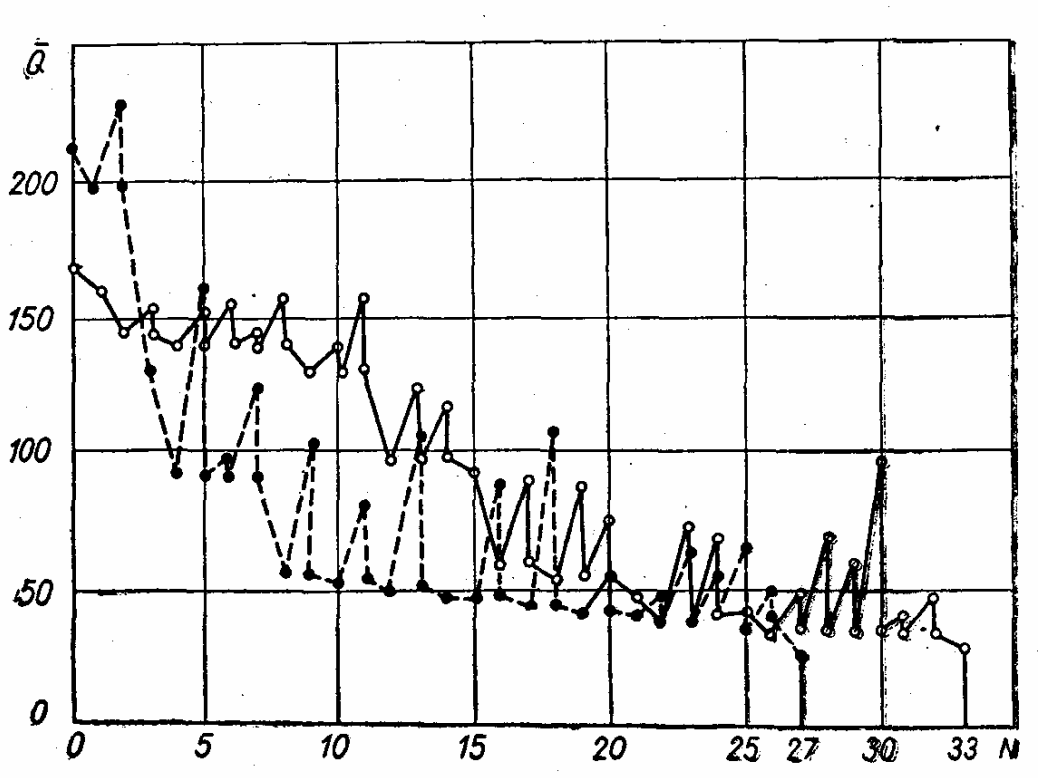


Рис. 4.5.6. Поведение сглаженного значения минимизируемой функции при восстановлении таблицы для двух начальных условий. Сплошная линия — процесс *А,* пунктир — процесс *Б* (см. рис. 4.5.5).

Поведение сглаженного значения (4.5.19) минимизируемой функции в этих процессах поиска показано на рис. 4.5.6. Наглядно видна сходимость процесса.

Таким образом, применение глобального случайного поиска позволяет эффективно восстанавливать утраченные элементы таблицы.

§ 4.6. Адаптация алгоритмов распознавания

Решение проблемы повышения эффективности распознающих решающих правил в настоящее время затрудняется значительной априорной неопределенностью структуры решаемой задачи распознавания. Там, где эта структура проста, удается предложить весьма эффективные способы синтеза распознающих решающих правил [13, 30, 89, 90, 151]. Однако научно-техническая практика очень часто выдвигает такие сложные задачи, относительно структуры которых априори почти ничего нельзя сказать. Именно в этом случае со всей остротой встает проблема выбора структуры решающего правила. Интуиция исследователя, на которую обычно уповают [30], здесь дает крайне мало, а зачастую и ухудшает выбор ввиду влияния малосущественных факторов (мода, опыт исследователя, случай и т. д.).

Действительно, успешному применению алгоритма должна предшествовать проверка некоторых условий его применимости. Примерами таких условий могли служить, например, гипотеза компактности в методе потенциальных функций [13], требование нормальности априорных распределений в статистических алгоритмах [89, 90], условие неразличимости признаков [90] и т. д. Проверка этих условий для сложных задач распознавания обычно невозможна [285].

Однако при синтезе алгоритмов необходимо прежде всего определять их структуру. Это можно сделать двояким образом. С одной стороны, можно пытаться создать метаструктуру объектов распознавания, обобщающую известные структуры. Примером такой метаструктуры являются всякого рода языки описания ситуаций, например двумерные грамматики при распознавании плоских изображений [241]. Этот путь требует чрезвычайно тщательного исследования и формализации рассматриваемого поля задач, что удается далеко не всегда и лишь для весьма узкого класса задач. Пожалуй, единственным примером такой формализации является создание уже упомянутых двумерных грамматик [241]. Именно поэтому для решения «сложных задач распознавания следует изыскивать другие подходы.

В качестве одного из таких подходов можно предложить использовать структуры уже известных решающих правил. Это означает, что искомую структуру следует искать как некую комбинацию известных структур. Такой путь в сочетании с адаптацией оказался весьма плодотворным и получил название коллективного (по аналогии принятия коллективных экспертных решений). Рассмотрим его [177].

Пусть имеется L различных алгоритмов распознавания R_i (i = 1, ..., L), каждый из которых имеет следующую структуру:

$$R_i : S_i = R_i(X) = \operatorname{sgn} f_i(X), \tag{4.6.1}$$

где $S_i \in \{0; 1\}$ — решение, принимаемое i-м алгоритмом в ситуации X, а

(4.6.2)

(для простоты рассматриваем случай дихотомии). Тогда коллективное решающее правило можно представить в виде

$$R : S = F(R_1(X), ..., R_L(X)), \tag{4.6.3}$$

где функция F реализует свертку индивидуальных решений в коллективное. Приведем очевидные примеры синтеза функции F. Простейшим способом организации коллективного решения является равноправное голосование членов коллектива (рис. 4.6.1, а). Это означает, что принимается решение, собирающее большинство голосов, т. е.

$$F : S = S_i, \text{ если } k_i(X) = \max_r k_r(X), \tag{4.6.4}$$

где $k_r(X)$ — число голосов, набранное r-м решением S_r в ситуации X. При этом $\sum k_r = L$, т. е. членам коллектива запрещается голосовать одновременно за два решения. Существует немало известных методов распознавания, которые работают

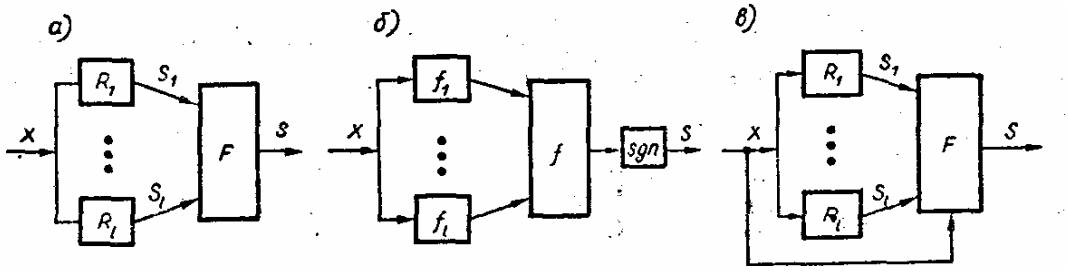


Рис. 4.6.1. Блок-схемы коллективных решений: а — голосование, *б* — гибрид, в — коллектив решающих правил.

по указанной схеме принятия коллективного решения. Примерами таких методов являются алгоритмы статистических решений [45], некоторые локальные методы [239], корреляционные методы [230] и т. д.

Такой подход можно улучшить, введя понятие взвешенного голосования. В этом случае решения каждого из членов коллектива взвешиваются и голосование производится с весами. Для этого достаточно в формуле (4.6.4) определить $k_\tau(X) = = \Sigma\, \upsilon_l$, где $\upsilon_l > 0$ — вес решающего правила R_l, а Ω_τ — множество правил, голосовавших за r-е решение. Вес υ_l может быть определен, например, как величина, обратная оценке ошибочности правила R_l. Примерами использования такого взвешенного голосования являются известные перцептронные алгоритмы [64].

Гибридное решающее правило [79] является обобщением описанных алгоритмов голосования. Оно реализуется с помощью скалярной функции свертки f, которая определяется не на множестве решений исходных решающих правил $\{S_1,...,S_L\}$, а на значениях $\{f_1, ..., f_L\}$ (см, рис. 4.6.1, *б*):

$$R : S = \operatorname{sgn} f\,(f_1(X), ..., f_L(X)\,), \qquad (4.6.5)$$

где функция f, определяющая гибрид, задается в виде пары $f = \langle\, \beta, \alpha \,\rangle$.

Здесь β — вид свертки (ее структура), а $\alpha = (\alpha_1, ..., \alpha_k)$ — вектор параметров этой свертки. Например, в случае линейной свертки гибридное решающее правило имеет вид

$$R(X, \alpha) = \operatorname{sgn}\left(\sum_{i=1}^{L} \alpha_i f_i(X) + \alpha_0 \right) \qquad (4.6.6)$$

при естественном нормирующем ограничении $\sum_{i=1}^{L} \alpha_i = 1$. В нелинейном случае F может быть представлена любой полиномиальной функцией L переменных:

$$R(X, \alpha) = \operatorname{sgn}\left(\sum_{i=1}^{k} \alpha_i \prod_{j=1}^{L} [f_j(X)]^{\upsilon_{ij}} + \alpha_0 \right), \qquad (4.6.7)$$

где $\upsilon_{ij} \in \{0;\ 1\}$, а k — число слагаемых. Выбор параметров α гибридных правил производится путем минимизации ошибок

распознавания на обучающей выборке, т. е. решением оптимизационной задачи

$$Q(R(X, \alpha), I) \to \min_{\alpha} \Rightarrow \alpha^*. \quad (4.6.8)$$

Здесь $Q(R,I)$ — функционал эффективности решающего правила R на обучающей выборке I:

$$I = \langle X_i, y_i \rangle \quad (i = 1, ..., N), \quad (4.6.9)$$

где X_i — i-я ситуация выборки, а y_i — ее принадлежность к классу ($y_i \in \{0;\ 1\}$). Функционал Q в простейшем случае может быть определен, например, как число ошибок правила R на обучающей выборке I. Однако наиболее интересна экстраполирующая способность правила R, которая определяется в виде числа ошибок правила, обученного на различных вариантах N—1 точки выборки [100], при проверке правила на оставшейся точке (метод скользящего контроля):

$$Q(R, I) = \sum_{i=1}^{N} [R(X_j, \alpha, I_j) - y_j]^2, \quad (4.6.10)$$

где $R(X_j, \alpha, I_j)$ — правило, обученное на выборке I_j, полученной из / усечением ее на один j-й элемент:

$$I_j = \langle X_i, y_i \rangle \quad (i = 1, ..., j\text{-}1; j\text{+}1, ..., N). \quad (4.6.11)$$

Для решения задачи (4.6.8) используются различные алгоритмы случайного поиска [166]. В случае большого числа k параметров оптимизация гибридного решающего правила осуществляется путем структурной оптимизации. Она реализуется последовательным добавлением к гибриду из L—1 правил нового L-го решающего правилами оптимизацией только тех параметров, которые связаны с решением L-го правила. Этим процесс оптимизации значительно упрощается, так как число оптимизируемых переменных резко сокращается. Так, для линейной свертки (4.6.6) при добавлении одного правила в свертку решение задачи оптимизации производится трлько по одному параметру (и лишь на первом шаге, когда $L = 1$, имеем двухпараметрическую оптимизацию по α_0 и α_1).

Легко видеть, что гибридное решающее правило (4.6.3), полученное путем оптимизации (4.6.8), обладает очевидным свойством: оно не может быть хуже лучшего из решающих правил, входящих в этот гибрид. Практически же оно всегда оказывается значительно лучше.

Указанное свойство гибридного решающего правила проиллюстрировано на нескольких примерах [79]. В одном из них

гибрид был образован линейной сверткой (4.6.6) двух алгоритмов Фикса—Ходжеса с различными радиусами сфер сбора информации: $f(X) = \alpha f_1(X) + (1 — \alpha) f_2(X)$. В результате оптимизации по параметру свертки а получена минимальная вероятность ошибки распознавания гибридом: P_{min}=0,29 (при $\alpha^* = = —$ 1,2), тогда как вероятности ошибки распознавания каждого алгоритма в отдельности были: P_1 = 0,43 (при α=1) и P_2=0,36 (при α = 0).

Аналогичный результат был получен при эксперименте на гибридном решающем правиле, которое было образовано линейной сверткой трех алгоритмов распознавания: Фикса—Ходжеса, потенциальной функции и линейной дискриминантной функции: $f(X) = \alpha_1 f_1(X) + \alpha_2 f_2(X) + (1 — \alpha_1 — \alpha_2) f_3(X)$. После оптимизации по параметрам свертки α_1 и α_2 глобальным алгоритмом случайного поиска вероятность ошибки распознавания гибридного решающего правила была P_{min}=0,35, а вероятности ошибки распознавания каждого из алгоритмов, свернутых в гибрид, составляли соответственно: P_1 = 0,49 (при α_1 = l, $\alpha_2 = = 0$); P_2 = 0,43 (при α_1=0, α_2=l) и P_3 = 0,54 (при $\alpha_1=\alpha_2=0$).

Нелинейная свертка (4.6.7) двух алгоритмов распознавания Фикса—Ходжеса с разными параметрами: $f(X) = \alpha_1 f_1(X) + + \alpha_2 f_2(X) + (1 — \alpha_1 — \alpha_2) f_1(X) f_2(X)$ дала вероятность ошибки распознавания оптимального гибрида P_{min}=0,24, тогда как каждый алгоритм в отдельности распознавал обучающую выборку со значительными ошибками: P_1 = 0,43 (при α_1=l, α_2=0) и $P_2 = = 0,46$ (при α_1=0, α_2=l). Отсюда следует, что гибрид всегда лучше, нежели лучшее из входящих в него правил, благодаря обязательному введению процедуры оптимизации параметров гибрида.

Таким образом, синтез оптимальных гибридных решающих правил распознавания дает возможность получить правила, свойства которых превышают свойства исходных правил, входящих в гибрид. Процедура синтеза гибрида заключается в выборе исходных правил R_i (i = l, ..., L) и минимизации критерия качества гибрида Q по параметрам свертки функции F, т. е. в решении задачи (4.6.8).

Однако гибрид обладает одним недостатком — неявной зависимостью решения от ситуации X. Это обстоятельство сужает возможности гибридных алгоритмов, хотя и упрощает их синтез.

Рассмотрим теперь решающие правила с явной зависимостью решения от ситуации X. В отличие от (4.6.5) они имеют вид (см. рис. 4.6.1, *в*):

$$S = F (R_1(X), R_2(X), ..., R_L(X), X). \qquad (4.6.12)$$

Здесь функция F имеет $L + n$ аргументов (n = dim X), и поэтому

ее синтез значительно труднее, чем (4.6.5). Однако и возможностей при этом больше.

Хорошие результаты при реализации (4.6.12) показала следующая эвристика, названная коллективом решающих правил [189—191]. Веса голосующих правил не детерминированы, а изменяются от ситуации к ситуации. Для этого все пространство ситуаций условно разбивается на области предпочтительности применения того или иного решающего правила и вводится зависимость веса l-го решающего правила от ситуации X в виде $\upsilon_l=\upsilon_l(X)$. Весовые коэффициенты определяются следующим образом:

$$\upsilon_l(X)=\begin{cases}1 & \text{при } X\in B_l;\\ 0 & \text{при } X\notin B_l.\end{cases}$$

Здесь $\sum_{l=1}^{L}\upsilon_l(X)=1$, а B_l — область компетентности l-го решающего правила, которая определяется как область ситуаций $\{X\}$, где данное решающее правило имеет наилучшее качество по заданному критерию Q, т. е.

$$B_l=\{X: Q(R_l(X), I)=\min_{i=1,\ldots,L} Q(R_i(X), I)\}. \quad (4.6.14)$$

Это означает, что решение коллектива в ситуации X определяется решением такого правила R_l, к области компетентности которого (B_l) эта ситуация принадлежит, т. е. $S=S_l$.

Как видно, предлагаемый подход представляет собой двухуровневую процедуру распознавания: на первом уровне осуществляется распознавание принадлежности ситуации X к той или иной области компетентности B_l (l=1,...,L), на втором — решение выделенного таким образом правила отождествляется с решением всего коллектива, т. е.

(4.6.15)

При подобной организации коллективного решения основной задачей является синтез алгоритма определения областей компетентности B_l (l=1, ..., L) решающих правил, входящих в коллектив. Естественно для членения пространства признаков на области компетентности воспользоваться методами распознавания — здесь решается L-классовая задача.

Один из алгоритмов синтеза коллектива основан на методе потенциальных функций, и его можно назвать алгоритмом восстановления потенциалов компетентности. Суммарный потенциал $\Phi_l(X)=\sum_{i=1}^{N}\varphi_l(X, X_i)$ определяет потенциал компетентности $F: S=S_l$, если $X\in B_l$.

l-го решающего правила, наводимый точками обучающей последовательности в произвольной точке X. Решение в точке X принимает правило с номером l^*, который определяется исходя из условия

$$\Phi_{l^*}(X) = \max_{l=1,\ldots,L} \Phi_l(X). \quad (4.6.16)$$

Вообще говоря, правилом выбора алгоритма может стать любое, легко решающее многоклассовую задачу, — например, правило Байеса, ближайшей точки, Фикса—Ходжеса и т. д. Каждое из этих правил может быть успешно использовано для определения областей компетентности при распознавании методом коллектива.

Надо отметить, что в результате выделения областей компетентности в коллективе наряду с компетентными правилами (у которых области компетентности велики) могут оказаться как малокомпетентные, так и некомпетентные решающие правила (область компетентности мала или вообще отсутствует). Естественно поставить задачу синтеза коллектива решающих правил, образованного компетентными правилами.

Существует немало методов поиска оптимального коллектива алгоритмов распознавания. Одним из них является алгоритм селекции, рассмотренный в работе [189], согласно которому вводятся критерии эффективности решающих правил:

1. Средний вклад l-го решающего правила, т. е. доля объектов, распознаваемых этим правилом как наиболее компетентным:

$$a_l = n_l / N, \quad (4.6.17)$$

где n_l — число объектов контрольной последовательности, для которых l-е правило наиболее компетентно; N — объем этой последовательности.

2. Вероятность правильного распознавания l-м правилом при условии, что оно является наиболее компетентным:

$$p_l = m_l / n_l. \quad (4.6.18)$$

Тогда задача селекции, т. е. задача синтеза оптимального коллектива правил, формируется как многокритериальная задача с $2L$ критериями:

$$a_l \to \max_{\{R_l\} \in V}; \quad p_l \to \max_{\{R_l\} \in V}, \quad (4.6.19)$$

где V — множество допустимых решающих правил, генерируемых для селекции. Поставленная многокритериальная задача

может быть сведена к однокритериальной путем введения обобщенного критерия вероятности ошибки распознавания коллективом, которую, естественно, нужно минимизировать в процессе синтеза коллектива:

$$P = 1 - \sum_{l=1}^{L} a_l p_l \to \min_{\{R_l\} \in V}. \quad (4.6.20)$$

Указанный критерий оптимизируется путем применения эвристического алгоритма селекции. С помощью этого алгоритма генерирование новых членов и замена малокомпетентных и некомпетентных решающих правил новыми членами улучшает качество работы коллектива. Очевидно, что указанные в работе [189] алгоритмы селекции не исчерпывают все возможности выбора разумных приемов для достижения указанных целей. Другим методом синтеза оптимального коллектива алгоритмов распознавания является оптимизация параметров решающих правил — членов коллектива — с помощью алгоритмов поиска [190]. Здесь рассматривается коллектив алгоритмов, где каждое решающее правило задается с точностью до конечного числа параметров. Тогда, очевидно, свойства коллектива решающих правил определяются параметрами решающих правил, составляющих коллектив. В пространстве этих параметров, которое естественно называть пространством коллективов, каждый конкретный коллектив отображается в точку. Для оптимизации такого коллектива алгоритмов можно воспользоваться адаптивным глобальным алгоритмом случайного поиска [166] параметров решающих правил этого коллектива. Обозначив через $C_l = (c_1^{(l)}, ..., c_{kl}^{(l)})$ — вектор параметров l-го решающего правила R_l, задачу синтеза оптимального коллектива можно записать в виде

$$Q(C) \to \min_{C \in \Omega}, \quad (4.6.21)$$

где Q — критерий эффективности коллектива, например, по (4.6.10); $C = (C_1,...,C_l)$ — вектор параметров коллектива размерностью $\prod_{l=1}^{L} k_l$, а Ω — область определения этих параметров. Очевидно, что использование предложенного параметрического метода оптимизации коллектива эффективно лишь при малых k_l и L.

Для решения этой задачи при большом числе параметров можно предложить прием структурной адаптации [78]. Его суть заключается в последовательном наращивании коллектива новыми членами — алгоритмами распознавания. Сначала коллек-

тив состоит из одного решающего правила R_1 и задача оптимизации (4.6.21) решается только по параметрам C_1 этого правила. Далее образуется коллектив из двух правил путем добавления второго решающего правила R_2 с неизвестными параметрами C_2, которые определяются путем оптимизации коллектива решающих правил по параметрам C_2. Эта процедура продолжается до тех пор, пока вероятность ошибки распознавания коллективом алгоритмов не станет достаточно малой. Как видно, здесь на каждом этапе производится оптимизация по параметрам C_i лишь одного решающего правила R_i, что значительно упрощает задачу оптимизации.

Эффективность предложенного способа структурной оптимизации, т. е. метода последовательного наращивания коллектива новыми членами, показана на модельной задаче [78]. В коллектив были объединены линейные дискриминантные функции, т. е. каждое из решающих правил задавалось n+1 параметром линейной формы. Добавление членов коллектива осуществлялось до выполнения критерия останова $P \leq 0{,}02$. Во всех десяти экспериментах этот критерий достигался коллективом, составленным из L = 3—5 алгоритмов при n=2. Для сравнения рассматривался случай наращивания коллектива случайными линейными правилами без оптимизации. В результате десятикратного эксперимента указанный критерий останова достигался, когда в коллектив были объединены в среднем одиннадцать линейных решающих правил. В другом примере на той же модельной задаче проводилась одновременная оптимизация параметров всех членов коллектива (т. е. решалась задача (4.6.21)). Значения критерия $P \leq 0{,}02$ не удалось достигнуть ни в одном эксперименте.

Приведенные результаты экспериментов указывают на преимущество метода последовательной оптимизации по сравнению с одновременной оптимизацией всех параметров коллектива или случайной генерацией новых членов коллектива без оптимизации.

Очевидно, что в коллектив кроме параметрических решающих правил могут быть объединены и непараметрические решающие правила, к которым относятся, например, алгоритм Байеса, «Кора» и др. Специфика работы с такими алгоритмами заключается в том, что ввиду отсутствия параметров они не могут быть оптимизированы параметрически. С другой стороны, параметры параметрических алгоритмов не всегда позволяют варьировать свойства этих алгоритмов в необходимых пределах. Эти обстоятельства заставляют искать новые пути создания оптимального разнообразия в коллективе таких алгоритмов. Для преодоления указанной трудности предлагается варьировать обучающую выборку [80]. Например, эту вариацию можно

осуществить путем стохастического изменения принадлежности объектов к заданным классам или путем случайного изменения координат объектов обучающей выборки в пространстве признаков. В первом случае элементы обучающей выборки случайно заменяются другими элементами того же множества. Такие преобразования характеризуются стохастической матрицей:

$$P = \| p_{ij} \|_{l \times l}, \quad . \qquad (4.6.22)$$

где / — число классов, а p_{ij} — вероятность изменения принадлежности элемента к i-му классу на принадлежность к j-му. Во втором случае производится случайное варьирование координат обучающей выборки в пространстве признаков. Затем стандартным методом обучения определяется решающее правило R_i. Очевидно, что каждое варьирование обучающей выборки позволяет получить новый алгоритм распознавания, что необходимо при организации коллектива алгоритмов.

Надо отметить, что после применения того или иного метода оптимизации полученный коллектив должен обладать некоторым разнообразием, т. е. правила, составляющие коллектив, должны быть достаточно разнообразны по своей структуре. С другой стороны, слишком большое разнообразие снижает эффективность распознавания. Поэтому естественно рассмотреть задачу синтеза коллективных и гибридных решающих правил с оптимальным разнообразием исходного множества решающих правил [81].

Для параметрических алгоритмов распознавания каждый член исходного множества решающих правил в пространстве признаков отображается в точку, а сам коллектив и гибрид — в совокупность таких точек. В этом случае разнообразие вводится в виде среднего значения расстояния между этими точками [189]. Для коллектива и гибрида непараметрических алгоритмов распознавания такой подход непригоден из-за отсутствия параметров. Поэтому для них [80] вводится другая мера разнообразия — коэффициент варьирования обучающей выборки (например, среднеквадратичное отклонение). Показано [80], что такое варьирование обучающей выборки позволяет получить разнообразие как коллектива, так и гибридов непараметрических решающих правил, а оптимизацией этого параметра варьирования можно получить коллектив и гибрид с оптимальным разнообразием алгоритмов, который минимизирует вероятность ошибки распознавания с минимальным числом членов коллектива.

В заключение отметим, что синтезированный указанным образом коллектив является весьма мощным средством распознавания ситуаций. Все перечисленные свойства коллектива и

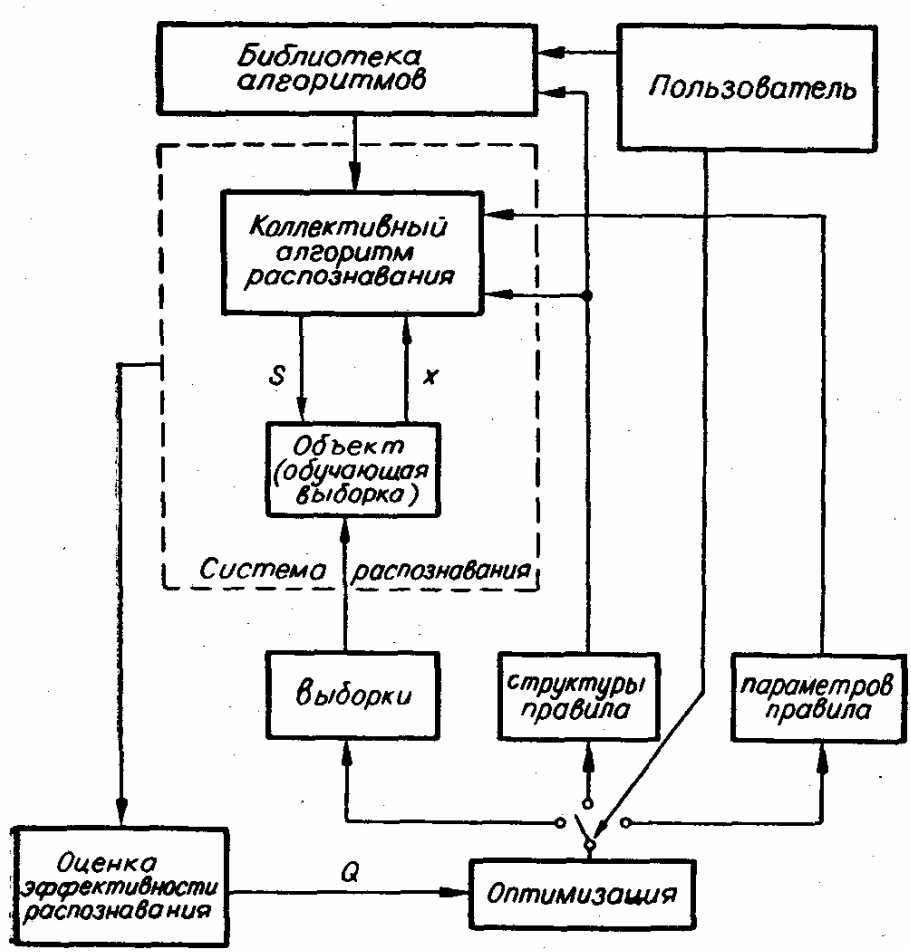


Рис. 4.6.2. Алгоритмическая блок-схема пакета программ для синтеза коллективных алгоритмов распознавания.

гибридных решающих правил получены за счет введения процедуры оптимизации, что создает реальные предпосылки для разработки специального пакета оптимизируемых программ распознавания. Его алгоритмическая структура представлена на рис. 4.6.2.

Основой пакета является библиотека алгоритмов распознавания и оптимизации. В библиотеке должны быть объединены как параметрические, так и непараметрические алгоритмы раслознавания. Признаки объектов распознавания должны иметь возможность измеряться как в метрической шкале (количественные признаки), так и в шкале наименований (качественные лризнаки). Так как пока существует мало алгоритмов распознавания для объектов со смешанными признаками, то необходимо предусмотреть алгоритмические средства для сведения подобных задач только к качественным признакам (при большом числе таких признаков) или только к количественным (при малом числе качественных признаков). В последнем случае задача сводится к синтезу нескольких решающих правил, соответствующих различным значениям качественных признаков.

В пакете необходимо предусмотреть возможность различных способов быстрой оценки критерия эффективности алгоритма, так как метод скользящего контроля [13] требует значительных затрат при большой обучающей выборке.

На пользователя следует возложить функцию выбора:

а) алгоритмов распознавания из библиотеки;

б) структуры синтезируемого решающего правила (отдель ный алгоритм, гибрид или коллектив);

в) способа варьирования правил или оптимизации (варьиро вание обучающей выборки, параметров алгоритмов и их сверток или последовательное наращивание структуры коллектива).

Сервисная часть пакета должна обеспечивать надежные оценки доверительных интервалов показателей, интересующих пользователя, и наглядное представление результатов на экране дисплея в интерактивной версии пакета.

§ 4.7. Адаптивная идентификация параметров распределения

Пусть

$$p\,(x, C) \qquad (4.7.1)$$

— заданная плотность распределения скалярной случайной величины x, где $C = (c_1, ..., c_k)$ — вектор неизвестных параметров распределения. Это означает, что плотность распределения (4.7.1) известна с точностью до параметров C, которые должны быть идентифицированы в процессе наблюдения за реализациями случайной величины x:

$$x1, x_2, ..., x_N, \qquad (4.7.2)$$

порожденными распределением (4.7.1) при $C = C^*$:

$$p\,(x, C^*), \qquad (4.7.3)$$

где C^* — искомый вектор неизвестных параметров. Таким образом, задача определения плотности (4.7.1) предполагается параметризованной.

Обычным (неадаптивным) способом решения этой задачи является известный метод максимума правдоподобия, при котором максимизируется функция правдоподобия вида

$$L_N(C) = \prod_{i=1}^{N} p(x_i, C), \qquad (4.7.4)$$

где $N>k$, или, логарифм этой функции

$$L'_N(C) = \sum_{i=1}^{N} \ln p(x_i, C). \quad (4.7,5)$$

Максимум функции правдоподобия соответствует оптимальной оценке неизвестных параметров $\hat{C}_N$ на базе наблюдений (4.7.2):

$$L'_N(\hat{C}_N) = \max_C L'_N(C) \quad (4.7.6)$$

Эта оценка, как известно [120], обладает свойствами состоятельности

$$\lim_{N\to\infty} \hat{C}_N = C^* \quad (4.7.7)$$

и асимптотической эффективности

$$D(\hat{C}_N) \to \min, \quad (4.7.8)$$

где D — знак дисперсии.

Однако задача минимизации функции (4.7.5) часто представляет собой не столько сложную, сколько громоздкую вычислительную задачу, которая обычно сводится к решению системы трансцендентных уравнений вида

$$\frac{\partial L'_N(C)}{\partial c_i} = 0 \qquad (i=1, \ldots, k). \quad (4.7.9)$$

Некоторое упрощение можно получить, если воспользоваться моментами случайной величины x. Для этого по распределению (4.7.1) определяются $m \geq k$ первых начальных моментов, которые являются функцией неизвестных параметров C:

$$M_i(C) = \int_{-\infty}^{\infty} x^i p(x, C)\, dx \qquad (i=1, \ldots, m).$$

(4.7.10)

Оценки этих же моментов можно найти по выборке (4.7.2):

$$\hat{M}_i = \frac{1}{N} \sum_{j=1}^{N} x_j{}^i. \quad (4.7.11)$$

Теперь, приравнивая эти выражения, получаем систему трансцендентных уравнений:

$$M_i(C) = \hat{M}_i \quad (i = 1, \ldots, m), \tag{4.7.12}$$

которую при $m>k$ можно решать, например, методом наименьших квадратов. Однако и этот подход сводится к чрезвычайно громоздким вычислениям (он был использован в работе [91] для случая $m = k = 2$).

Эти соображения заставляют обратиться к итерационным методам оценки неизвестных параметров распределений. Рассмотрим применение методов адаптации к решению изложенной задачи.

Исходя из свойства (4.7.7), выражение (4.7.5) при $N\to\infty$ можно записать в виде

$$\bar{L}'(C) = \lim_{N\to\infty} L'_N(C) = \int_{-\infty}^{\infty} \ln p(x, C)\, p(X, C^*)\, dx, \quad (4.7.13)$$

т. е.

$$\bar{L}'(C) = M_x[\ln p(x, c)]. \quad (4.7.14)$$

Задача оценки параметров C, обеспечивающих максимум этого выражения, теперь может быть сформулирована, например, в виде итеративного процесса адаптации [237]:

$$C_N = C_{N-1} + \Gamma_N \nabla_C \ln p(x^{(N)}, C_{N-1}), \quad (4.7.15)$$

где C_N — значение вектора определяемых параметров на N-м шаге:

$$C_N = (c_1^{(N)}, \ldots, c_k^{(N)}); \quad (4.7.16)$$

Γ_N — матрица коэффициентов $k\times k$:

$$\Gamma_N = \| \gamma_{ij}^{(N)} \|_{k\times k}; \quad (4.7.17)$$

вектор градиента:

$$\nabla_C \ln p(x^{(N)}, C) = \left(\frac{\partial}{\partial c_1} \ln p(x^{(N)}, C), \ldots, \frac{\partial}{\partial c_k} \ln p(x^{(N)}, C) \right); \quad (4.7.18)$$

$x^{(N)}$ — элемент выборки (4.7.2), получаемый путем ее циклического обхода.

Для упрощения матрицу (4.7.17) удобно представить в диагональном виде

$$\gamma_{ij}^{(N)} = \begin{cases} \gamma_i^{(N)} & \text{при } i=j; \\ 0 & \text{при } i\neq j. \end{cases} \quad (4.7.19)$$

Проиллюстрируем этот метод на примере определения параметров нормального закона распределения

$$p(x, a, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-a)^2}{2\sigma^2} \right]. \quad (4.7.20)$$

Здесь $C = (a, \sigma^2)$. Пусть сначала определяется математическое ожидание a, а дисперсия σ^2 предполагается известной. Тогда $k=1$, $c_1 = a$ и получаем

$$\nabla_a \ln p(x, a) = \frac{x-a}{\sigma^2},$$

4.7.21) причем алгоритм адаптации (4.7.15)

принимает вид

$$a_N = a_{N-1} + \gamma_N \frac{x^{(N)} - a_{N-1}}{\sigma^2}, \quad (4.7.22)$$

где a_0 — начальная априорная оценка a. При $\gamma_N = \sigma^2/N$ получаем известный алгоритм оценки среднего значения [237]:

$$a_N = a_{N-1} + \frac{1}{N}(x^{(N)} - a_{N-1}). \quad (4.7.23)$$

Теперь рассмотрим адаптацию по двум параметрам — математическому ожиданию и дисперсии. В этом случае $k = 2$; $c_1 = a$; $c_2 = \sigma^2$ и

$$\nabla \ln p(x, a, \sigma^2) = \left(\frac{x-a}{\sigma^2}, -\frac{1}{2\sigma^2} + \frac{(x-a)^2}{2\sigma^4} \right). \quad (4.7.24)$$

В результате получаем алгоритм адаптации в виде

$$a_N = a_{N-1} + \gamma_N^{(1)} \frac{x^{(N)} - a_{N-1}}{\sigma_{N-1}^2}; \quad (4.7.25)$$

$$\sigma_N^2 = \sigma_{N-1}^2 + \gamma_N^{(2)} \frac{(x^{(N)} - a_N)^2 - \sigma_{N-1}^2}{2\sigma_{N-1}^4},$$

где начальные значения a_0, σ_0 равны априорным значениям параметров a и σ. Эти выражения могут быть упрощены:

$$a_N = a_{N-1} + \gamma_N^{(3)}(x^{(N)} - a_{N-1}); \quad (4.7.26)$$
$$\sigma_N^2 = \sigma_{N-1}^2 + \gamma_N^{(4)}[(x^{(N)} - a_N)^2 - \sigma_{N-1}^2],$$

где параметры $\gamma_N^{(i)}$ (i = 1, ..., 4) выбираются в соответствии с поставленной задачей.

Так, при $\gamma_N^{(3)} = \frac{1}{N}$ и $\gamma_N^{(4)} = \frac{1}{N-1}$ получаем известные оптимальные [102] оценки математического ожидания и дисперсии:

$$a_N = \frac{1}{N}\sum_{i=1}^{N} x_i;$$

$$\sigma_N^2 = \frac{1}{N-1}\sum_{i=1}^{N} (x_i - a_N)^2. \quad (4.7.27)$$

Покажем, как применяется адаптация такого рода для оценки параметров объекта в процессе его оптимизации методом случайного поиска.

Начнем с оценки модуля градиента показателя качества Q и дисперсии помехи, накладывающейся на Q. Приращение показателя качества в процессе случайного поиска с парными пробами (3.3.5) имеет вид

$$\Delta Q' = Q'(U + g\xi) - Q'(U - g\xi), \quad (4.7.28)$$

где $U = (u_1, ..., u_q)$ — оптимизируемые параметры; g — величина пробного шага; $\xi = (\xi_1, ..., \xi_q)$ — единичный случайный вектор, равномерно распределенный в пространстве параметров $\{U\}$. Помеха накладывается аддитивным образом на показатель качества:

$$Q'(U) = Q(U) + \varepsilon(\sigma). \quad (4.7.29)$$

Здесь $\varepsilon(\sigma)$ — случайная помеха, имеющая нормальное распределение с дисперсией σ^2 и нулевым средним. Подставляя (4.7.29) в (4.7.28), получим

$$\Delta Q' = \Delta Q + \varepsilon(\sqrt{2}\sigma). \quad (4.7.30)$$

Здесь оба слагаемых являются случайными величинами. Величина ΔQ распределена при случайном поиске в линейном поле по закону [161]:

$$p(\Delta Q) = B_q \left(1 - \frac{\Delta Q^2}{k^2 g^2}\right)^{\frac{q-3}{2}}, \quad (4.7.31)$$

где q — размерность оптимизируемого объекта; $B_q = \frac{\Gamma(q/2)}{\sqrt{\pi}\Gamma\left(\frac{q-1}{2}\right)}$;

Г — гамма-функция; k — модуль градиента оптимизируемой функции:

$$k = |\nabla Q(U)|. \quad (4.7.32)$$

Помеха $\varepsilon(\sqrt{2}\sigma)$ распределена нормально с нулевым математи-
ческим ожиданием и дисперсией $2\sigma^2$:

$$p(\varepsilon) = \frac{1}{2\sqrt{\pi}\sigma} e^{-\frac{\varepsilon^2}{4\sigma^2}}. \quad (4.7.33)$$

Определим плотность распределения случайной величины $\Delta Q'$. В соответствии с выражением (4.7.30) это будет композиция двух распределений (4.7.31) и (4.7.33). Стандартным образом [47] получаем

$$p(\Delta Q') = \frac{B_q}{\sqrt{\pi}\sigma} \int_0^{kg} \left[1 - \frac{(\Delta Q' - \alpha)^2}{k^2 g^2} \right]^{\frac{q-3}{2}} e^{-\frac{\alpha^2}{4\sigma^2}} d\alpha \quad (4.7.34)$$

В простейшем случае для q=3 имеем

$$p(\Delta Q') = \frac{1}{4kg} \left[\Phi\left(\frac{kg + \Delta Q'}{2\sigma} \right) - \Phi\left(\frac{kg - \Delta Q'}{2\sigma} \right) \right], \quad (4.7.35)$$

где Ф — функция Лапласа [47].

Теперь определим составляющие градиента логарифма функции (4.7.35):

$$\frac{\partial \ln p(\Delta Q')}{\partial k} = -\frac{1}{k} - \frac{g\left[\exp\left(-\frac{kg + \Delta Q'}{2\sigma} \right) - \exp\left(-\frac{kg - \Delta Q'}{2\sigma} \right) \right]}{\sqrt{\pi}\sigma \left[\Phi\left(\frac{kg + \Delta Q'}{2\sigma} \right) - \Phi\left(\frac{kg - \Delta Q'}{2\sigma} \right) \right]}; \quad (4.7.36)$$

$$\frac{\partial \ln p(\Delta Q')}{\partial \sigma} = \frac{(kg + \Delta Q') \exp\left(-\frac{kg + \Delta Q'}{2\sigma} \right) - (kg - \Delta Q') \exp\left(-\frac{kg - \Delta Q'}{2\sigma} \right)}{2\sigma^2 \left[\Phi\left(\frac{kg + \Delta Q'}{2\sigma} \right) - \Phi\left(\frac{kg - \Delta Q'}{2\sigma} \right) \right]}$$

Алгоритм адаптивной оценки модули градиента k и среднеквадратичного отклонения а имеет вид

$$k_N = k_{N-1} + \gamma_N^{(1)} \frac{\partial}{\partial k} \ln p(\Delta Q'_N)\Big|_{\substack{k=k_{N-1} \\ \sigma=\sigma_{N-1}}}; \quad (4.7.37)$$

$$\sigma_N = \sigma_{N-1} + \gamma_N^{(2)} \frac{\partial}{\partial \sigma} \ln p(\Delta Q'_N)\Big|_{\substack{k=k_N \\ \sigma=\sigma_{N-1}}}.$$

Входящие в выражения (4.7.37) частные производные довольно громоздки. При $q \neq 3$ приведенные зависимости будут еще сложнее, а при нечетных q, возможно, и не удастся найти их явное выражение для (4.7.34), необходимое для организации процесса адаптации в указанной форме. В таких случаях для оценки частных производных (4.7.18) целесообразно воспользоваться методом Монте-Карло. Рассмотрим этот подход в общем виде.

Пусть случайная величина x является заданной функцией независимых случайных величин $y_1, ..., y_l$:

$$x = f(y_1, ..., y_l) \quad (4.7.38)$$

с заданными законами распределения

$$p_i(y_i, C^{(i)}) \quad (i = 1, ..., l), \quad (4.7.39)$$

где $C^{(i)} = (c_1^{(i)}, ..., c_{k_i}(i))$ — векторы параметров распределений (4.7.39), которые нужно определить по наблюдаемым реализациям (4.7.2) величины x.

Пусть плотность распределения x имеет по-прежнему вид (4.7.1), где, однако,

$$c_1 = c_1^{(1)}, c_2 = c_2^{(1)}, ..., c_k = c_{kl}^{(l)} \quad (4.7.40)$$

и

$$k = \sum_{i=1}^{l} k_i. \quad (4.7.41)$$

Ввиду сложности функции (4.7.38) зависимость (4.7.1) в явном виде представить нельзя. Поэтому в алгоритмы адаптивной идентификации (4.7.15) вместо градиента (4.7.18) следует ввести его оценку

(4.7.42)

$$\hat{\nabla} \ln p(x, C).$$

Таким образом, задача сводится к определению оценки (4.7.42). Именно для нее и предлагается воспользоваться методом Монте-Карло. Сделать это можно следующим образом.

Пусть надо оценить частную производную по параметру c_i.

$$\frac{\partial}{\partial c_i} \ln p(x^{(N)}, C) \quad (4.7.43)$$

Как известно,

$$p(x^{(N)}, C) = \lim_{\delta \to 0} \frac{1}{2\delta} \text{Вер} \left[x^{(N)} - \delta < x < x^{(N)} + \delta\right] \quad (4.7.44)$$

где

$$\text{Вер} \left[x^{(N)} - \delta < x < x^{(N)} + \delta\right] = \lim_{T \to \infty} \frac{t}{T} \quad (4.7.45)$$

T — объем «разыгрываний» случайной величины x; $t = t(\delta, C)$ — число попаданий x в промежуток $[x^{(N)} - \delta, x^{(N)} + \delta]$ при параметрах C. Ограничиваясь конечным значением T, можно для (4.7.44) получить следующую приближенную оценку:

$$\hat{p}(x^{(N)}, C) = \frac{t(\delta, C)}{2\delta T}. \quad (4.7.46)$$

Выбор величины промежутка δ должен быть оптимальным. Действительно, δ не должна быть слишком малой, так как тогда t будет мало, оценка (4.7.46) будет слишком грубой и понадобится значительно увеличить объем T, чтобы уточнить оценку. С другой стороны, величина δ не должна быть слишком большой, иначе оценка будет носить не локальный характер.

Теперь частные производные (4.7.46) можно представить в виде приближенных выражений:

$$\frac{\partial}{\partial c_i} \ln p(x^{(N)}, C) \approx \frac{1}{2\Delta_i} \ln \frac{t(\delta, C + \Delta_i e_i)}{t(\delta, C - \Delta_i e_i)}, \quad (4.7.47)$$

где e_i ($i = 1, \ldots, k$) — орт i-й координаты пространства искомых параметров $\{C\}$. Здесь $t(\delta, C \pm \Delta_i e_i)$ обозначает число попаданий случайной величины x в зону $x^{(N)} - \delta < x < x^{(N)} + \delta$ при параметрах C, варьированных на величину $\pm\Delta_i$ по параметру c_i, т. е. $c_i \to c_i \pm \Delta_i$.

Выбор величины Δ_i, как и δ, должен быть оптимальным — как при всяком численном дифференцировании.

Любопытно, что оценка (4.7.47) не зависит явно от объема испытаний T. Однако этот объем должен быть одинаков для оценки t при обеих вариациях параметров $\pm\Delta_i$.

Предложенный метод может применяться для идентификации параметров оптимизируемого объекта и как средство для организации оптимального поиска. Такими параметрами объекта, необходимыми для построения оптимальных алгоритмов поиска экстремума, являются модуль градиента показателя качества, дисперсия помехи, кривизна гиперповерхностей равного уровня показателя качества и др.

§ 4.8. Адаптивный синтез датчика случайных чисел с заданной автокорреляционной функцией

Задача синтеза датчика случайных процессов с заданными корреляционными свойствами представляет собой сложную вычислительную проблему [207]. Применение адаптивного подхода позволяет построить довольно простой алгоритм синтеза такого датчика, который легко реализуется на ЦВМ [162].

Рассмотрим центрированные случайные процессы в дискретном времени, получаемые при помощи следующего линейного рекуррентного выражения:

$$x_t = \sum_{i=1}^{n} p_i x_{t-i} + \sum_{i=1}^{m} a_i \varepsilon_{t-i+1}. \quad (4.8.1)$$

Здесь x_t — значение случайной функции x в момент t; ε_t — независимые случайные числа с нулевым математическим ожиданием и единичной дисперсией. Простейший частный случай выражения. (4.8.1) при $n = m = 1$ рассмотрен и подробно проанализирован в работе [160]. Автокорреляционная функция процесса (4.8.1)

$$K(q) = M_t(x_t x_{t-q}), \quad (4.8.2)$$

где M_t — знак математического ожидания, очевидно, зависит от параметров $p_1, \ldots, p_n, a_1, \ldots, a_m$ датчика. Задача синтеза случайного процесса с заданными корреляционными свойствами заключается в определении указанных параметров так, чтобы полученная автокорреляционная функция $K(q)$ наименьшим образом уклонялась от заданной $K^0(q)$.

В качестве меры этого уклонения (невязки) могут быть предложены различные функции, например

$$Q = \sum_{q=1}^{\infty} F(K(q) - K^0(q)). \quad (4.8.3)$$

где F — четная положительная функция, или

$$Q=\max_{q} F(K(q)-K^0(q)) \quad (4.8.4)$$

Очевидно, что эта функция зависит от параметров датчика, которые следует подобрать так, чтобы минимизировать невязку Q, т. е. решить задачу

$$Q(P, A) \to \min_{P,A},$$

4.8.5) где

$$P = (p_1, ..., p_n); \quad (4.8.6)$$
$$A = (a_1, ..., a_m).$$

Заметим, что каждое определение минимизируемой функции Q (P, A) по формуле (4.8.3) или (4.8.4) всегда связано с вычислением корреляционной функции (4.8.2), что крайне трудоемко. Так, в работе [243] эта задача решалась аппаратурно с применением автоматического оптимизатора, причем основные потери были связаны с оценкой автокорреляционной функции при заданных параметрах P и A.

Однако в случае линейной зависимости (4.8.1) удается получить явные выражения для корреляционной функции $K(q)$ (4.8.2) в зависимости от параметров P и A. Для этого достаточно решить некоторую систему линейных алгебраических уравнений, вывод которой приведен ниже.

Используя выражение (4.8.2) и предполагая, что $K(0) = = D(x) = 1$; $D(\varepsilon) = 0$; $M(\varepsilon_i\varepsilon_j)=0$ ($i\neq j$), получаем после подстановки (4.8.1) в (4.8.2) и ряда преобразований следующее рекуррентное выражение для автокорреляционной функции процесса (4.8.1):

$$K(q)=\sum_{i=1}^{n} p_i K(q-i)+\sum_{j=1}^{m} a_j l(j-q-1), \quad (4.8.7)$$

где $l(t)$ определяется также рекуррентно:

$$l(t)=\begin{cases} 0_n \text{ для } t<0; \\ \sum_{i=1} p_i l(t-i)+a_{t+1} \text{ для } t\geq 0, \end{cases} \quad (4.8.8)$$

причем $a_{m+t} = 0$ для $t>0$. Как видно, эти формулы для заданных P и A дают возможность довольно просто вычислить все необходимые значения автокорреляционной функции $K(q)$ и

далее определить степень ее невязки Q с заданной функцией $K^0(q)$ по формуле (4.8.3) или (4.8.4).

Чтобы начать этот рекуррентный процесс определения $K(q)$, необходимо знать n начальных значений автокорреляционной функции, из которых известно только одно $K(0) = 1$. Для этого достаточно решить линейную систему уравнений (n-1)-го порядка относительно $K(1)$, ..., $K(n—1)$, которая получается из (4.8.7) при $q = 1, 2, ..., n$-1. Автокорреляционную функцию естественно считать четной, т. е. $K(q) = K(—q)$.

На выбор параметров P и A накладывается ряд ограничений. Одно из них связано с единичной дисперсией генерируемого процесса и выражается равенством

$$K(0) = \sum_{i=1}^{n} p_i K(i) + \sum_{j=1}^{m} a_j l(j-1) = 1. \quad (4.8.9)$$

Другие ограничения накладываются на выбор параметров и выражаются в виде очевидных неравенств:

$$| K(q) | \leq 1 \ (q = 1, 2, ...), \quad (4.8.10)$$

которые обеспечивают устойчивость генерируемого случайного процесса (4.8.1). Это условие эквивалентно выполнению критерия Гурвица для разностных уравнений [235]. В процессе поиска значения искомых параметров, которые приводят к нарушению указанных ограничений, следует отбрасывать.

Как легко видеть, определение корреляционной функции $K(q)$ при любом q представляет собой следующую процедуру.

1. Для заданных значений P и A решается линейная система уравнений (4.8.7) при $q = 1, ..., n$-1, из которой определяются

$K(1), ..., K(n\text{-}1)$.

2. Последующие значения $K(q)$ при $q \geq n$ определяются рекуррентно из системы (4.8.7).

Решение задачи (4.8.5) определения параметров P и A теперь представляется в виде

$$Q(P, A) \to \min_{P,A \in S} \Rightarrow P^*, A^*, \quad (4.8.11).$$

где ограничения S образуются равенством (4.8.9) и неравенствами (4.8.10). Решение задачи (4.8.11) и дает искомые значения параметров

$$P^* = (p^*_1, ..., p^*_n); \quad (4.8.12)$$
$$A^* = (a^*_1, ..., a^*_m)$$

датчика (4.8.1), автокорреляционная функция которого $K(q)$ наименьшим образом отличается от заданной $K^0(q)$.

Задача (4.8.11) имеет $n+m$ переменных, что при больших n затрудняет процесс поиска оптимума. Число неизвестных переменных можно сократить до m в случае, когда автокорреляционная функция $K^0(q)$ задана точно (а не получена в результате наблюдений). Делается это следующим образом.

Формула (4.8.7), как можно заметить, при $q \geq m$ принимает простой вид:

$$K(q) = \sum_{i=1}^{n} p_i K(q-i). \quad (4.8.13)$$

Очевидно, что для определения P достаточно положить $K(q) = = K^0(q)$ и взять n реализаций этой формулы при $q = m, \ldots, m$ — — n—1. Получаем систему уравнений, линейных относительно неизвестных $p_1, \ldots, p_n$:

$$K^0(m) = \sum_{i=1}^{n} p_i K(m-i); \quad (4.8.14)$$

. . . . . .

$$K^0(m+n-1) = \sum_{i=1}^{n} p_i K^0(m+n-1-i).$$

Здесь предполагается, что $K^0(-q) = K^0(q)$.

Разрешая эту систему, получим P. Этим число неизвестных параметров сократится до m: $a_1, \ldots, a_m$. Однако если исходная функция $K^0(q)$ получена приближенно — например, в результате соответствующей обработки конечной реализации случайного процесса, то этим приемом воспользоваться нельзя и поиск следует производить по всем переменным.

Рассмотрим теперь несколько наиболее распространенных случаев.

Случай $n = 1$, $m = 2$. Формула (4.8.1) принимает вид

$$x_t = px_{t-1} + a_1\varepsilon_t + a_2\varepsilon_{t-1}.$$

Корреляционная функция (4.8.7):

$$K(q) = pK(q+1) + \begin{cases} a_1 a_2 & \text{для } q=1; \\ 0 & \text{для } q \geqslant 2. \end{cases}$$

Условие устойчивости: $|p|<1$. Параметры p, a_1, a_2 связаны следующим очевидным соотношением:

$$p^2 + a_1^2 + a_2^2 + pa_1a_2 = 1.$$

На рис. 4.8.1 показан характер поведения полученной корреляционной функции при различных значениях параметра a_2. Из

этих графиков хорошо видно, что, варьируя значения a_2, можно достаточно произвольно изменять значение $K(1)$. Последующее поведение корреляционной функции при $q>0$ имеет экспоненциальный характер.

С л у ч а й $n=1$, $m>1$. Нетрудно убедиться, что в этом случае экспоненциальный характер функции автокорреляции проявляется лишь при $q \geq m$. Ее значения при $0<q<m$ могут быть, вообще говоря, достаточно произвольными, но в естественных пределах, обеспечивающих $|K(q)| \leq 1$. Это обстоятельство позволяет синтезировать широкий класс функций с экспоненциальным «хвостом».

Случай $n=2$, $m=1$ приводит к следующей рекуррентной формуле для корреляционной функции ($q>0$):

$$K(q) = p_1 K(q-1) + p_2 K(q-2)$$

с начальными значениями

$$K(0) = a^2 = 1; \quad K(1) = \frac{p_1}{1-p_2}.$$

Из естественных условий $|K(1)| \leq 1$ и $|K(2)| \leq 1$ получаем уравнения для допустимой области параметров, обеспечивающих устойчивость генерируемого случайного процесса:

$$p_1 + p_2 \leq 1;$$

$$p_1 - p_2 \geq 1;$$

$$p_2 \geq -1.$$

На плоскости параметров (рис. 4.8.2) эта область имеет форму равнобедренного треугольника, каждая точка которого опреде-

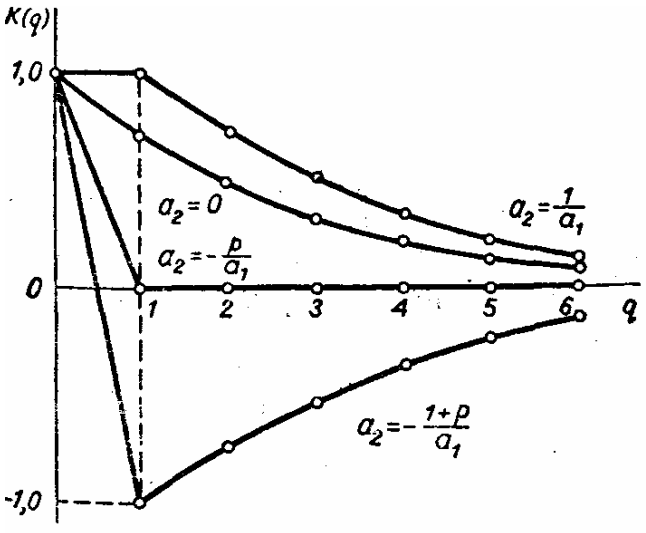


Рис. 4.8.1. Поведение корреляционной Функции для случая $n=1$, $m=2$.

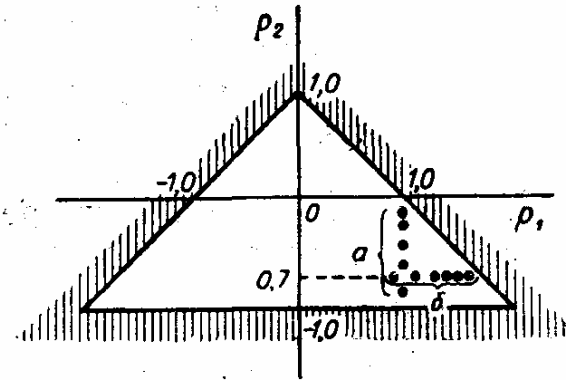


Рис. 4.8.2. Область допустимых параметров ($n=2$, $m=1$). Наборы точек «*а*» и «*б*» соответствуют кривым на рис. 4.8.3.

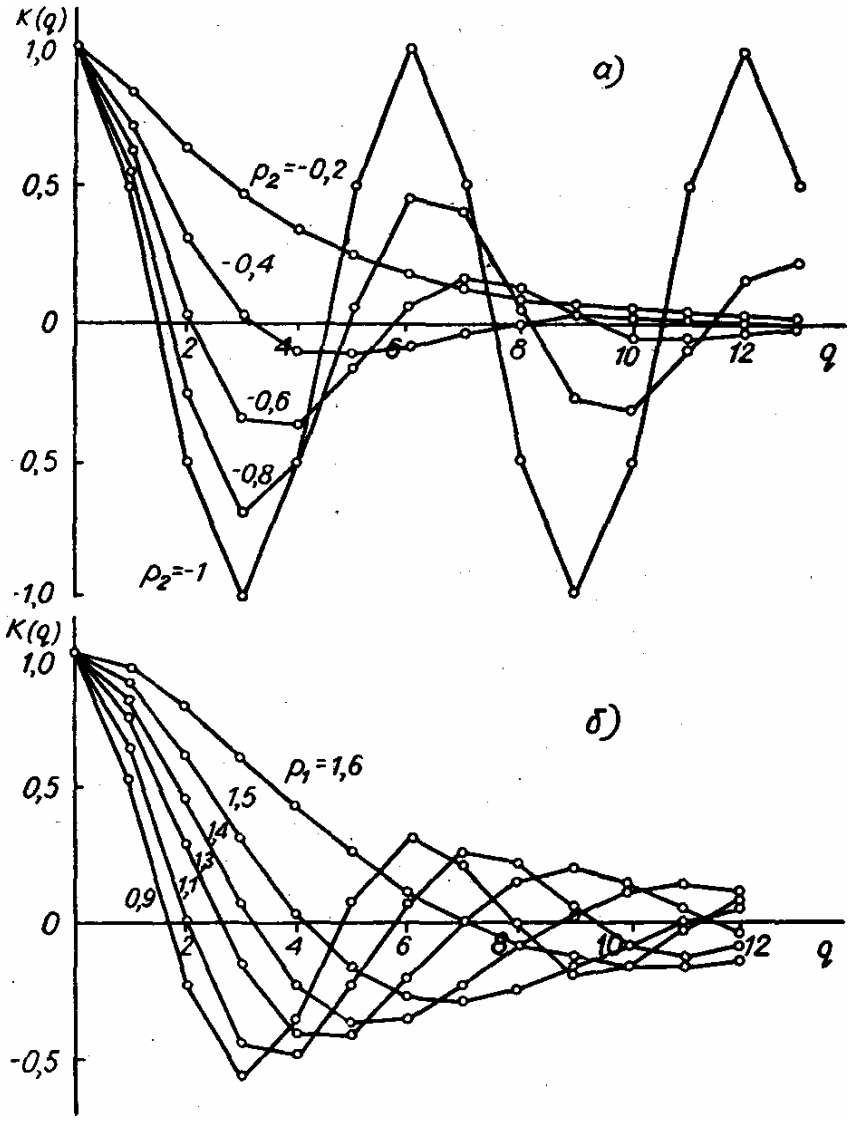


Рис. 4.8.3. Характер поведения корреляционных функций при значениях параметров, отмеченных точками на рис. 4.8.2: *а* — p_1=1, *б* — p_2= —0,7.

ляет корреляционную функцию. Характер поведения этих функций показан на рис. 4.8.3 для значений параметров, отмеченных на рис. 4.8.2 точками. Из рисунков хорошо видно, что рассматриваемый случай весьма богат разнообразием корреляционных функций.

Случай n=m=2. Рекуррентная формула при $q \geq 0$ имеет тот же вид, что и для предыдущего случая. Однако

$$K(1) = \frac{p_1 + a_1 a_2}{1 - p_2}$$

Следовательно, не изменяя характера поведения корреляционной функции при $q \geq 2$, можно, варьируя произведение $a_1 a_2$, несколько ее деформировать.

Проиллюстрируем это на следующем простом примере. Для реализации случайного процесса с корреляционной функцией вида

$$K^0(q) = e^{-\alpha q} \cos \omega q$$

следует воспользоваться рассмотренным случаем (n=m=2). Так как затухание α и частота ω корреляционной функции не зависят от начальных значений функции, то эти параметры могут быть определены при a_2 = 0, т. е. так же, как и в предыдущем случае (n=2, m=l).

На рис. 4.8.4 показаны корреляционные функции с неизменными полученными значениями параметров p_1 = l,5; p_2 = —0,7 и различными a_2. Хорошо видно, что затухание и частота при этом не изменяются.

Выбор оптимального значения a_2 производится путем минимизации функции невязки

$$Q = \sum_{q=1}^{N} |e^{-\alpha q} \cos \omega q - K(q)|,$$

которая зависит лишь от одного параметра K(1).

На рис. 4.8.5 показан характер поведения этой функции для значений α=0,184 и ω = 0,482, которые обеспечиваются p_1 = = 1,5 и p_2 = —0,7 при N=20. Хорошо виден экстремальный характер этой зависимости. Следует отметить ненулевое значение невязки в экстремальной точке K(1) =0,747, хотя в принципе, как показано в работе [207], экстремум равен нулю. Это объясняется ограниченной точностью расчетов (не превышавшей третьего знака после запятой), что и дало в сумме минимальное значение невязки $Q_{\min}$ = 0,091.

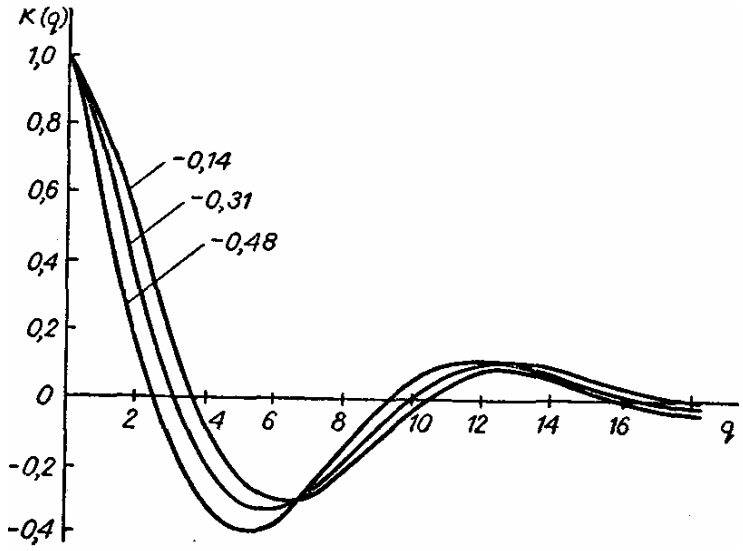


Рис. 4.8.4. Поведение корреляционных функций (n=m=2) при изменении a_1a_2 (значения указаны на рисунке).

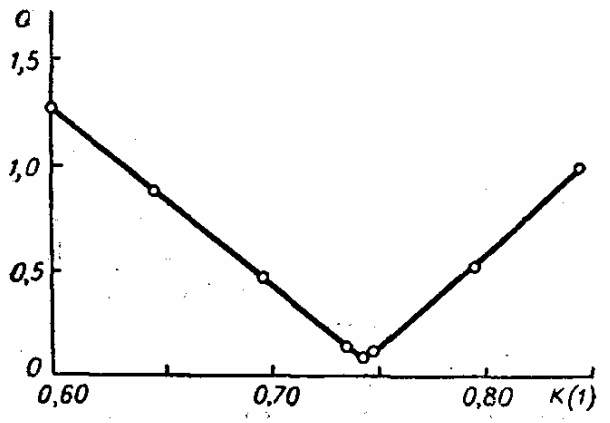


Рис. 4.8.5. Экстремальная зависимость функции невязки от $K(1)$.

В заключение отметим, что предлагаемый способ синтеза рекуррентной зависимости (4.8.1) может быть использован как метод кодирования встречающихся корреляционных функций. Действительно, в пространстве $\{P, A\}$ каждая точка определяет некоторую корреляционную функцию $K(q)$. При заданной невязке $Q(P,A)$ для любой корреляционной функции $K^0(q)$ можно найти вектор (P^*, A^*), минимизирующий эту невязку. Параметры этого вектора и определяют код функции $K(q)$.

Другой особенностью предлагаемого метода является простота его программирования на ЦВМ. Действительно, по сути дела, определение необходимых параметров сводится к многократному решению системы линейных алгебраических уравнений и к процессу многопараметрической оптимизации, что очень просто реализуется путем применения соответствующих стандартных программ.

Параметрическая адаптация является наиболее разработанным инструментом адаптация — как в алгоритмическом, так и в прикладном плане. Однако ее ограниченность, связанная с тем, что далеко не все объекты удается «разъять» до параметров, заставляет искать новые типы адаптации, которым посвящены две последующие главы.

Глава 5

Альтернативная адаптация

Быть иль не быть, вот в чем вопрос...
Шекспир. «Гамлет»

Когда число варьируемых состояний объекта невелико, следует обращаться к альтернативной адаптации, которая позволяет поддерживать в объекте оптимальное состояние. При этом естественно воспользоваться автоматными моделями адаптации. В данной главе рассматриваются два алгоритма альтернативной адаптации — детерминированные автоматы с целесообразным поведением и стохастические автоматы с переменной структурой. Эти автоматы применяются для переключения алгоритмов при решении задач адаптивной поисковой оптимизации, адаптивной сортировки массивов, адаптивного выбора способа кодирования при передаче данных по каналу связи и адаптивного определения длины информационной части пакета в канале связи двух ЭВМ.

§ 5.1. Алгоритмы альтернативной адаптации

5.1.1. Постановка задачи

Пусть 5 — множество возможных альтернатив, допустимых в процессе адаптации

$S = \{U_1, ..., U_q\}$,

где q — число таких альтернатив. Задача адаптации (1.2.9) (см. первую главу) в данном случае заключается в указании, какую альтернативу следует реализовать в сложившейся ситуации в среде X и объекте Y. Эта задача решается правилом (алгоритмом) R:

$$U_j = R(X, Y, U_i), \qquad (5.1.1)$$

где U_i, $U_j \in S$. Проще говоря, алгоритм адаптации является, по сути дела, управляемым переключателем (рис. 5.1.1), с помощью которого в объекте адаптации поддерживается та альтер-

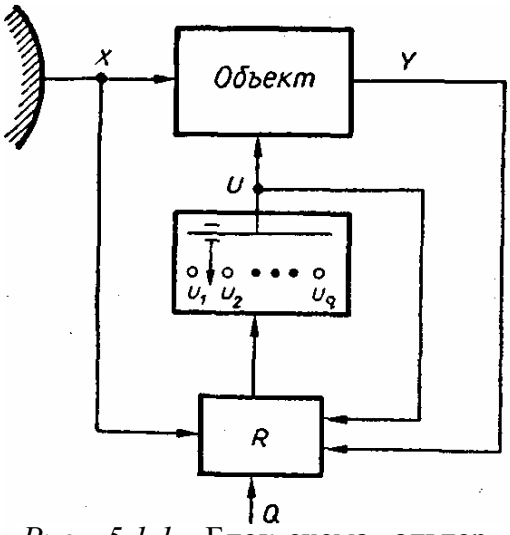


Рис. 5.1.1. Блок-схема альтернативной адаптации.

натива, которая минимизирует заданный критерий качества Q объекта в сложившейся ситуации.

Решающее правило R (5.1.1) должно решать задачу альтернативной адаптации

$$Q(X, Y) \to \min_{U \in S}$$

по локальным наблюдениям оценок критерия

$$Q_1, \ldots, Q_N, \ldots, \quad (5.1.3)$$

причем состояние Y объекта, как обычно, зависит от управления U неизвестным образом. Это обстоятельство позволяет записать задачу (5.1.2) в форме

$$Q(U) \to \min_{U \in S}, \quad (5.1.4)$$

где зависимость $Q(U)$ неизвестна.

Ввиду конечности числа альтернатив для синтеза алгоритма адаптации R естественно обратиться к конечным автоматам, алфавит выхода которых состоит из альтернативных управлений S. Здесь эффективно могут быть использованы автоматы с целесообразным поведением [134] и стохастические автоматы с переменной структурой [46]. Рассмотрим их применение для альтернативной адаптации.

5.1.2. Алгоритмы-автоматы

Прежде всего введем понятие штрафа c, используемое в теории обучения автоматов. Будем считать, что входом автомата является сигнал, характеризующий эффективность его функционирования в данный момент времени. Этот сигнал имеет двоичный характер:

$$c = \begin{cases} 0 - \text{«нештраф»}; \\ 1 - \text{«штраф»}, \end{cases} \quad (5.1.5)$$

где c=0 соответствует позитивной реакции среды (объекта), а c=1 — негативной.

Естественно связать этот штраф с изменением минимизирумого критерия Q функционирования объекта. Это можно сделать следующим образом:

$$c_N = \begin{cases} 0 \text{ при } \Delta\hat{Q}_N < 0, \\ 1 \text{ при } \Delta\hat{Q}_N \geqslant 0, \end{cases} \quad (5.1.6)$$

где

$$\Delta\hat{Q}_N = \hat{Q}(U'_N) - \hat{Q}(U'_{N-1}), \quad (5.1.7)$$

а U'_N — управление, реализованное в объекте на N-м шаге

$$U'_N \in S.$$

Так как оценка приращения критерия (5.1.7) происходит в обстановке значительной неопределенности о состоянии среды и оператора объекта, влияющих на эту оценку, то удобно воспользоваться рекуррентным сглаживанием:

$$\bar{Q}_N = k\bar{Q}_{N-1} + (1-k)\hat{Q}_N, \quad (5.1.8),$$

где $0<k<1$ — коэффициент сглаживания.

В этом случае оценка (5.1.7) принимает вид

$$\Delta\hat{Q}_N = \hat{Q}(U'_N) - \bar{Q}_{N-1}. \quad (5.1.9)$$

Теперь рассмотрим автомат с целесообразным поведением. Будем называть его поведение целесообразным, если средний штраф при функционировании автомата меньше половины, т. е.

$$M_c < \frac{1}{2}. \quad (5.1.10)$$

Иначе говоря, автомат за свои действия штрафуется (c=1) реже, чем поощряется (c=0), что, очевидно, и характеризует целесообразность его поведения. В терминах адаптации (5.1.10) означает, что оценка приращения критерия качества чаще отрицательна, чем положительна.

Заметим, что это определение целесообразности не более чем эвристика и можно легко представить ситуацию, когда при выполнении условия (5.1.10) поведение будет нецелесообразным — например, когда положительные приращения критерия (5.1.7) по модулю значительно больше отрицательных. Именно такая ситуация имеет место в окрестности экстремума критерия, что обычно затрудняет реализацию точной адаптации и требует введения специальных мер типа увеличения объема накопления и т. д. Однако в большинстве случаев эвристика (5.1.10) работает вполне эффективно, чем мы и воспользуемся.

Рассмотрим алгоритм адаптации как автомат, т. е. пятерку вида

$$A = \langle C, W, S, \mu, \upsilon \rangle. \quad (5.1.11)$$

Здесь C — алфавит входов (это двоичный сигнал штрафа c (5.1.5)); S — алфавит выходов автомата, который образуется заданными альтернативами; W — множество состояний автомата:

$$W = \{w_1, ..., w_m\}; \quad (5.1.12)$$

μ — функция переходов от одного состояния к другому:

$$w' = \mu (w, c), \quad (5.1.13)$$

где w' — новое состояние, в которое переходит автомат из состояния w при входе c; υ - функция выходов, определяющая выход автомата по его состоянию w и входу c:

$$U = \upsilon(w, c) \in S. \quad (5.1.14)$$

Таким образом, для определения автоматного алгоритма адаптации объекта необходимо знать:

1) множество состояний (5.1.12);
2) функцию переходов (5.1.13);
3) функцию выходов (5.1.14).

Разные способы задания этих факторов и отличают различные автоматные алгоритмы адаптации. Рассмотрим два из них.

5.1.2.1. Автоматы с целесообразным поведением

Из всех многочисленных автоматов с целесообразным поведением [46, 134] наибольший интерес для альтернативной адаптации представляют, так называемые автоматы с линейной тактикой. Они отличаются тем, что действие, приведшее к положительному результату (нештрафу, c=0), закрепляется автоматом и повторяется, а действие, приведшее к отрицательному результату (штрафу, c=1), автомат «стремится» сменить на другое. Так обычно поступают живые существа. В этом и состоит линейность тактики.

Введем l — параметр глубины памяти автомата. Каждое из q действий автомата (альтернативных решений) имеет l состояний памяти. Тогда число состояний автомата (5.1.12) $m=ql$.

Функцию переходов автомата с линейной тактикой удобно представить в виде двух графов переходов (рис. 5.1.2) для двух состояний входа автомата. Функция выхода этого автомата образуется так, что на различных «усах» графа переходов (ем.

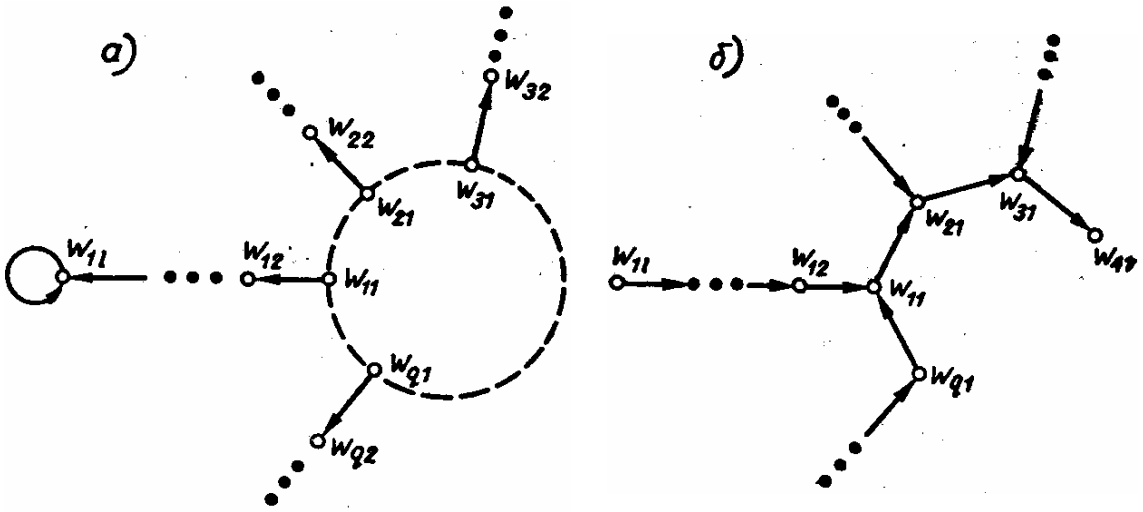


Рис. 5.1.2. Графы переходов состояний автомата с линейной тактикой: *а* — при с=0, б — при с=1.

рис. 5.1.2) производятся различные действия, т. е. действие U_i производится при $w = w_{ij}$.

Таким образом, w_{ij} является состоянием автомата, в котором он производит i-е действие U_i.

Легко видеть, что при нештрафе (c=0) последнее действие повторяется и закрепляется, а при штрафе (c=1) внутреннее состояние автомата изменяется так, чтобы быстрее сменить это действие на другое. Доказано [46, 134], что в стационарной среде, т. е. при неизменных вероятностях штрафа за каждое действие, этот автомат асимптотически оптимален:

$$\lim_{l\to\infty} Mc = a_{\min}, \tag{5.1.15}$$

где M — знак математического ожидания; $a_{\min}$ — минимальная вероятность штрафа:

$$a_{\min} = \min_{i=1,\ldots,q} a_i; \tag{5.1.16}$$

a_i — вероятность штрафа за i-е действие автомата $(a_i = \operatorname{const}_t)$

Асимптотическая оптимальность означает, что в стационарной среде при увеличении объема памяти l автомат всегда будет совершать наилучшее действие, минимизирующее его штраф.

Очевидно, что в нестационарной среде нельзя объем памяти l делать слишком большим, так как при этом затрудняется перестройка с одного действия автомата на другое. Чем более нестационарна среда, тем меньше должен быть параметр па мяти l.

Таким образом, при использовании автомата с линейной тактикой для целей альтернативной адаптации эффективность

процесса определяется одним параметром, оптимальный выбор которого обычно представляет некоторую трудность.

Рассмотренный автомат является детерминированным. Далее перейдем к стохастическим автоматам.

5.1.2.2. Стохастические автоматы с переменной структурой

Простейшим стохастическим автоматом с переменной структурой является «автомат-строка», реализующий независимый выбор альтернатив с вероятностями

$$P_N = (p_1^N, ..., p_q^N), \qquad (5.1.17)$$

$$\sum_{i=1}^{q} p_i^N = 1.$$

где

Процесс изменения структуры (обучения) такого автомата заключается в изменении вероятностей P на каждом шаге таким образом, чтобы вероятность альтернативы при нештрафе увеличивалась, а при штрафе — уменьшалась с учетом нормирования вероятностей:

$$p_i^N = p_i^{N-1} + \Delta p_i^N \quad (i = 1, ..., q), \qquad (5.1.18)$$

где

$$\Delta p_i^N = \begin{cases} f(p_i^{N-1}) & \text{при } U'_N = U_i; \\ \varphi(p_i^{N-1}, p_j^{N-1}) & \text{при } U'_N = U_j, \quad (j \neq i), \end{cases} \qquad (5.1.19)$$

причем

$f(\bullet)>0$; $\varphi(\bullet,\bullet)<0$ при $c = 0$;

$f(\bullet)<0$; $\varphi(\bullet,\bullet)>0$ при $c = 1$. (5.1.20)

На функции f и φ накладываются очевидные ограничения:

$$0 \le z + f(z) \le 1;$$

$$0 \le z + \varphi(z, w) \le 1; \qquad (5.1.21)$$

$$f(z_1) + \sum_{i=2}^{q} \varphi(z_1, z_i) = 1$$

для любого $0 \le z \le 1$ и $\sum_{i=1}^{q} z_i = 1$ при $0 \le z_i \le 1$. Этим условиям

удовлетворяют, например, такие функции:

$$f(z)=\begin{cases}\alpha(1-z)^{\beta} & \text{при } c=0;\\ -\alpha z^{\beta} & \text{при } c=1\end{cases} \tag{5.1.22}$$

и

$$\varphi(z_i, z_j) = \begin{cases} -\alpha \dfrac{z_j^{\beta}(1-z_i)^{\gamma}}{\sum\limits_{k\neq j}^{q}(1-z_k)^{\gamma}} & \text{при } c=0; \\ \alpha \dfrac{(1-z_j)^{\beta} z_i^{\gamma}}{\sum\limits_{k\neq j}^{q} z_k^{\gamma}} & \text{при } c=1, \end{cases} \qquad (5.1.23)$$

где всего имеется три параметра:

$0<\alpha<1;\ \beta,\gamma>0.$ (5.1.24)

В простейшем случае $\beta = \gamma = 1$.

5.1.2.3. Алгоритм «многорукого бандита»

Известная задача о «двуруком бандите» [256, 263]* может быть интерпретирована как задача о двуальтернативной адаптации. Для простоты рассмотрим сначала этот случай ($q = 2$).

Будем сочетать две тактики поведения, рассмотренные выше. При $c = 0$ сохраним удачную альтернативу, т. е. будем действовать в соответствии с линейной тактикой (рис. 5.1.3, *а*), а при $c = 1$ сохраним первую альтернативу с вероятностью p, а вторую — с вероятностью $1—p$ и перейдем к другой альтернативе с дополняющими вероятностями (рис. 5.1.3, *б*).

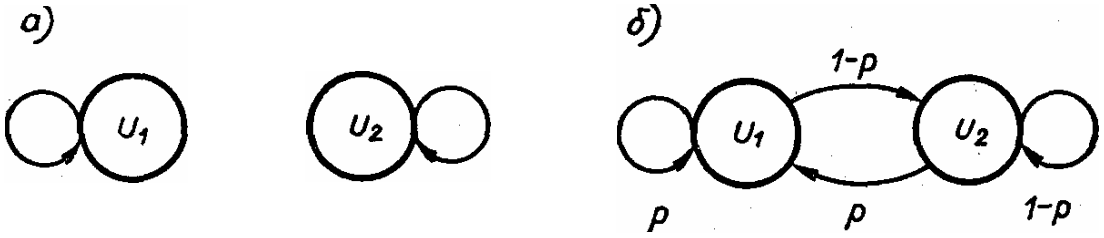


Рис. 5.1.3. Граф алгоритма двуальтернативного выбора: *а* — при с=0, *б* — при с=1.

* «Одноруким бандитом» американцы называют игральный автомат, приводимый в действие одной рукояткой («рукой»). Задача о «двуруком бандите» возникает при двух рукоятках, если априори известно, что вероятность выигрыша при запуске автомата какой-то одной «рукой» больше, чем другой. Задача состоит в том, чтобы, манипулируя обеими рукоятками, найти ту, которая обеспечивает наибольший выигрыш, и при этом свести к минимуму про-

Задача заключается в определении и изменении вероятности p в процессе адаптации. Очевидно, что при $U_1 \succ U_2$ вероятность p должна возрастать, а при $U_1 \prec U_2$ — уменьшаться.

Для оценки p введем функцию риска [250—253]. В общем случае ее логично записать в виде

$$W = \sum_{i=1}^{q} (p^*_i - p_i)^2 \upsilon_i. \quad (5.1.25)$$

Здесь p^*_i — вероятность того, что i-я альтернатива является наилучшей, т. е.

$$p^*_i = \text{Вер}\,[U_i = U^*] \quad (i = 1, \ldots, q) \quad (5.1.26)$$

где U^* — оптимальная альтернатива; p_i — вероятность использования i-й альтернативы алгоритмом адаптации, а υ_i — ущерб, испытываемый при использовании; i-й альтернативы, если она неоптимальна.

Задача синтеза оптимальной стратегии поведения сводится к решению задачи минимизации

$$\sum_{i=1}^{q} (p^*_i - p_i)^2 \upsilon_i \to \min_{P \in S} \Rightarrow P^* \quad (5.1.27)$$

где S: $\sum_{i=1}^{q} p_i = 1$

Отсюда вытекает, что при $\upsilon_i \neq 0$ оптимальная стратегия имеет вид (5.1.26), т. е.

$$P^* = (p^*_1, \ldots, p^*_q). \quad (5.1.28)$$

Таким образом, оптимальной является рандомизированная стратегия, которая получается путем оценивания вероятностей (5.1.26). Проведем оценивание для $q = 2$.

Естественно предположить, что

$$\hat{Q}(U_1) = Q_1 + \varepsilon(\sigma_1);$$
$$\hat{Q}(U_2) = Q_2 + \varepsilon(\sigma_2), \quad (5.1.29)$$

где Q_1 и Q_2 — средние значения показателя качества для каждой альтернативы соответственно, а $\varepsilon(\sigma)$ — реализация независимых случайных величин с нулевым математическим ожиданием и дисперсией а, априори неизвестной. Будем считать, что распределение этих случайных величин нормальное. Тогда вероятность того, что альтернатива U_1 лучше U_2:

$$p = \text{Вер}\,[U_1 \succ U_2] = \text{Вер}\,[\hat{Q}(U_1) < \hat{Q}(U_2)] =$$
$$= \frac{1}{2}\left[1 + \Phi\left(\frac{\hat{Q}_1 - \hat{Q}_2}{2\sqrt{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}}\right)\right], \quad (5.1.30)$$

где Ф — функция Лапласа, а «крышечкой» обозначены оценки, которые определяются рекуррентно на каждом шаге адаптации:

$$\hat{Q}_{1,N+1}=\begin{cases}\hat{Q}_{1,N}, & \text{если } U'_{N+1}=U_2;\\ \mu\hat{Q}_{1,N}+(1-\mu)\hat{Q}(U_1), & \text{если } U'_{N+1}=U_1;\end{cases} \quad (5.1.31)$$

$$\hat{Q}_{2,N+1}=\begin{cases}\hat{Q}_{2,N}, & \text{если } U'_{N+1}=U_1;\\ \mu\hat{Q}_{2,N}+(1-\mu)\hat{Q}(U_2), & \text{если } U'_{N+1}=U_2.\end{cases} \quad (5.1.32)$$

Оценки

$$\hat{\sigma}^2_{1,N+1}=\begin{cases}\hat{\sigma}_{1,N}^2, & \text{если } U'_{N+1}=U_2;\\ \mu^2\hat{\sigma}_{1,N}^2+(1-\mu)^2(\hat{Q}_{1,N+1}-\hat{Q}(U_1))^2 & \text{при } U'_{N+1}=U_1;\end{cases} \quad (5.1.33)$$

дисперсий:

$$\hat{\sigma}^2_{2,N+1}=\begin{cases}\hat{\sigma}_{2,N}^2, & \text{если } U'_{N+1}=U_1;\\ \mu^2\hat{\sigma}_{2,N}^2+(1-\mu)^2(\hat{Q}_{2,N+1}-\hat{Q}(U_2))^2, & \text{если } U'_{N+1}=U_2.\end{cases} \quad (5.1.34)$$

Здесь параметр сглаживания μ выбирается исходя из сведений об уровне нестационарности объекта адаптации. Если объект стационарен, то оптимальное значение параметра $\mu^* = (N - 1) / N$, В нестационарном случае $\mu^* = \text{const}$ и значение его тем больше, чем больше нестационарность.

Легко видеть, что в стационарном случае

$$\lim_{N\to\infty}\hat{\sigma}_{iN}=0 \quad (i=1,2) \quad (5.1.35)$$

При этом $p\to1$ при $Q_1<Q_2$ и $p\to0$ при $Q_1>Q_2$, т. е. в пределе выбирается оптимальная альтернатива.

В многоальтернативном случае при $c = 0$ реализуется линейная тактика, т. е. матрица переходов единичная:

$$P_0=\begin{Vmatrix}100\ldots0\\ \cdots\cdots\\ 00\ldots01\end{Vmatrix} \quad (5.1.36)$$

При штрафе $c = 1$ стохастическая матрица переходов

$$P_1 = \| p_{ij} \|_{q\times q}, \quad (5.1,37)$$

имеет элементы диагональные:

$$p_{ii} = p^*_i \quad (5.1.38)$$

из (4.1.26) и недиагональные:

$$p_{ij} = \upsilon_i \,\text{Вер}\,(U_j \succ U_i) \quad (i \neq j = 1, \ldots, q), \quad (5.1.39)$$

где υ_i — нормирующий множитель:

$$\upsilon_i = \left(\sum_{j \neq i}^{q} \text{Вер}\ (U_j \succ U_i) + p_{ij} \right)^{-1}. \quad (5.1.40)$$

Отсюда видно, что при штрафе i-я альтернатива сохраняется с вероятностью, равной оценке вероятности, что эта альтернатива оптимальна, и изменяется на j-ю вероятностью, пропорциональной вероятности того, что j-я альтернатива лучше i-й. Оценки вероятностей производятся подобно (5.1.30) с использованием оценок, аналогичных (5.1.31) — (5.1.34).

Анализ показывает, что изложенный алгоритм адаптации позволяет находить оптимальную альтернативу и перестраиваться на другую, еще более эффективную, в случае изменения оптимальной альтернативы.

§ 5.2. Исследование алгоритмов альтернативной адаптации

Рассмотрим аналитически процессы, происходящие во время альтернативной адаптации, с помощью автомата с переменной структурой. Для простоты будем изучать двуальтернативный (q=2) процесс [86].

Пусть имеются конкурирующие альтернативы U_1 и U_2. Вероятность предпочтительности одной альтернативы перед другой p = Вер ($U_1 \succ U_2$) априори неизвестна и может изменяться во времени, т. е. $p = p_t$. Способ выбора одной из них, осуществляемый в каждый заданный момент времени по результатам предшествующих независимых экспериментов, представляет собой алгоритм альтернативной адаптации.

Решение подобной задачи заключается в реализации в каждый момент времени простого правила:

$$U = \begin{cases} U_1 \text{ при } p_t > \dfrac{1}{2}; \\ U_2 \text{ при } p_t < \dfrac{1}{2} \end{cases} \quad (5.2.1)$$

и связано с оценкой величины p_t. Использование классических процедур оценивания, требующих либо априорных сведений о характере изменения p_t, либо предположения о стационарности последовательности $\{p_t\}$, в настоящей постановке, отвечающей реальным условиям, невозможно. Это заставляет обращаться к адаптивной процедуре.

Представим результаты экспериментов с альтернативами U_1 и U_2 в дискретные моменты времени t = 1, 2, ... в виде последовательности $\{\alpha_t\}$:

$$\alpha_t = \begin{cases} +1 \text{ при } U_1 \succ U_2; \\ -1 \text{ при } U_1 \prec U_2. \end{cases} \quad (5.2.2)$$

Для простоты предположим, что Вер ($U_1 \sim U_2$) = 0, где «~» — знак эквивалентности альтернатив по заданному критерию. Однако при необходимости эквивалентность альтернатив можно учесть с помощью $\alpha_t = 0$. Тогда, очевидно, Вер ($\alpha_t = 1$) = p_t. В каждый момент времени t процедура должна по результату *at* «рекомендовать» выбор U_1 или U_2.

Определим индикаторную функцию p'_t результатов а эксперимента как вероятность, с которой на t-м шаге следует выбирать альтернативу U_1 (при этом U_2 выбирается с вероятностью 1 — p'_t). Подчиним такую индикаторную функцию естественному требованию: если в течение достаточно долгого времени выполняется $p_t > 1/2$ ($< 1/2$), то $p' \to 1$ ($\to 0$) при $t \to \infty$. Кроме того, указанная тенденция в поведении $p't$ в соответствии с постановкой должна достаточно быстро меняться на противоположную при переходе значений p_t из интервала (1/2, 1] в [0, 1/2), что и обеспечивает адаптивность процедуры.

Индикаторную функцию p'_t, удовлетворяющую сформулированным требованиям, будем искать в классе стохастических итеративных процедур вида

$$p'_{t+1} = p'_t + \lambda_{t+1}\alpha_{t+1}b_t, \quad (5.2.3)$$

где p'_0 и $\{\lambda_t\}$ — детерминированные параметры ($0 \le p'_0 \le 1$; $0 \le \lambda_t \le 1$); $b_t = p'_t(1 - p'_t)$; $\{\alpha_t\}$ — последовательность независимых в совокупности случайных величин с распределениями Вер ($\alpha_t = 1$) = p_t, Вер ($\alpha_t = -1$) = $1 - p_t$ ($0 \le p_t \le 1$). Процесс (5.2.3) является марковским. Если $p'_0 = 0$, то $p'_t = 0$; если $p'o = 1$, то $p'_t = 1$. Иначе говоря, точки 0 и 1 являются неподвижными для преобразования (5.2.3). Чтобы исключить эти и другие особые случаи, будем полагать $0 < p'_0, \lambda_t, p_t < 1$. При этих условиях, очевидно, $0 < p'_t < 1$ почти наверное (п. н.) при $t = 1, 2, ...$

Теорема 1. Пусть $p_t \le p < 1/2$ ($p_t \ge p > 1/2$) при t=1,2,... Тогда если $\lambda_t \le \lambda < 1-2p$ ($\lambda_t \le \lambda < 2p-1$) и

$$\sum_{t=1}^{\infty} \lambda_t = \infty, \quad (5.2.4)$$

то $p'_t \to 0$ ($p'_t \to 1$) п. н. и в среднем при $t \to \infty$.

Доказательство основано на стохастическом аналоге второго метода Ляпунова и проводится по той же схеме, что и доказательство теорем (2.5.2) и (2.7.1) в работе [138]. В качестве функции Ляпунова выбирается $V(x)=x/(1—x)$, $x\in$ $\in [0; 1)$. При $p_t \geq p > 1/2$ вводятся обозначения $x_t = 1—p'_t$, $\beta_{t+1} = = —\alpha_{t+1}$ и процесс (5.2.3) можно записать в виде $x_{t+1} = x_t + + \lambda_{t+1}\beta_{t+1}b_t$. Аналогично для случая $p_t \leq p < 1/2$.

Оценка среднего приращения $V(x)$ по процессу (5.2.3) за один шаг на t-м фиксированном шаге (при $p_t \leq p < 1/2$)

$$M\{V(p'_{t+1}) - V(p'_t) / p'_t\} \leq \lambda_{t+1} V(p'_t)[\lambda-(1-2p)]. \qquad (5.2.5)$$

Следствие. Пусть процесс (5.2.3) стационарен, т. е. $p_t \equiv \equiv p$, $\lambda_t \equiv \lambda$. Тогда если $p<1/2$ ($p>1/2$) и $\lambda<1-2p$ ($\lambda>2p-1$), то $p'_t \to 0$ ($\to 1$) при $t\to\infty$ п. н. и в среднем.

Оценим скорость сходимости процесса (5.2.3) к нулю. Пусть $\lambda_t \geq l > 0$. Из выражения (5.2.5) следует:

$$MV(p_{t+1}) < [1-l((1-2p)-\lambda)]^t \frac{p'_0}{1-p'_0}.$$

Поскольку функция $V(x)$ выпукла, то, воспользовавшись не равенством Йенсена [231], получим оценку скорости сходимо сти в среднем:

$$Mp'_t < [1-l((1-2p)-\lambda)]^t \frac{p'_0}{1-p'_0+p'_0[1-l(1-2p-\lambda)]^t} \qquad (5.2.6)$$

Оценку (5.2.6) можно заменить на менее сложную (но и более грубую):

$$Mp'_t < [1-l(1-2p-\lambda)]^t \frac{p'_0}{1-p'_0}. \qquad (5.2.7)$$

Легко проверить, что правая часть выражения (5.2.7) минимальна при $l=\lambda=1/2—p$. При этом

$$Mp'_t < \left[1-\left(\frac{1}{2}-p\right)^2\right]^t \frac{p'_0}{1-p'_0}.$$

5.2.8) Из (5.2.3) можно непосредственно оценить Mp'_t и p'_t снизу:

$$Mp'_t > \prod_{k=1}^{t}[1-\lambda_k(1-p_k)]p'_0; p'_t > (1-\lambda)^t p'_0 \text{ п. н.} \qquad (5.2.9)$$

Пусть теперь $p_t \geq p > 1/2$ и $0 < l \leq \lambda_t \leq \lambda < 2p-1$. Проведя те же преобразования, что и при доказательстве теоремы 1, получим оценки, аналогичные (5.2.6) —

(5.2.9), скорости сходимости

процесса (5.2.3) в среднем к единице. При этом в левых частях неравенств окажется $M(1 - p'_t)$, а в правых вместо $(1 - 2p)$ следует записать $(2p - 1)$.

Рассмотрим ситуацию, когда последовательность $\{p_t\}$ существенно нестационарна, т. е. при различных t вероятности p_t могут быть как больше 1/2, так и меньше. Обозначим $\beta_t = M\alpha_t = 2p_t - 1$. Не ограничивая общности, будем считать $p < 1/2$,

$$\beta_t < 0 \left(p_t < \frac{1}{2} \right) \text{ при } t \in T_0 = \{1, 2, \ldots, t_1\};$$

$$\beta_t > 0 \left(p_t > \frac{1}{2} \right) \text{ при } t \in T_1 = \{t_1 + 1, \ldots, t_2\};$$

$$\beta_t < 0 \left(p_t < \frac{1}{2} \right) \text{ при } t \in T_2 = \{t_2 + 1, \ldots, t_3\} \text{ и т. д.}$$

(5.2.10)

Будем говорить, что T_0, T_2, ... являются промежутками отрицательной, а T_1, T_3, ... — положительной квазистационарности. В этих условиях процесс (5.2.3), вообще говоря, не будет сходиться к 0 или к 1 при $t \to \infty$. Однако при $t \in T_t$ последовательность $\{p'_t\}$ будет иметь определенную тенденцию — монотонно приближаться в среднем к нулю или к единице в зависимости от четности i.

Далее ограничимся классом процедур вида (5.2.3), в которых $\lambda_t \equiv \lambda$ (условие (5.2.4) всегда выполнено). Пусть $p't_i$ произвольно, но фиксированно и $T_i = \{t_i + 1, ...\}$, т. е. $t_{i+1} = \infty$. Тогда

по теореме 1, если $p_t < \frac{1}{2} - \frac{\lambda}{2} \left(p_t > \frac{1}{2} + \frac{\lambda}{2} \right)$ при $t \in T_i$, то $p'_t \to 0$

$(p'_t \to 1)$ п. н. Если $\frac{1}{2} - \frac{\lambda}{2} \leqslant p_i < \frac{1}{2} \left(\frac{1}{2} < p_t \leqslant \frac{1}{2} + \frac{\lambda}{2} \right)$, то

сходимость к нулю (единице) теоремой 1 не гарантируется, однако Mp'_t монотонно убывает (возрастает) при $t \to \infty$. Параметр λ тем самым определяет области чувствительности процедуры (5.2.3).

Назовем число $\gamma \left(0 < \gamma < \frac{1}{2} \right)$ порогом чувствительности процедуры (5.2.3), если оно удовлетворяет априорному требованию: $p'_t \to 0$ и $p'_t \to 1$ по крайней мере при $p_t < \frac{1}{2} - \gamma$ и $p_t > \frac{1}{2} + \gamma$. Эта цель будет достигнута при любом $\lambda < 2\gamma$. Однако, как следует из

формул (5.2.6) — (5.2.9), скорость сходимости различна при разных λ. Легко проверить, что при заданном γ скорость сходимости в среднем оптимальна, если $\lambda=\gamma$.

Пусть промежуток T_i имеет конечную длину. Тогда при помощи формул (5.2.6) — (5.2.9) можно оценить, как близко про-

цесс (5.2.3) успеет подойти к 0 или 1. Эти оценки существенно зависят от начального значения p'_0, в качестве которого здесь будет выступать реализация случайной величины p'_{t_i}. Чем ближе реализация p'_{t_i}, например, к 1, тем медленнее p'_t приближается к 0. Другими словами, чем лучше процесс (5.2.3) адаптировался к одному режиму, тем хуже он перестраивается при изменившихся условиях. Кроме того, поскольку все вычисления (и на ЭВМ) округляются, то при реализации p'_{t_i}, достаточно близкой к единице (или к нулю), процесс (5.2.3) уже не выйдет из точки 1 (или 0) ни при каких последующих значениях p_i.

В реальных задачах всегда можно априори задать точность адаптации ε таким образом, что приближение p'_t к 0 или 1 ближе, чем на ε, не является практической необходимостью. Воспользовавшись такой ситуацией, можно избежать губительной «переадаптации» процедуры (5.2.3).

Пусть $\{\alpha_t\}$, λ, $\{b_t\}$ — те же, что и в процессе (5.2.3), и $\varepsilon \le p'_0 \le 1-\varepsilon$. Определим новый процесс, который назовем ε — адаптивной процедурой выбора:

$$p'_{t+1} = \begin{cases} \varepsilon & \text{при} \quad x_{t+1} < \varepsilon; \\ x_{t+1} & \text{при} \quad \varepsilon \leqslant x_{t+1} \leqslant 1-\varepsilon; \\ 1-\varepsilon & \text{при} \quad 1-\varepsilon < x_{t+1}, \end{cases} \tag{5.2.11}$$

где $x_{t+1} = p'_t + \lambda\alpha_{t+1}b_t$.

Организованная таким образом индикаторная функция удовлетворяет условию $\varepsilon \le p'_t \le 1-\varepsilon$ п. н. при $t = 1, 2, \ldots$ Следовательно, процедура (5.2.11) приобретает способность к быстрой переадаптации.

Асимптотические свойства процессов (5.2.11) и (5.2.3) различны. Однако поскольку при $\varepsilon \le p'_t \le 1-\varepsilon$ п. н. они совпадают, то формулы (5.2.6) — (5.2.9) являются некоторыми характеристиками и процедуры (5.2.11). Пусть $t \in T_i$. Не ограничивая общности, будем считать промежуток T_i отрицательно квазистационарным. Тогда, как следует из выражения (5.2.6),

$$M\{p'_t \mid p'_{t_i}\} < [1-\lambda(2\gamma-\lambda)]^{t-t_i} \frac{p'_{t_i}}{1-p'_{t_i}+p'_{t_i}[1-\lambda(2\gamma-\lambda)]^{t-t_i}} \tag{5.2.12}$$

Эта оценка справедлива для процесса (5.2.11) до тех пор, пока $p'_t \ge \varepsilon$ п. н. Поскольку $p'_t > p'_{t_i}(1-\lambda)^{t-t}$ п. н., то, предполагая процесс адаптации на промежутке T_{i-1} законченным, т. е. $p'_{t_i} = 1-\varepsilon$, найдем время, в течение которого справедливо (5.2.12):

$$t - t_i \leqslant \ln \frac{\varepsilon}{1-\varepsilon} \Big/ \ln(1-\lambda).$$

Процедура (5.2.11) по существу «отрезает» те траектории процесса (5.2.3), которые опускаются ниже уровня ε (поднимаются выше 1—ε). Поэтому может оказаться полезной оценка максимального времени достижения ε-уровня процессом (5.2.3) в среднем при условии, что он начинается из состояния $\varepsilon \leq \leq p'_{t_i} \leq 1-\varepsilon$ п. н. Несложными преобразованиями из выражения (5.2.3) получим

$$t - t_i \leqslant T(\varepsilon, \gamma, \lambda) = 2 \ln \frac{\varepsilon}{1-\varepsilon} \Big/ \ln [1 - \lambda(2\gamma - \lambda)].$$

Эта косвенная характеристика процесса (5.2.11), которую назовем временем ε-адаптации, однозначно определяется задаваемыми параметрами самого алгоритма и не зависит от поведения $\{\alpha_t\}$.

Легко проверить, что время $T(\varepsilon,\gamma,\lambda)$ минимально при $\lambda=\gamma$, что соответствует найденной выше оптимальной скорости сходимости процесса (5.2.3). При этом $T(\varepsilon,\gamma,\lambda)_{\text{опт}} = T(\varepsilon,\gamma) =$

$$= 2 \ln \frac{\varepsilon}{1-\varepsilon} \Big/ \ln (1-\gamma^2).$$

Чтобы практически использовать процедуру (5.2.11), необхо димо определить согласно смыслу задачи точность ε и порог чувствительности γ. Полагая затем в формуле (5.2.11) $\lambda=\gamma$, получим оптимальный процесс со средним временем адаптации $T(\varepsilon,\gamma)$.

§ 5.3.
Альтернативная адаптация в процессах поисковой оптимизации

Рассмотрим структурную адаптацию в процессах поисковой оптимизации [168].

Проблема поисковой оптимизации возникла в связи с необ-ходимостью решения сложных задач типа математического программирования:

$$Q(X) \to \min_{X \in \Omega}, \quad (5.3.1)$$

возникающих при алгоритмизации процессов управления, принятии оптимальных решений и т. д. Здесь $Q(X)$ — скалярная функция качества векторного аргумента $X=(x_1,...,x_n)$, опре-деленного в области Ω. Эта область может быть или континуальной и, задаваться системой равенств и неравенств:

$$\Omega: \begin{cases} g_i(X) = 0 & (i = 1, \ldots, l < n); \\ h_j(X) \geqslant 0 & (j = 1, \ldots, p), \end{cases} \quad (5.3.2)$$

или конечным множеством, т. е. определять дискретные значения аргумента:

$$\Omega : \{X_i\} \quad (i = 1, ...), \tag{5.3.3}$$

или сочетанием этих ограничений. Возможны и другие формы задания области Ω.

Сложность решения (5.3.1) для мало-мальски реальных задач заключается прежде всего в том, что функция $Q(X)$ и область Ω заданы не аналитически, а алгоритмически, т. е. в виде всякого рода правил, инструкций и указаний, имеющих как формальный, так и неформальный (экспертный) характер. Это обстоятельство практически исключает применение стандартных методов математического программирования, опирающихся на известную структуру и вид функции Q(X) и области Ω.

Задача, таким образом, заключается в отыскании хотя бы одного решения X^*, удовлетворяющего очевидному условию

$$Q(X^*) = \min_{X \in \Omega} Q(X). \tag{5.3.4}$$

Суть поискового метода отыскания X^* сводится к построению последовательности

$$X_0, X_1, ..., X_N, ..., \tag{5.3.5}$$

которая должна сходиться к достаточно малой окрестности решения X^*. Алгоритм поиска F связывает следующие друг за другом состояния:

$$X_{N+1} = F(X_N, W_N), \tag{5.3.6}$$

где W_N — фактор предыстории, например $W_N = \{X_{N-1}, ..., X_{N-l}\}$; l — глубина предыстории. Однако удобнее алгоритм F определять для смещения

$$\Delta X_{N+1} = F(X_N, W_N), \tag{5:3.7}$$

связывающего два соседних состояния $X_{N+1} = X_N + \Delta X_{N+1}$ в пространстве оптимизируемых параметров $\{X\}$.

Очевидно, что алгоритм F должен удовлетворять определенным требованиям, предъявляемым к процессу поиска решения задачи. Эти требования можно представить в виде набора критериев, характеризующих эффективность процесса поиска:

$$K = (k_1, ..., k_m). \tag{5.3.8}$$

Каждый из критериев зависит от алгоритма F и ситуации L, сложившейся в процессе оптимизации:

$$L = \langle Q(X), L \rangle. \tag{5.3.9}$$

Таким образом, каждый критерий в (5.3.8) имеет вид

$$k_i = k_i(F, L) \quad (i = 1, ..., m). \tag{5.3.10}$$

Задача выбора оптимального алгоритма сводится тем самым к задаче оптимизации:

$$\tilde{K}(F, L) \to \min_{F \in E} \Rightarrow F^*_L, \tag{5.3.11}$$

где $\tilde{K}$ — оптимизируемый критерий эффективности алгоритма F, выбранный из множества (5.3.10); E — множество допустимых алгоритмов, удовлетворяющих остальным критериям из (5.3.10). Решением задачи (5.3.11) является оптимальный алгоритм F^*_L, зависящий, естественно, от ситуации L.

Введя понятие класса ситуаций

$$\{L\} = \{Q(X), L\} \tag{5.3.12}$$

и правило оценки критерия $\tilde{K}$ на этом классе (например, как максимума или среднего на классе), получаем оптимальные алгоритмы поиска. Примерами таких оптимальных алгоритмов являются известные алгоритмы Кифера и Ньютона: первый — для класса одномерных унимодальных функций без случай ных помех, заданных на ограниченном интервале, при мини максном критерии оптимальности, связанном с интервалом не определенности в оценке положения экстремума; второй — для класса положительно-определенных квадратичных форм при $\Omega = R^n$ и отсутствии случайных помех.

Однако число подобных оптимальных алгоритмов весьма мало, да и ценность их при решении сложных задач оптимизации сомнительна, так как обычно чрезвычайно трудно определить принадлежность конкретного объекта к тому или иному классу, для которого построен оптимальный алгоритм. Именно это обстоятельство заставляет обращаться к адаптации алгоритмов поиска, т. е. приспосабливать алгоритм на каждом шаге к сложившейся ситуации L (здесь под ситуацией понимается уже не вся задача оптимизации (5.3.9), а лишь ее локальный фрагмент в окрестности точки X_N).

Представим алгоритм F поиска в виде двойки

$$F = \langle S, C \rangle, \tag{5.3.13}$$

где S — его структура, а $C = (c_1, ..., c_k)$ — параметры. Такое разделение условно, поскольку нет строго формального критерия выделения структуры алгоритма. Однако интуитивно структурные категории конкретного алгоритма довольно согласованно определяются специалистами по поиску.

Адаптация алгоритма F может производиться по двум направлениям. Прежде всего, можно идти традиционным путем

изменения параметров алгоритма C (см. § 3,5, где описаны методы адаптации рабочего шага, объема накопления и параметров распределения случайного шага). Этот вид адаптации имеет ярко выраженный параметрический характер и широко используется при поисковой оптимизации сложных систем. Он хорошо исследован, и его применение не вызывает, как правило, серьезных алгоритмических трудностей [182].

Однако эффект от параметрической адаптации часто бывает не столь высок, как необходимо. Именно это обстоятельство заставляет обращаться к адаптации структуры S алгоритма F. Тогда приходится решать задачу, которая получается путем декомпозиции из (5.3.11):

$$\tilde{K}(F, L) \to \min_{S \in E_S} \min_{C \in E_{CS}} \Rightarrow S^*_L C^*_L, \qquad (5.3.14)$$

решением которой является оптимальная структура S^*_L алгоритма F, естественным образом зависящая от ситуации L. Здесь ES — множество допустимых структур алгоритма F, а E_{CS} — множество допустимых параметров алгоритма, имеющего структуру S.

Как видно, задача (5.3.14) адаптации структуры является двухуровневой задачей, где на первом уровне решается задача параметрической адаптации (быстрый контур адаптации), а на втором — задача структурной адаптации (медленный контур) (рис. 5.3.1). Это означает, что каждый шаг структурной адаптации должен сопровождаться целым циклом работы контура параметрической адаптации. В противном случае решение, принятое на верхнем уровне, будет недостоверным из-за того, что значение критерия качества данной структуры оказалось неоптимизированным по параметрам этой структуры.

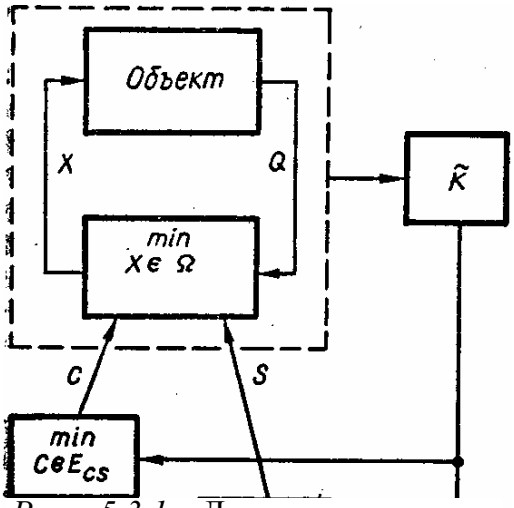


Рис. 5.3.1. Двухконтурная система адаптации алгоритма поиска.

Рассмотрим основные возможные подходы к решению задачи структурной адаптации алгоритма поиска.

1. Идентификация ситуации из конечного множества [74, 84, 120] опирается на заранее описанное конечное множество альтернативных ситуаций

$$\{L_j\} \quad (j = 1, ..., z<\infty) \qquad (5.3.15)$$

типа «яма», «овраг», «долина» и т. д. По выбранному критерию

$\bar{K}$ каждой ситуации соответствует один из альтернативных алгоритмов:

$$\{U_i\} \quad (i = 1, .., q), \tag{5.3.16}$$

т. е. имеет место определенное соответствие между сложившейся ситуацией и оптимальной структурой алгоритма: $L_i \rightarrow S_i$, которое реализуется функцией $i=\varphi(j)$ (i = 1, ..., q; j = 1, ..., z). Задание (или определение) функции φ однозначно решает задачу адаптации структуры. Теперь она сводится к идентификации ситуации L, т. е. к отнесению имеющейся ситуации к одной из множества (5.3.15). Процедура идентификации должна использовать прежде всего результаты работы предыдущего алгоритма, а также специальные тесты, наиболее быстро выявляющие принадлежность текущей ситуации к одной из (5.3.15). **2. Идентификация ситуации из континуального множества.** В этом случае каждую ситуацию удобно кодировать вектором

$$L = (p_1, ..., p_u) \tag{5.3.17}$$

в u-мерном пространстве. Каждой точке L этого пространства соответствует лучший алгоритм U_i, определяемый из очевид ного соотношения

$$\bar{K}(U_i, L) = \min_{j=1,...,q} \bar{K}(U_j, L). \tag{5.3.18}$$

Таким образом, имеет место q-классовая задача распознавания образов, построенная на основе конечной обучающей выборки

$$\langle L_j, U_j \rangle \quad (j = 1, ..., N), \tag{5.3.19}$$

элементы которой получены заранее путем N-кратного решения оптимизационной задачи

$$\bar{K}(U, L_j) \rightarrow \min_{U_1,...,U_q} \Rightarrow U_j, \tag{5.3.20}$$

где L_j (j=1,...,N) — ситуации, представляющие различные части пространства $\{L\}$.

Пример [*165*, *167*]. Рассмотрим два алгоритма случайного поиска U_1 и U_2 (т. е. две альтернативные структуры S_1 и S_2): с нелинейной тактикой —

$$U_1: \Delta X_{N+1} = \begin{cases} \xi & \text{при } \Delta Q_N < 0; \\ -\Lambda X_N & \text{при } \Delta Q_N \geqslant 0; \end{cases}$$

с линейной тактикой —

$$U_2: \Delta X_{N+1} = \begin{cases} \xi & \text{при } \Delta Q_N \geqslant 0; \\ \Delta X_N & \text{при } \Delta Q_N < 0, \end{cases}$$

(5.3.21) (5.3.22)

где ξ — случайный шаг. Пространство ситуаций $\{L\}$ двумерно (u=2) и образуется двумя вероятностями:

$$p_1 = \text{Вер}\,(\Delta Q_N < 0 \mid \Delta X = \xi) \tag{5.3.23}$$

— вероятностью того, что случайный шаг окажется удачным, и

$$p_2 = \text{Вер}\,(\Delta Q_N < 0 \mid \Delta Q_{N-1} < 0,\ \Delta X_N = \Delta X_{N-1}) \tag{5.3.24}$$

— вероятностью того, что шаг в удачном направлении удачен. Если в качестве критерия K воспользоваться средним штрафом, т. е. средним числом неудачных шагов, то пространство (точ нее, квадрат) $\{L\}$ ситуаций ($0 \le p_1,\ p_2 \le 1$) разбивается пара болой $p_2 = p_1^2 - p_1 + 1$ на две части, соответствующие ситуациям, в которых целесообразно применять один из указанных алго ритмов:

$$U = \begin{cases} U_1 & \text{при}\ \ p_2 < p_1^2 - p_1 + 1; \\ U_2 & \text{при}\ \ p_2 > p_1^2 - p_1 + 1. \end{cases} \tag{5.3.25}$$

При $p_2 = p_1^2 - p_1 + 1$ оба алгоритма по критерию среднего штрафа эквивалентны. Как видно, располагая значениями вероятностей (5.3.23), и (5.3.24), можно «включить» оптимальную в данной ситуации альтернативную структуру U^*. Этим задача адаптации сводится к задаче оценивания вероятностей p_1 и p_2 на каждом шаге поиска.

Исходная информация для этой оценки — показатель d:

$$d_N = \begin{cases} 1, & \text{если}\ \ \Delta X_N = \Delta X_{N-1}; \\ 0, & \text{если}\ \ \Delta X_N = \xi; \\ -1, & \text{если}\ \ \Delta X_N = -\Delta X_{N-1}. \end{cases} \tag{5.3.26}$$

Определяющим является либо последний повторный шаг (d = 1), либо случайный (d = 0), либо возвратный (d = —1). Располагая парой

$$\langle c_N, d_N \rangle \tag{5.3.27}$$

на каждом шаге поиска, можно оценивать искомые вероятности по следующим рекуррентным формулам:

$$p_1^{(N+1)} = \begin{cases} \gamma_1 p_1^{(N)} + 1 - \gamma_1 & \text{при}\ \ c_N = 0 \\ \gamma_2 p_1^{(N)} & \text{при}\ \ c_N = 1 \\ p_1^{(N)} & \text{при}\ \ d_N \neq 0; \end{cases} \quad d_N = 0;$$

$$p_2^{(N+1)} = \begin{cases} \gamma_3 p_2^{(N)} + 1 - \gamma_3 & \text{при}\ \ c_N = 0 \\ \gamma_4 p_2^{(N)} & \text{при}\ \ c_N = 1 \\ p_2^{(N)} & \text{при}\ \ d_N \neq 0, \end{cases} \quad d_N = 0; \tag{5.3.28}$$

где $0<\gamma_i<1$ (i = 1, ..., 4) — параметры, отражающие интенсивность изменения ситуации $L = (p_1, p_2)$ в оптимизируемом объекте в процессе поиска.

Однако предварительная идентификация несвойственна процессам адаптации. Поэтому для структурной адаптации поиска естественно использовать алгоритмы, описанные в § 5.1.

3. Начнем с алгоритмов, реализуемых **автоматами с целесообразным поведением** (см. п. 5.1.2.1). Приведем примеры реализации различных алгоритмов адаптации поиска [215].

Пусть в моменты времени N_1, ..., N_k, ... происходит выбор алгоритма оптимизации. Определим функцию

$$K(i)=\frac{Q(X_{N_{i+1}})-Q(X_{N_i})}{|X_{N_{i+1}}-X_{N_i}|}-\frac{Q(X_{N_i})-Q(X_{N_{i-1}})}{|X_{N_i}-X_{N_{i-1}}|}, \quad (5.3.29)$$

которая является аналогом ускорения процесса поиска. Естественно выбрать тот алгоритм, который обеспечивает отрицательное значение этой функции. Тогда под удачей (c=0) понимаем событие $K(i)<0$, а под неудачей (c=1) — событие $K(i)>0$. Заметим, что функцию $K(i)$ можно ввести и другим образом.

Был проведен ряд модельных экспериментов для исследования такого подхода к альтернативной адаптации алгоритмов поиска [215].

В первой серии экспериментов в качестве алгоритмов оптимизации были выбраны два одинаковых алгоритма «с пересчетом» с адаптацией величины рабочего шага [184]:

$$\Delta X_k=\begin{cases} a_k\xi_k, & \text{если } Q(X_{k-1}+a_k\xi_k)<Q(X_{k-1}); \\ 0, & \text{если } Q(X_{k-1}+a_k\xi_k)\geqslant Q(X_{k-1}); \end{cases} \quad (5.3.30)$$

$$a_{k+1}=\begin{cases} \gamma_1 a_k, & \text{если } Q(X_k)<Q(X_{k-1}); \\ \gamma_2 a_k, & \text{если } X_k=X_{k-1}, \end{cases} \quad (5.3.31)$$

где ξ_k — случайные независимые векторы, равномерно распределенные на единичной сфере; $\gamma_1>1$, $0<\gamma_2<1$. Эти алгоритмы отличались друг от друга только параметром p, согласно которому выбирались коэффициенты γ_1 и γ_2 исходя из рассмотренного в § 3.5 равенства: $\gamma_1^{\,p}\gamma_1^{\,1-p}=1$.

В работах [184, 215] было доказано (см. также § 3.5), что Для гладких модельных функций оптимальная величина p^* асимптотически стремится к 0,27, а для небольших n приблизительно равна 0,2. Поэтому естественно в одном алгоритме

взять p = 0,2, а во втором, неоптимальном, — p = 0,45. Работа альтернативной адаптации исследовалась на функции вида

$$Q(X) = \sum_{i=1}^{n} x_i^2. \quad (5.3.32)$$

При начальном условии x_0= (10, 0, ..., 0) моменты времени переключения алгоритмов определялись следующим образом: $N_i = iK$, т. е. через K тактов. На рис. 5.3.2 показано осредненное на базе десяти экспериментов поведение минимизируемой функции при n=8, l=3 и K=3; 5; 10. Видно, что $K_{опт}$ = 5 и эффективность поиска незначительно уступала эффективности оптимального алгоритма (напомним, что l — глубина памяти автомата).

На рис. 5.3.3 представлены аналогичные данные для n = 8, K=5 и l=2; 3; 5. Как видно, лучшие результаты дает l=2—3, т. е. незначительная глубина памяти.

В следующем эксперименте изучался разброс двуальтернативных адаптивных алгоритмов. На рис. 5.3.4 представлены результаты расчетов для случая n=8, K=5, l=3. Здесь даны лучший и худший (по останову) результаты из десяти и осредненная траектория. Хорошо видно, что разброс невелик и худший

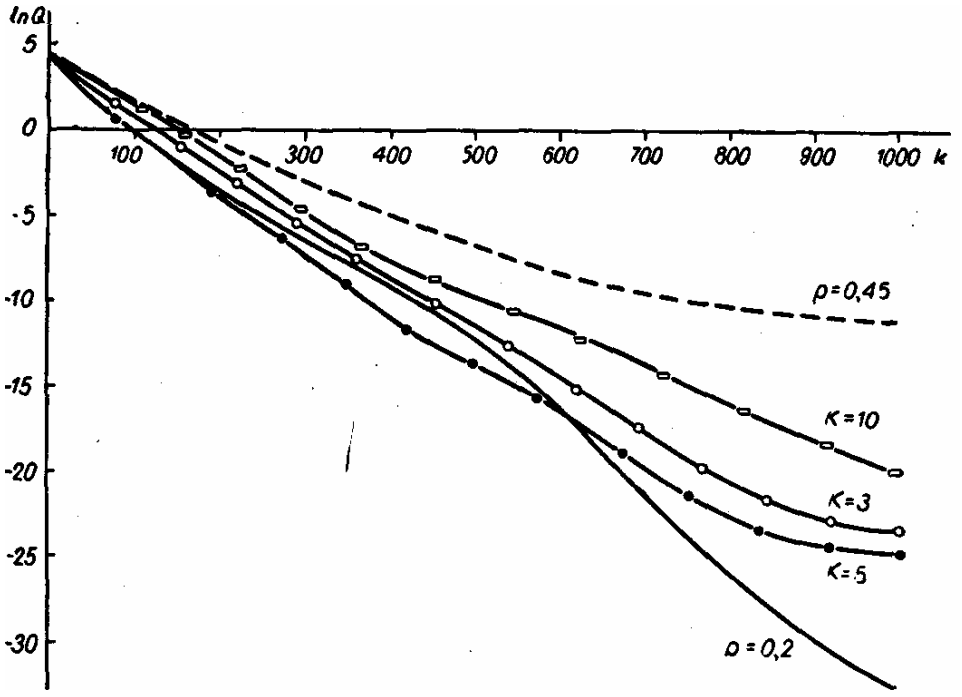


Рис. 5.3.2. Результаты оптимизации адаптивным двуальтернативным алгоритмом при различных значениях параметра тактности K. Для сравнения приведены результаты работы альтернативных алгоритмов в отдельности (p=0,2; 0,45).

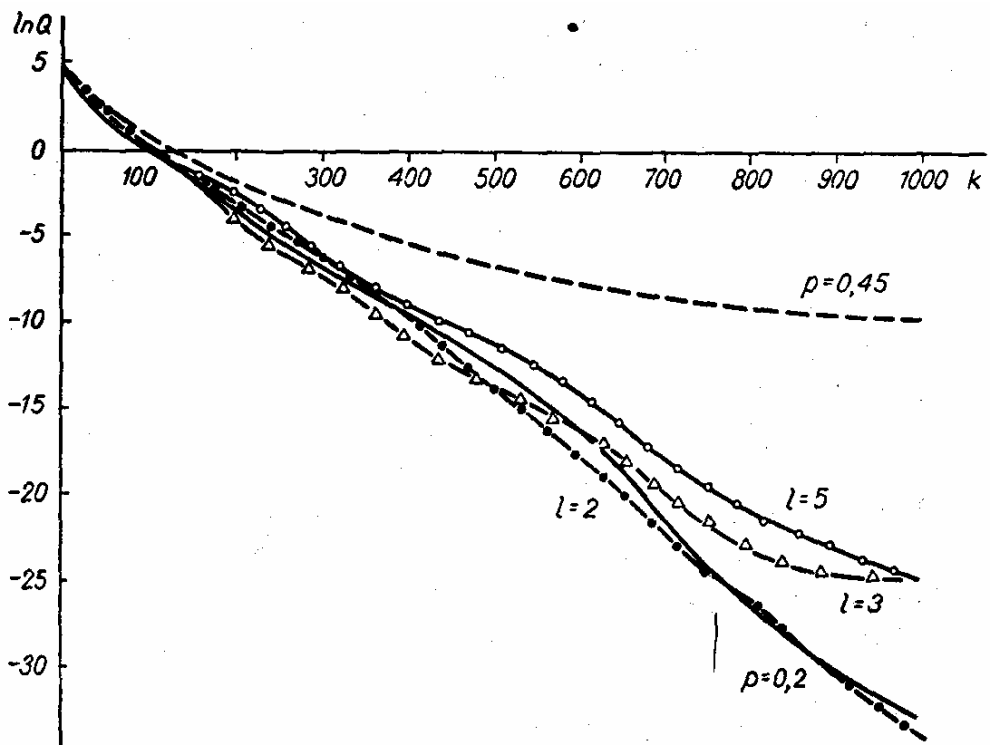


Рис. 5.3.3. Результаты оптимизации адаптивным двуальтернативным алгоритмом при различной глубине памяти l автомата. Для сравнения приведены результаты работы альтернативных алгоритмов в отдельности (p=0,2; 0,45).

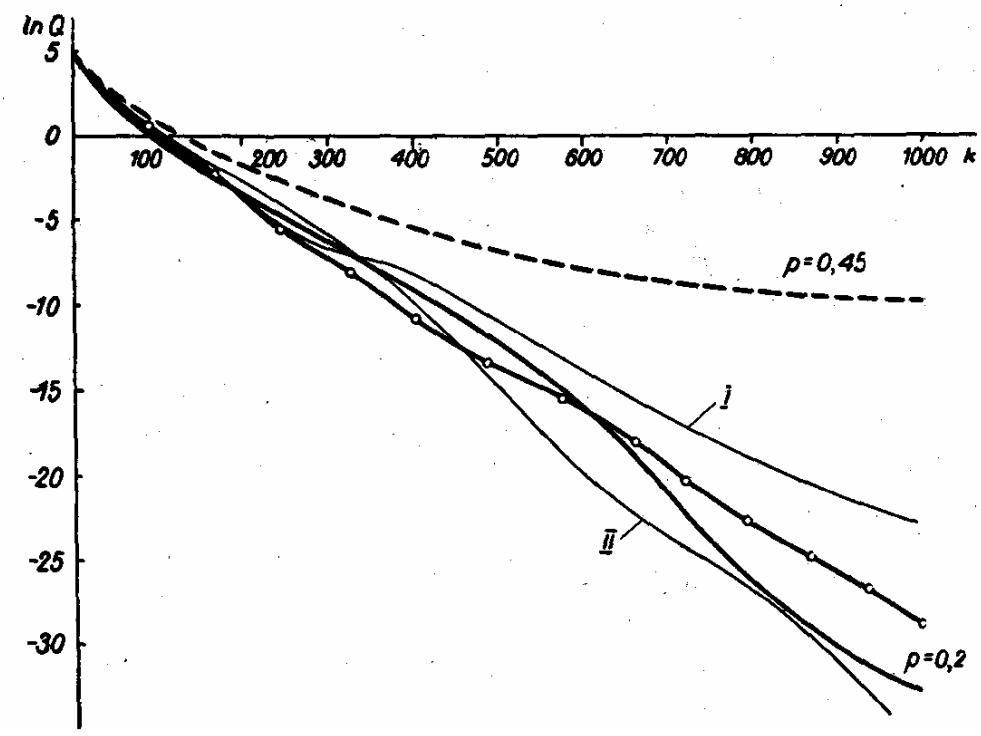


Рис. 5.3.4. Результаты оптимизации адаптивным двуальтернативным алгоритмом.
I — худшая траектория, *II* — лучшая траектория. Точками обозначена осредненная траектория. Для сравнения приведены результаты работы альтернативных алгоритмов в отдельности (p=0,2; 0,45).

вариант адаптивного алгоритма всегда лучше худшего алгоритма (p = 0,4) из двух его составляющих, а в среднем он примерно совпадает с лучшим.

Вторая серия экспериментов проводилась с тремя аналогичными алгоритмами случайного поиска. Значения p выбирались следующие: p = 0,07; p = 0,2 и p = 0,45. Результаты экспериментов полностью повторили данные предыдущей серии: совместная работа трех алгоритмов не хуже, чем работа одного оптимального алгоритма.

В третьей серии экспериментов рассмотрена совместная работа «реальных» поисковых алгоритмов — метода Давидона— Флетчера— Пауэлла (ДФП) [233] и метода случайного поиска «с пересчетом» с адаптацией величины рабочего шага и плотности распределения случайных векторов [182, 220]. В качестве класса минимизируемых функций было выбрано множество функций вида

$$Q(X) = \sum_{i \in I_1} a_i x_i^2 + \sum_{i \in I_2} a_i |x_i|, \quad (5.3.33)$$

где $I_1 \cup I_2 = \{1, \ldots, n\}$; $0 < a_{\min} \le a_i \le a_{\max} < \infty$ (i=1,...,n). Множества I_1 и I_2 и начальная точка X_0 разыгрывались случайно. Переключение алгоритмов проводилось через K итераций.

Для функций, в которых присутствуют члены второй суммы (5.3.33), метод ДФП не является сходящимся, но для функции (5.3.32) он имеет неоспоримые преимущества перед рассматриваемым методом случайного поиска. Результаты экспериментов подтвердили высокую эффективность использования альтернативного метода структурной адаптации по сравнению с отдельными алгоритмами: для всех функций минимум находился с заданной точностью.

4. Рассмотрим теперь эксперименты по использованию **стохастического автомата с переменной структурой** (см. п. 5.1.2.2) для целей альтернативной адаптации поиска [174]. Альтернативные алгоритмы были те же, что и ранее — (5.3.30) и (5.3.31), но с другими значениями параметра p: $U_1 = U_{\text{опт}}$: $p = = p^* = 0{,}2$; U_2: $p = 0{,}45$. Таким образом, априори было известно, что второй алгоритм заведомо хуже первого по быстродействию. Эксперименты показали, что устойчивая работа алгоритма обеспечивалась лишь при дополнительном сглаживании вида

$$\bar{\alpha}_N = \bar{\alpha}_N + \beta \operatorname{sign} \alpha_N, \quad (5.3.34)$$

где

$$\alpha_N = \frac{\Delta Q_N^{(1)} - \Delta Q_N^{(2)}}{2\left(\hat{\sigma}_N^{(1)} + \hat{\sigma}_N^{(2)}\right)^{1/2}}, \quad (5.3.35)$$

а $\Delta Q_N^{(i)}$ — приращение показателя качества на N-м этапе при работе i-го алгоритма U_i ($i = 1, 2$).

Параметрами альтернативной адаптации являлись:

— число K тактов, образующих один этап поиска, за время которого происходит оценка

$$\Delta Q_N = \frac{1}{K} \sum_{i=1}^{K} \Delta Q_{N+i} \qquad (5.3.36)$$

(в течение одного этапа алгоритм поиска не. изменяется);

— параметр μ сглажи вания оценок (5.1.31) — (5.1.34);

— параметр β сглажи вания в формуле (5.3.34).

Результаты экспериментов, осредненные по десяти реализациям, представлены на рис. 5.3.5, где показано поведение десятичного логарифма минимизируемой функции в процессе оптимизации. Параметры алгоритма адаптации были следующими:

Случа	K	μ	β
а	1	0,9	0,2
б	5	0.5	0.5
в	1	0.5	0.1
г	5	0.9	0.5

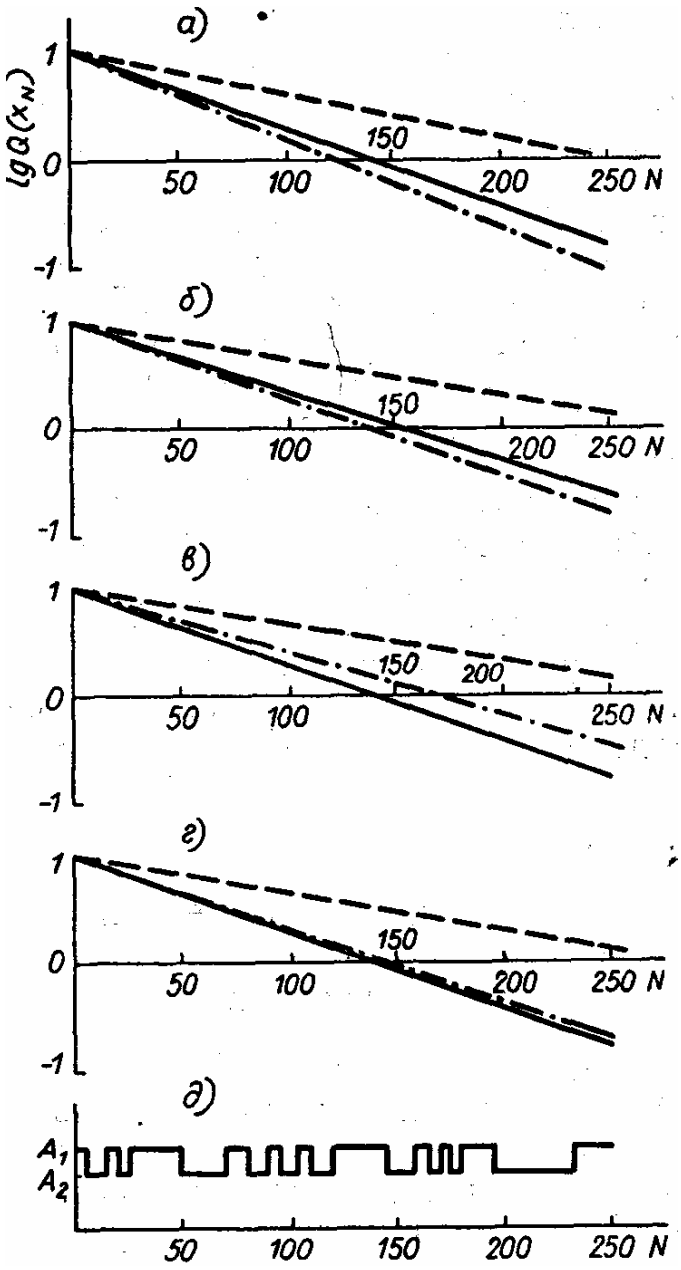


Рис. 5.3.5. Результаты экспериментов. *а—г* — динамика оптимизации для различных значений параметров адаптации. Пунктир — неоптимальный алгоритм (A_1), штрихпунктир — оптимальный (A_2); сплошная линия — алгоритм со структурной адаптацией, *д* — диаграмма переключения алгоритмов для случая «*г*».

Из рисунка видно, что алгоритм поиска со структурной адаптацией всегда лучше худшего в данной ситуации алгоритма. Более того, при удачном выборе параметров адаптации этот алгоритм может работать как лучший (см. рис. 5.3.5, г) или Даже лучше него (см. рис. 5.3.5, *в*).

Дело в том, что алгоритм переключения включает наилучший по критерию быстродействия алгоритм поиска. Но в процессе параметрической адаптации случайно таким наилучшим может оказаться худший в среднем (p = 0,45) алгоритм, который и внесет свой вклад в процесс поиска. Это объяснение подтверждается тем, что, хотя в среднем всюду $U_1 \succ U_2$, структурная адаптация не исключала полностью U_2, что ожидалось в экспериментах (см. рис. 5.3.5, *д*). Как видно, альтернативная адаптация такого рода позволяет использовать положительные обстоятельства, складывающиеся в процессе оптимизации, и тем самым значительно повысить эффективность процесса поиска.

Заметим в заключение параграфа, что использование подобной альтернативной адаптации в пакете программ оптимизации, осуществляющей «переключение» различных программ пакета, позволяет существенно повысить эффективность работы пакета оптимизации, особенно в обстановке значительной неопределенности относительно специфики структуры решаемой задачи.

§ 5.4. Альтернативная адаптация в процессах передачи данных

5.4.1. Постановка задачи

Подсистема передачи данных в вычислительной сети обычно считается звеном, наиболее подверженным воздействию неопределенных факторов в виде нестационарного уровня случайных помех в каналах связи. Эти помехи вызываются всякого рода электрическими источниками (атмосферными, техническими и т. д.), влияющими на каналы связи, повреждениями линии связи, а также непредсказуемыми изменениями загрузок и пропускной способности отдельных каналов вычислительной сети.

Наиболее перспективным средством управления системами, передачи данных, подверженными воздействиям неопределенных факторов, является адаптация как метод управления [170]. Среди задач, решаемых адаптивными методами, выделим:

а) задачу выбора маршрутов передачи данных («адаптив ная маршрутизация»);

б) задачу управления потоком передачи данных в канале связи в условиях нестационарных помех.

Отметим следующие результаты. Вопросы адаптации для динамического управления потоками пакетов, т. е. решения задачи маршрутизации в вычислительной сети в условиях,

когда загрузка линий связи и конфигурация сети изменяются в широких пределах, рассматривались в работах [74, 203]. Среди прочих методов маршрутизации широкое распространение получили локальные алгоритмы адаптации, реализуемые в распределенных (децентрализованных) системах и оперативно использующие для выбора очередного маршрута передачи пакета текущую информацию, накапливаемую в узле коммутаций системы передачи данных. Адаптация процесса направления потоков данных в сети с коммутацией сообщений была рассмотрена в работе [121]. Предложенные алгоритмы предназначены для автономного выбора наиболее удачного маршрута передачи сообщения в каждом узле сети на основе прогноза удачности выбора методом линейной фильтрации предыдущих наблюдений.

Интересно предложение использовать стохастические обучающиеся автоматы, обладающие свойством быстрого приспособления к изменяющимся условиям, для адаптации процесса маршрутизации вызовов в сети телефонной связи [121]. С помощью модельного эксперимента показано преимущество адаптивного рандомизированного выбора направления прохождения телефонных вызовов по сравнению с регулярными способами выбора маршрутов, используемыми в настоящее время.

Простейший детерминированный адаптивный способ управления скоростью передачи потока дискретной информации в условиях пачек ошибок в канале, работающем в полудуплексном режиме, представлен в работе [51]. Для этого предлагается использовать два кода, корректирующих ошибки, которые обладают различной помехозащищенностью. Система переходит к более помехоустойчивому, а следовательно, и более избыточному коду в случае обнаружения ошибки в сообщении, т. е. по запросу повторной передачи блока, и к менее помехоустойчивому коду — в случае отсутствия ошибок. Такая оперативная реакция на состояние канала связи является альтернативной адаптацией, хотя и лишена фильтрации, необходимой в квазистационарных условиях. Из сказанного видно, что методы альтернативной адаптации, основанные на оперативном использовании информации о текущем состоянии управляемого объекта (процесса), могут быть успешно применены в децентрализованных сложных системах. передачи данных с высоким уровнем неопределенности. Рассмотрим этот вопрос подробнее.

В процессе передачи данных вероятность искажения сообщения в канале связи обычно изменяется непредсказуемым образом, вследствие чего любой заранее выбранный способ кодирования в канале связи наверняка теряет свою эффективность по отношению к другим, альтернативным кодам. Для учета этого обстоятельства предлагается использовать алгоритм

управления выбором кодов, адаптивно выбирающий лучший в сложившейся ситуации код из набора альтернативных кодов.

Как отмечается в ряде исследований [20, 51, 130, 244], анализ статистики ошибок в реальных каналах связи показал, что единичные ошибки (т. е. переход «1» в «0» или «0» в «1») имеют тенденцию группироваться в пачки ошибок. Уровень помех в каналах связи принято характеризовать вероятностью ошибки на бит (p_B) или вероятностью ошибки на символ передаваемой информации. Целью управления потоком передачи данных обычно является максимальное повышение скорости передачи данных, т. е. повышение эффективности использования канала связи при заданном уровне ошибок. Для достоверной передачи Данных в присутствии пачки ошибок требуется использовать код с высокой избыточностью, обнаруживающий и исправляющий ошибки, в то время как применение такого кода в интервалах между пачками ошибок приводит к снижению эффективности использования каналов. Отсюда возникает задача адаптивного управления потоком данных в условиях нестационарных помех. Эта задача для двуальтернативного случая может быть сформулирована в следующем виде.

Пусть имеется система передачи информации, которая может работать в полудуплексном режиме. Предполагается, что канал связи может находиться в одном из двух состояний — A или C — с соответствующими вероятностями ошибок на символ p_A и p_C. Принимая, что $p_C >> p_A$, будем считать, что состояние C соответствует наличию пачки ошибок в канале, а состояние A — отсутствию. Требуется построить адаптивную процедуру, управляющую потоком передачи данных так, чтобы повысить среднюю скорость (эффективность) передачи по каналу.

5.4.2. Двуальтернативный выбор кода

Для решения указанной задачи воспользуемся процедурой альтернативной адаптации, рассмотренной в § 5.1. В качестве альтернативных управлений выберем коды передачи данных с различной помехоустойчивостью: код с проверкой на четность, реализуемый в канале с переспросом, и код, исправляющий ошибки. Рассмотрим их свойства.

Как известно [20, 51, 76, 130], средняя скорость передачи данных при работе в режиме переспроса будет

$$R = \frac{k}{n}(1 - p_l)\nu \quad (5.4.1)$$

или же

$$R = \frac{k\upsilon}{(n+\tau)\, m_\lambda}, \quad (5.4.2)$$

где υ — скорость передачи модема (бит/с); k — число информативных бит в блоке; n — общее число бит в блоке; m_λ, — математическое ожидание кратности передачи блока; p_l — вероятность возникновения ошибки в блоке; τ — время задержки при переспросе. Здесь

$$p_l = 1 - (1 - p_B)^n, \quad (5.4.3)$$

где p_B — вероятность ошибки на бит.

Из этих выражений следует, что при изменениях состояния канала p_B изменяется R и путем выбора различной избыточности k/n можно адаптироваться к состоянию канала. Однако такие способы ограниченны, поскольку не позволяют широко варьировать характеристики кода.

Для кодов, исправляющих ошибки, скорость передачи постоянна (если не считать время на исправление ошибок) и равна

$$R' = \frac{k}{n}\upsilon, \quad (5.4.4)$$

причем $k<<n$, так как подобная избыточность обычно значительна.

Код с проверкой на четность менее избыточен, и его целесообразно применять при низком уровне помех, т. е. в состоянии A канала связи. Код с исправлением ошибок более помехозащищен, и его следует использовать при высоком уровне помех, т. е. в состоянии C канала.

Суть работы адаптивного кодирования, приспосабливающегося к состоянию канала в смысле минимизации затрат времени на единицу передаваемых данных, заключается в том, что алгоритм адаптации так изменяет вероятность выбора альтернативных кодов, чтобы в среднем чаще выбирался тот, который оптимален в данный момент.

Основные требования к алгоритму адаптивного выбора кода в канале связи можно сформулировать следующим образом:

— алгоритм должен оперативно реагировать на изменение состояния канала;

— работа алгоритма должна минимально нарушать нор мальный режим эксплуатации канала;

— алгоритм должен быть достаточно простым и быстродей ствующим.

Этим требованиям удовлетворяет алгоритм «многорукого бандита», рассмотренный в подразделе 5.1.5. Используем его для альтернативной адаптации кода в канале связи с изменяющимися свойствами.

Критерием эффективности Q в данном случае следует считать время, затрачиваемое на передачу одного бита полезной информации (избыточная информация при этом исключается).

Для проверки работоспособности этого алгоритма при альтернативной адаптации кодирования были проведены эксперименты на ЭВМ с двумя альтернативными кодами:

1. Пятиэлементный код с проверкой на четность. Кодовое слово содержало четыре бита полезной информации и один бит проверки на четность. Передаваемый блок состоял из 10 таких кодовых слов, или 50 бит. При обнаружении хотя бы одной ошибки весь блок передавался заново, т. е. реализовался ре жим переспроса.

2. Избыточный код (7,4) с исправлением ошибок [205]. Ко довое слово содержало четыре бита полезной информации и три бита служебной, избыточной, позволяющей исправлять одну ошибку в полезной информации. Передача проводилась бло ками по 10 кодовых слов, т. е. по 70 бит.

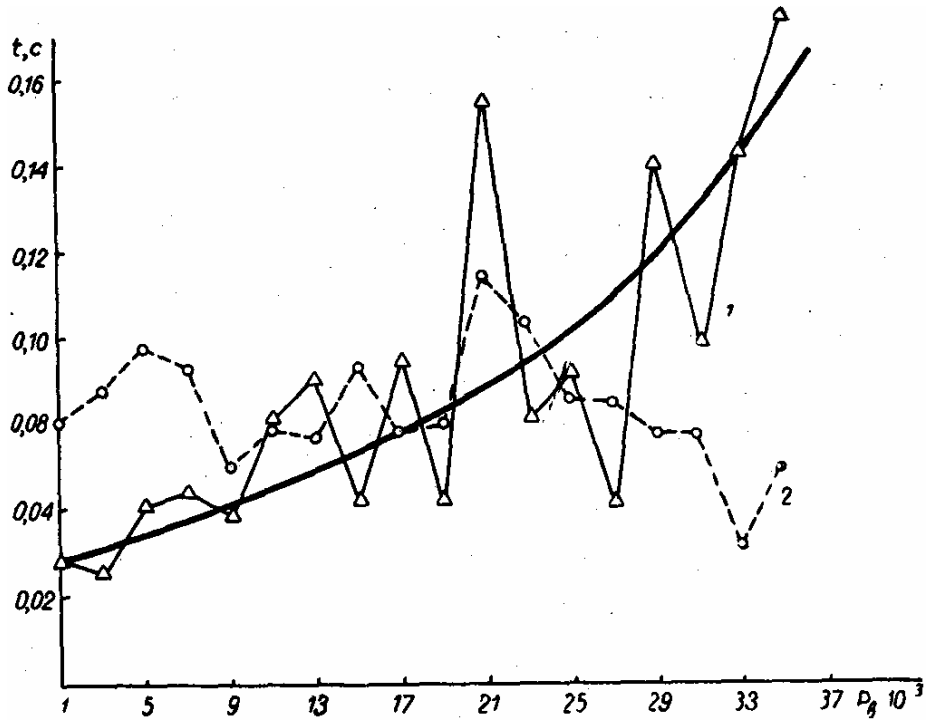


Рис. 5.4.1. Сравнительные характеристики эффективности работы двух кодов при различном уровне помех в канале связи:
1 — код с проверкой на четность, работающий в канале с переспросом; *2* — код с исправлением ошибок.

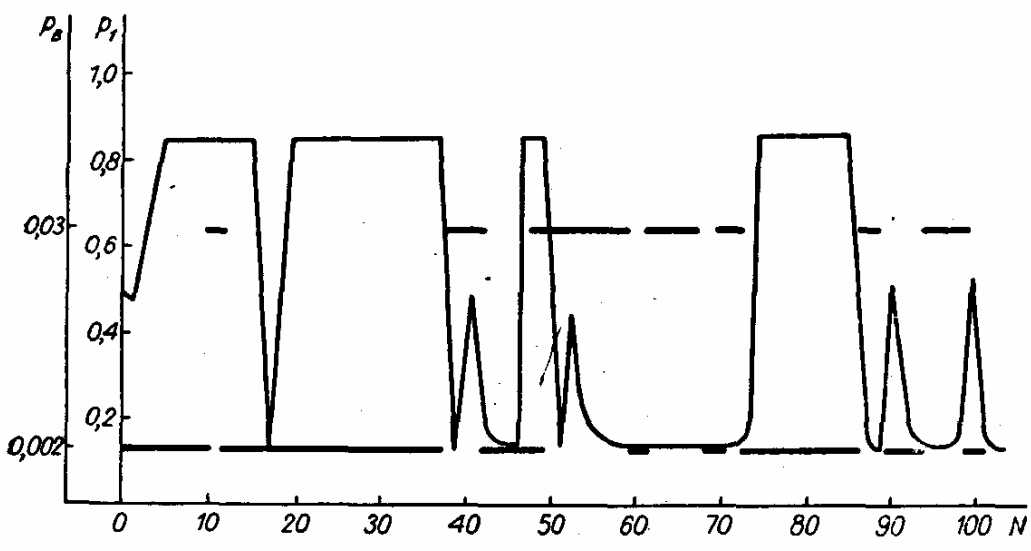


Рис. 5.4.2. Работа системы адаптации кодирования при двухуровневом изменении состояния канала.

Время передачи одного правильного блока, т. е. блока, который считался правильным (с учетом переспроса и исправления), фиксировалось таймером и определяло эффективность кода в сложившейся ситуации. На рис. 5.4.1 показаны экспериментально полученные характеристики среднего времени передачи указанных блоков с помощью выбранных кодов. Из рисунка видно, что при низком уровне помех код с проверкой на четность лучше кода с исправлением ошибок, а при высоком уровне (начиная с p_B=0,02) — наоборот.

Работа алгоритма адаптации показана на рис. 5.4.2. Состояние канала случайно изменялось на двух уровнях: p_A = 0,002 и p_C = 0,03. При этом параметр Сглаживания в выражениях (5.1.31) — (5.1.34) был выбран равным μ=0,4 и принимались меры против «передетерминирования» вероятности выбора одного из альтернативных кодов, т. е. соблюдалось условие 0,15≤ ≤p≤0,85. Хорошо виден адаптивный характер процесса. Алгоритм адаптации надежно отслеживает изменение состояния канала и переключает альтернативу примерно за четыре такта работы системы.

Результаты экспериментов показали, что использование адаптивного алгоритма выбора кодов в среднем на 20% повышает эффективность работы канала по сравнению с использованием в тех же условиях любого из альтернативных кодов в отдельности. Эти условия моделировались в виде марковского процесса, в котором вероятность изменения режимов *A* и *C* была равна 0,1.

5.4.3. Адаптация информационного поля протоколов связи

Связь между двумя ЭВМ в сети осуществляется кадрами. Рассмотрим применение альтернативных алгоритмов для адаптации длины информационного поля в протоколе второго уровня [153].

Пусть u — длина информационного поля кадра (в битах), а H — длина остальной части кадра по протоколу. Тогда при правильной передаче кадров время передачи одного информационного бита

$$q(u) = \frac{\Delta t^{(p)}}{pu}, \quad (5.4.5)$$

где $\Delta t^{(p)}$ — время передачи p кадров, включающее и затраты на подтверждение приема, и повторную передачу кадра при обнаружении ошибок. Это время пропорционально величине $u+H$.

Очевидно, что при наличии помех в канале связи $q(u)$ имеет минимум. Действительно, малая величина u не может быть оптимальной ввиду возрастания доли «накладных расходов» H, *а* при большом значении u увеличивается доля переспросов, что также увеличивает показатель (5.4.5).

Введем дискреты a_i (i = 1, ..., k) допустимой длины u информационной части кадра. Тогда, замеряя интервалы $\Delta t^{(p)}$, с помощью таймера можно оценить на каждом шаге эффективность канала, т. е. время передачи одного бита при неизвестном уровне помех в канале связи. Очевидно, что оптимальная длина u^* зависит от уровня помех и будет изменяться при изменении этого уровня. В результате получаем задачу адаптивного выбора величины информационной части кадра, минимизирующей время передачи полезной информации:

$$Q(u, t) = M_p q_t(u) \to \min_{u \in S} \Rightarrow u^*_t, \quad (5.4.6)$$

где S — область допустимых значений информационного поля кадра:

$$S: u \in \{a_1 < a_2 < \ldots < a_k\}. \quad (5.4.7)$$

Будем предполагать, что оптимум u^* случайно блуждает вдоль оси с малой скоростью — например, переходя от a_i к a_{i+1} за время по крайней мере на порядок больше, чем время одной оценки $Q(u, t)$. Модель такого медленного дрейфа экстремума вполне удовлетворительно описывает поведение канала связи (если, разумеется, исключить кратковременную нестационарность) .

Рассмотрим сначала простейшую схему накопления. Будем оценивать значение приращения

$$\Delta Q = Q(a_i, t) - Q(a_{i+1}, t') \qquad (5.4.8)$$

путем накопления оценок:

$$\Delta Q'_j = Q'(a_i, t_j) - Q'(a_{i+1}, t_{j+1}) \quad (j = 1,\ldots). \qquad (5.4.9)$$

Если $\Delta Q>0$, то $a_{i+1} \succ a_i$, и наоборот, при $\Delta Q<0$ имеет место $a_i \succ a_{i+1}$ Алгоритм адаптации при этом принимает естественный вид

$$a^{(N+1)} = \begin{cases} a^{(N)} = a_i & \text{при} \quad \Delta Q > 0; \\ a_{i+1} & \text{при} \quad \Delta Q < 0, \end{cases} \qquad (5.4.10)$$

где $a^{(N+1)}$ — значение параметра u на $(N+1)$-м этапе адаптации. Здесь значение ΔQ (точнее, его знак) можно оценить по-разному:

1. Фиксирование объема L накопления —

$$\operatorname{sign} \Delta Q = \operatorname{sign} \sum_{j=1}^{L} \Delta Q'_j. \qquad (5.4.11)$$

Очевидно, что с увеличением L вероятность ошибочного решения, т. е. вероятность того, что

$$\operatorname{sign} \Delta Q \neq \operatorname{sign} \Delta Q, \qquad (5.4.12)$$

уменьшается.

2. Последовательная автоматная модель оценки связана с вычислением суммы

$$S_l = \sum_{j=1}^{l} \operatorname{sign} \Delta Q'_j \qquad (5.4.13)$$

и принятием решения

$$\operatorname{sign} \Delta Q = \operatorname{sign} S_l \qquad (5.4.14)$$

при

$$| S_l | \geq A, \qquad (5.4.15)$$

где $A>0$ — заданное пороговое значение. Чем больше A, тем надежнее решение (5.4.14).

3. Другая последовательная модель оценки знака (см. § 5.2) имеет дело с рекуррентными вычислениями:

$$q_i = q_j\text{-}1 + \delta\,(1 - q_{j\text{-}1})\, q_{j\text{-}1} \operatorname{sign} \Delta Q'_j, \qquad (5.4.16)$$

где по построению $0<q<1$, и решение принимается следующим образом:

$$\operatorname{sign}\Delta\bar{Q}=\begin{cases}1 & \text{при } q\geqslant 1-\alpha;\\ -1 & \text{при } q\leqslant\alpha.\end{cases} \quad (5.4.17)$$

Очевидно, чем меньше $\alpha>0$, тем надежнее принятое решение, т. е. меньше вероятность ошибки (5.4.12).

Как видно, все эти алгоритмы имеют один параметр (L, A или α), определяющий надежность процедуры принятия решений, и одинаково часто обращаются к обеим конкурирующим альтернативам (a_i и a_{i+1}). Если первое обстоятельство (определение коэффициентов) легко преодолевается введением стандартных процедур оценивания статистических свойств случайных величин, то для второго необходимо изыскание новых алгоритмов. Особенно ясно это видно в случае двух альтернатив (k=2), когда ни одна из описанных процедур не обеспечивает преимущество ни одной из альтернатив, т. е. они вообще не решают задачу адаптации.

Алгоритм адаптации, решающий поставленную задачу, т. е. обеспечивающий более частое обращение к лучшей альтернативе, построен на основе вероятностного автомата с переменной структурой, рассмотренного в § 5.1. Этот алгоритм в своей основе имеет линейную тактику: повторять альтернативу при удаче и с определенной вероятностью переходить к конкурирующей альтернативе при неудаче. Однако здесь линейная тактика дополнена нелинейным решением: с вероятностью P при неудаче сохранять первую (a_i) альтернативу и с вероятностью $1\text{-}P$ — вторую (a_{i+1}).

Здесь величина $\Delta\tilde{Q}'_j$ определяется при каждом обращении к каждой альтернативе. Для $a = a_i$ в момент t

$$\Delta\tilde{Q}'_j=Q'(a_i, t_j)-Q'(a_{i+1}, \tilde{t}_j), \quad (5.4.18)$$

где $\tilde{t}_j$ — момент первого от t_j предыдущего измерения показателя при альтернативе a_{i+1}. Для $a = a_{i+1}$ в момент t_j

$$\Delta\tilde{Q}'_j=Q'(a_i, \tilde{\tilde{t}}_j)-Q'(a_{i+1}, t_j), \quad (5.4.19)$$

где $\tilde{\tilde{t}}_j$ — момент первого предыдущего наблюдения Q при a_i

Вероятность P в соответствии с изложенным в § 5.1 должна быть равна вероятности того, что $a_i \succ a_{i+1}$. В этом случае принимаемое решение оптимально по критерию минимума дисперсии оценки наилучшей альтернативы.

Вероятность P равна P = Вер $\{a_i \succ a_{i+1}\}$ = Вер $\{Q'_i < Q'_{i+1}\}$,

где Q'_i — случайная величина, равная оценке критерия при альтернативе a_i. Ее естественно считать нормальной с математическим ожиданием и дисперсией:

$$DQ'_i = \frac{1}{k_i - 1} \sum_{j \in Z_i} [Q'(a_i, t_i) - MQ'_i]^2, \quad (5.4.20)$$

где Z_i — множество номеров моментов времени, когда реализовалась i-я альтернатива a_i.

Вероятность P можно оценить теперь стандартным образом:

$$P = \frac{1}{2}\left[1 - \Phi\left(\frac{MQ'_i - MQ'_{i+1}}{2(DQ'_i + DQ'_{i+1})^{1/2}}\right)\right], \quad (5.4.21)$$

где Ф — функция Лапласа:

$$\Phi(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

Легко видеть, что при k_i, $k_{i+1} \to \infty$ получаем P=1 или P= = 0 — в зависимости от того, какая альтернатива оптимальна — i-я или (i+1)-я. В результате алгоритм детерминирует именно оптимальную альтернативу.

Если свойства альтернатив быстро дрейфуют во времени, то вместо вероятности (5.4.21) следует воспользоваться скользящими оценками типа

$$M_j Q'_i = \upsilon M_{j-1} Q'_i + (1 - \upsilon)\, Q'(a_i, t_j); \quad (5.4.22)$$
$$D_j Q'_i = \upsilon^2 D_{j-1} Q'_i + (1 - \upsilon)^2 [\, Q'(a_i, t_j) - M_j Q'_i \,]^2,$$

где M_j и D_j — операторы оценок математического ожидания и дисперсии в j-й момент времени.

Эксперименты, проведенные на модели канала связи с помощью последнего алгоритма адаптации, показали высокую эффективность предложенного подхода. Эффективность оценивалась при блуждающей вероятности ошибки в канале связи по сравнению с u=const при настройке на среднюю вероятность ошибки, что, таким образом, обеспечивало оптимальность в среднем. При медленном изменении вероятности ошибки описанный алгоритм адаптации поддерживал оптимальную длину

информационного поля u^* и был всегда лучше неадаптируемого лротокола по критерию (5.4.5). С возрастанием скорости изменения вероятности ошибки эффективность адаптивного протокола падала и в пределе совпадала с эффективностью протокола, оптимального в среднем [194].

§ 5.5. Адаптация процессов сортировки массивов

Известно, что существует большое количество методов сортировки массива [105, 124, 148]. Они отличаются различными алгоритмами, условиями целесообразного применения и техникой реализации массива (на магнитных дисках, лентах и т. д.). Формальные признаки массива позволяют несколько снизить разнообразие возможных методов сортировки, но не снимают проблему выбора. Если в распоряжении пользователя имеется несколько альтернативных программ сортировки, относительно выбора одной из которых у него нет априорных соображений, то ему следует воспользоваться методом альтернативной адаптации.

Рассмотрим процедуру адаптивного выбора программы сортировки из набора альтернативных по критерию быстродействия, т. е. по среднему времени, затрачиваемому на сортировку одного элемента массива.

Здесь следует отметить, что успешность адаптации существенно зависит от наличия устойчивых статистических свойств поступающих массивов, которые могут медленно изменяться во времени. Если это статистическое свойство (неважно, какое) априори обеспечивает одному из методов сортировки преимущество по выбранному критерию, то алгоритм альтернативной адаптации должен в процессе работы чаще выбирать именно этот метод.

Для простоты определим это устойчивое статистическое свойство массива в виде его упорядоченности. Воспользуемся следующей моделью образования массива, которая необходима не только для моделирования, но и для кодирования массива по критерию упорядоченности:

$$a_{i+1} = a_i + c + \varepsilon_i \quad (i = 1, \ldots, N\text{-}1), \tag{5.5.1}$$

где a_i — i-й элемент массива (a_0=0); ε_i — случайное число с равномерным законом распределения (—l≤ε_i≤l); —l≤c≤ ≤l — коэффициент упорядоченности массива. При с = 0 массив полностью не упорядочен, при c= —1 он имеет обратную упорядоченность:

$$a_{i+1} < a_i \quad (i = 1, \ldots, N\text{-}1), \tag{5.5.2}$$

а при c=1 массив полностью упорядочен:

$a_{i+1} > a_i$ (i=l, ..., N-1). . (5.5.3)

Рассмотрим для конкретности два алгоритма сортировки:

1. Алгоритм «min». В соответствии с этим алгоритмом на каждом j-м этапе сортировки определяется минимальный эле мент массива (a_i^j), который переносится j-м в новый список, а оставшийся массив перенумеровывается. Иначе говоря, мас сив укорачивается на единицу так, что первые i—1 элементов, сохраняются:

$a_k^{j+1} = a_k^j$ (k=1, ..., i—l), (5.5.4)

а последующие смещаются влево на единицу:

$a_l^{j+1} = a_{l+1}^j$ (l = 1, ..., N-j). (5.5.5)

Легко видеть, что этот алгоритм быстрее сортирует массивы, близкие к обратноупорядоченным, чем к прямоупорядоченным (за счет уменьшения числа процедур смещения).

2. Алгоритм «пузырек». Этот алгоритм заключается в после довательном сравнении соседних элементов массива и их об мене при $a_i > a_{i+1}$. Критерием остановки является отсутствие обменов при просмотре всего массива. Алгоритм быстрее ра ботает на хорошо упорядоченных массивах.

На рис. 5.5.1 показаны временные характеристики обоих алгоритмов при сортировке массивов различной упорядоченности объемом N=500 на ЭВМ ЕС-1030. Из рисунка видно, что при c>0 следует использовать алгоритм «пузырек», а при c<0 — «min».

Используем для альтернативной адаптации сортировки алгоритм «двурукого бандита», описанный в подразделе 5.1.2.3. На рис. 5.5.2, а представлена динамика переключения с одного алгоритма, оптимального, на другой при скачкообразном изменении упорядоченности сортируемых массивов от c= = —0,8 до c=0,8. Видно, что вероятность выбора алгоритма быстро перестраивается и за одну-две сортировки

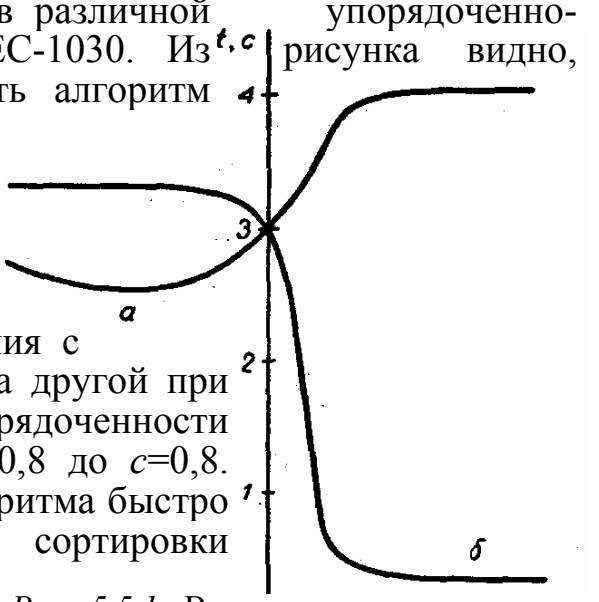


Рис. 5.5.1. Временные характеристики сортировки при различной упорядоченности массива алгоритмами «min»- (а) и «пузырек» (б).

алгоритм

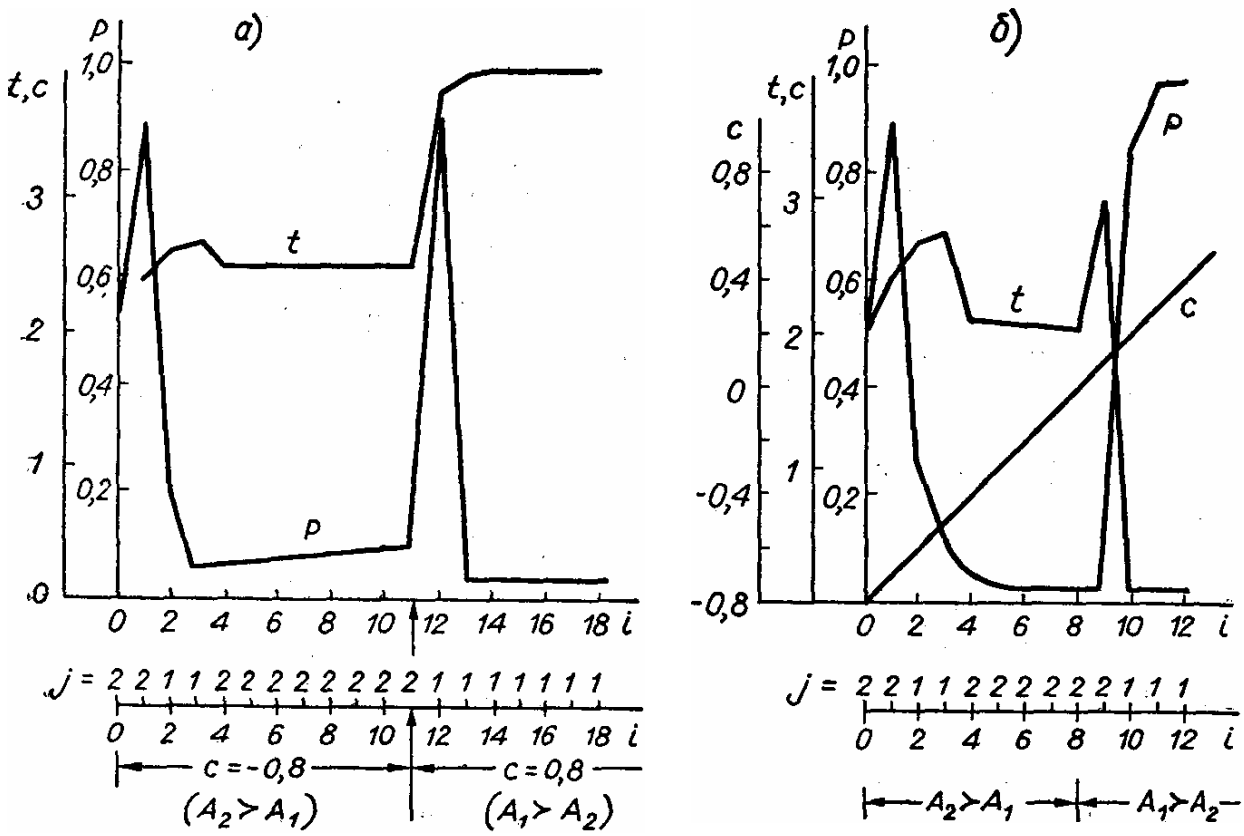


Рис. 5.5.2. Динамика работы алгоритма альтернативной адаптации процесса сортировки:
а — при скачкообразном изменении коэффициента упорядоченности массивов, *б* — при линейном изменении.

начинает устойчиво выбирать оптимальный в данной ситуации алгоритм («min» при c = —0,8 и «пузырек» — при c = 0,8).

Динамика адаптации (рис. 5.5.2, б) при непрерывном измелении во времени упорядоченности сортируемых массивов:

$$c\,(t) = 0{,}2\,k. \qquad (5.5.6)$$

где k — номер очередного массива, показывает, что алгоритм адаптации уверенно «включает» алгоритм, оптимальный в сложившейся ситуации c.

Таким образом, предложенный алгоритм альтернативной адаптации сортировки массивов позволяет надежно определять наилучший в сложившейся ситуации алгоритм сортировки и переходить к другому, лучшему при изменении ситуации.

Описанные алгоритмы альтернативной адаптации применимы лишь в том случае, когда число конкурирующих альтернатив крайне мало (2—5). Если их число велико, то надо использовать алгоритмы эволюционной адаптации, описанные в следующей главе.

Глава 6

Эволюционная адаптация

Ничто в мире не может вдруг объявиться в конце после ряда совершаемых эволюцией переходов, если оно незаметно не присутствовало в начале.

Пьер Тейяр де Шарден

Идеи биологической эволюции, рассмотренные в § 3.7, дают богатую пищу для структурной адаптации сложных систем. При этом структуру объекта следует описать на языке графов, эволюция которых реализуется прямым моделированием биологической эволюции, т. е. введением случайных мутаций и процедуры селекции, позволяющей отбирать на следующий этап эволюции лучшие структуры. Реализация такой эволюции на графах получила название эволюционного моделирования.

В этой главе рассмотрены основные алгоритмы эволюционной адаптации и описано их применение для решения задач агрегации графов, адаптации дисциплин обслуживания в вычислительных сетях, адаптации структуры решающих правил и синтеза оптимальных планов эксперимента для факторной модели объекта.

§ 6.1. Алгоритмы эволюционной адаптации

Эти алгоритмы являются квазибионическими алгоритмами случайного поиска, параметрические аналоги которых рассмотрены в § 3.7. Рассмотрим теперь эволюцию структуры.

6.1.1. Общая модель эволюции структуры

Начнем с модели структурной эволюции, которая лежит в основе структурной адаптации.

Пусть структура W объекта может изменяться, причем эти изменения, т. е. вариации (мутации) δW структуры W, принадлежат заданному множеству Ξ возможных вариаций:

$$\delta W \in \Xi. \qquad (6.1.1)$$

Множество Ξ определяется ограничениями H и G (см. § 1.2) так, что при соблюдении (6.1.1) выполняются условия

$$S: \begin{cases} H(W+\delta W) \geqslant 0; \\ G(W+\delta W) = 0, \end{cases} \quad (6.1.2)$$

если $H(W) \geq Q$ и $G(W)=0$, где H и G — заданные функционалы ограничений.

Таким образом, будем считать, что множество S вариаций δW определено и задача состоит в оптимизации заданного функционала:

$$Q(W) \to \min_{W \in S} \quad (6.1.3)$$

определенного на структуре W.

Процесс эволюции структуры W происходит поэтапно. На первом этапе порождаются возмущенные (мутированные) структуры:

$$W_{0i} = W_0 + \delta W_i \quad (i = 1, \ldots, k_0), \quad (6.1.4)$$

где δW_i — i-я случайная вариация (мутация) структуры, ограниченная (6.1.1), а число новых структур k_0 является параметром, который назначается исходя из конкретных условий эволюции.

Новые структуры (6.1.4) оцениваются по критерию эффективности

$$Q_{0i} = Q(W_{0i}) \quad (i = 1, \ldots, k_0), \quad (6.1.5)$$

и происходит отбор, в процессе которого «вымирают» структуры с большим значением минимизируемого функционала Q, в результате чего на следующий этап эволюции остается q_0 лучших структур. Можно воспользоваться алгоритмом вероятностного отбора, при котором структура, имеющая большое значение минимизируемого критерия, выбывает с большей вероятностью, чем структура с меньшим критерием. Например, вероятность выбывания для $Q>0$ может быть определена так:

$$p_{0i} = \frac{Q_{0i}}{\sum_{i=1}^{k_0} Q_{0i}} \quad (i = 1, \ldots, k_0) \quad (6.1.6)$$

При этом процесс «разыгрывания» выбывающих структур заканчивается тогда, когда осталось q_0 структур. Так моделируется естественный отбор, составляющий основу биологической эволюции.

Заметим, что вполне может оказаться (особенно при малом k_0), что лучшая из новых структур хуже исходной W_0 В этом случае естественно сохранить W_0 на следующий этап эволюции.

На втором этапе эволюции каждая из оставшихся структур мутирует аналогично (6.1.4) и дает столько «потомков», что их общее число вместе с «родителями» равно k_1. Последующий отбор оставляет на следующий этап эволюции q_1 структур. Далее процесс повторяется.

Легко видеть, что такого рода эволюция структуры будет «стремиться» отбирать структуры с малым значением критерия качества, среди которых находится и оптимальная структура. Случайность вариации δW обеспечивает сходимость процесса эволюции к оптимальному решению W^*.

Рассмотрим влияние параметров k_i и q_i (i = 0, 1, ...). Эти параметры позволяют изменять численность «популяции» структур и уровень отбора. При q=1 на следующий этап эволюции оставляется одна лучшая структура. Такая стратегия эффективна при «унимодальности» задачи (6.1.3). Многоэкстремальность требует q_i>1 и тем больше, чем сложнее поиск глобального экстремума. Численность популяции k_i также влияет на . эффективность процесса эволюции. При большем k_i эволюция имеет глобальную тенденцию, но идет медленнее, так как требует значительных затрат времени и памяти ЭВМ.

Значение этих параметров следует адаптировать в процессе эволюции, учитывая ее специфику, цели и результаты. Алгоритмы адаптации должны иметь сугубо эвристический характер.

Заметим, что немаловажную роль играет эффективность оценки структуры. Эта оценка, как правило, бывает очень трудоемкой и сводится или к вычислению многомерных интегралов, или к оценке функционирования имитационной модели с назначенной структурой. В обоих случаях удобно воспользоваться методом Монте-Карло (см. § 3.2).

Здесь описывается адаптация без параметрической подстройки (см. § 1.5). Параметрическая адаптация, если она необходима для повышения эффективности структур, легко вводится перед стадией отбора и осуществляется стандартными параметрическими методами (см., например, § 3.5).

Рассмотрим в качестве примера и наиболее распространенного случая объекты, структура которых описывается графом.

6.1.2. Эволюционная адаптация графа

Пусть структура W описывается графом

$$\Gamma = \langle A, B \rangle. \qquad (6.1.7)$$

Здесь A — множество из n вершин, а B — множество ребер с их параметрами:

$$B = \| b_{ij} \|_{n \times n}. \qquad (6.1.8)$$

где b_{ij} — параметры дуги, соединяющей i-ю вершину с j-й.

На графе Г задан функционал качества, который следует минимизировать:

$$Q(\Gamma) \to \min_{\Gamma \in S}, \qquad (6.1.9)$$

где S — ограничения, которым должен удовлетворять адаптируемый граф Г в соответствии с условиями (6.1.2).

Опишем множество Ξ возможных вариаций графа $\delta\Gamma$ (6.1.1). Оно может состоять, например, из следующих изменений:

$$\Xi = (\xi_1, ..., \xi_p), \qquad (6.1.10)$$

где ξ_1 — объединение двух случайно выбранных вершин графа в одну; ξ_2 — введение новой $(n+1)$-й вершины со случайными связями $b_{n+1,i}$ $(i = i_1, ..., i_k)$ и $b_{j,n+1}$ $(j = j_1, ..., j_q)$; ξ_3 — устранение случайно выбранной вершины вместе с ее связями; ξ_4 — введение новой связи двух случайно выбранных вершин; ξ_5 — устранение случайно выбранного ребра; ξ_6 — случайное «переключение» случайно выбранной дуги, и т. д.

Как видно, спектр возможных случайных мутаций графа может быть достаточно велик, что обеспечивает эволюции большое разнообразие, необходимое для отыскания оптимального графа.

Значения параметров дуг графа (6.1.8) могут подстраиваться специально на стадии параметрической адаптации, предшествующей отбору лучших структур.

6.1.3. Эволюционная адаптация автомата

Другим важным примером является адаптация автомата. Пусть детерминированный конечный автомат, описывающий поведение адаптируемого объекта, имеет вид шестерки:

$$\langle X, Z, Y, \mu(x, z), \upsilon(x, z), z_0 \rangle, \qquad (6.1.11)$$

где $X = \{x_1, ..., x_n\}$ — алфавит входов и $Y = \{y_1, ..., y_m\}$. — алфавит выходов, заданные связями автомата со средой; $Z = = \{z_1, ..., z_l\}$ — алфавит внутренних состояний автомата; $z' = = \mu(x, z)$ — функция переходов, определяющая состояние, в которое переходит автомат из состояния a при входе x; $\upsilon(x, z) =$

$= y$ — функция выхода, определяющая выход автомата при состоянии z и входе x; z_0 — начальное состояние автомата.

В автомате можно изменять алфавит состояний Z, начальное состояние z_0 и функции перехода $\mu(\cdot,\cdot)$ и выхода $\upsilon(\cdot,\cdot)$. Задача оптимизации заключается в том, чтобы, варьируя эти факторы, построить автомат, минимизирующий функционал, заданный на изменяющихся состояниях автомата, т. е. решить задачу

$$Q(y(x)) \to \min_{Z, z_0, \varphi, \psi}. \quad (6.1.12)$$

Для вычисления этого функционала необходимо иметь последовательность состояний среды $x_1, \ldots, x_N, \ldots$, которая образует вход автомата.

Случайный поиск оптимального автомата, минимизирующего заданный функционал, образуется путем использования операций преобразования графа автомата, в котором вершинами являются внутренние состояния $z_1, \ldots, z_l$, а направленными дугами — условные переходы, определяемые функцией перехода $\mu(x, z)$. Очевидно, что операции преобразования этого графа обеспечивают соответствующее изменение автомата.

При оптимизации структуры автомата случайные вариации его структуры образуются, например, следующими операциями над графом, аналогичными перечисленным в п. 6.1.2:

ξ_1: введение новой вершины z_{l+1} со случайными дугами, связывающими ее с другими вершинами, т. е. случайное задание функции $\mu(x, z_{l+1})$;

ξ_2: устранение случайно выбранной вершины z_i ($i = 1, \ldots, l$) вместе с ее дугами;

ξ_3: введение новой дуги, связывающей две случайно выбранные вершины z_i и z_j, т. е. задание функций $z_i = \mu(x, z_j)$ и $z_j = = \mu(x, z_i)$;

ξ_4: устранение случайно выбранной дуги;

ξ_5: случайное «переключение» случайно выбранной дуги, т. е. переход от $z_i = \mu(x, z_j)$ к $z_k = \mu(x, z_j)$, где z_i, z_j и z_k — случайные вершины.

Как видно, реализация одного из указанных операторов-мутаций несколько изменяет автомат и тем самым — значение минимизируемого функционала. Естественно те мутации, которые уменьшают значение функционала, считать благоприятными, а мутации, увеличивающие его, — неблагоприятными, что и использует процесс отбора.

Параметрическая адаптация, предшествующая этому отбору, состоит в определении начального состояния z_0 и функции выхода, задаваемой значением выхода автомата y, соответствующего паре ‹x, z›, что эквивалентно заданию функции выхода $\upsilon(\cdot,\cdot)$. Это означает, что алфавит У автомата и его начальное

Состояние z_0 определяются на стадии параметрического синтеза, т. е. методами параметрической адаптации (непрерывной и дискретной), рассмотренной в главе 4.

Направление эволюционной адаптации интенсивно развивается в настоящее время и получило наименование эволюционного моделирования [36, 126, 127], что связано с оценкой функционала (6.1.12) путем моделирования поведения автомата в заданной среде.

§ 6.2. Адаптивная агрегация графов [291] 6.2.1.

Постановка задачи

Пусть имеется ориентированный граф, определяемый парой

$$\Gamma = \langle A, B \rangle. \qquad (6.2.1)$$

Здесь A — конечное множество вершин графа, каждая из которых имеет вес

$$A = \{a_1, ..., a_n\}; \qquad (6.2.2)$$

$a_i > 0$ — вес i-й вершины; B — множество ребер, каждое из которых определяется своим весом. Эти веса образуют матрицу

$$B = \| b_{ij} \|_{n \times n}, \qquad (6.2.3)$$

где $b_{ij} \geq Q$ — вес ребра, связывающего i-ю вершину с j-й. Нулевой вес ($b_{ij} = 0$) соответствует отсутствию ребра.

В общем случае предполагаем, что веса вершин и ребер зависят от состояния X среды, в которую погружен объект, описываемый графом, т. е.

$$a_i = a_i(X) \qquad (6.2.4)$$
$$b_{ij} = b_{ij}(X).$$

Это состояние изменяется во времени непредсказуемым об разом, т. е.

$$X = X(t). \qquad (6.2.5)$$

Введем агрегированный граф

$$\Gamma' = \langle A', B' \rangle, \qquad (6.2.6)$$

который получается из Г (6.2.1) путем объединения его вершин:

$$A' = (a'_1, ..., a'_m), \qquad (6.2.7)$$

где $m<n$;

$$a'_l = \sum_{i \in I_l} a_i \qquad (l = 1, \ldots, m);$$

/г — множество номеров вершин графа Г, объединенных (агрегированных) в одну вершину a'_l графа Г′ ($\bigcup_{l}^{m} I_l = \{1, \ldots, n\}$, $I_l \cap \cap I_k = 0$ при $l \neq k$). $B' = \| b'_{ij} \|_{m \times m}$ (6.2.8)

— матрица графа Г′, где

$$b'_{lk} = \sum_{i \in I_l} \sum_{j \in I_k} b_{ij} \qquad (l, k = 1, \ldots, m) \quad (6.2.9)$$

Как видно, агрегация определяется однозначно:

$U = (I_1, .., I_m)$. (6.2.10)

Задача оптимальной агрегации заключается в минимизации суммы весов ребер агрегированного графа

$$Q(U, X) = \sum_{l \neq k}^{m} b'_{lk}(U, X) \quad (6.2.11)$$

при соблюдении ограничений на веса его вершин:

$a'_l (U, X) \leq c_l \quad (l = 1, ..., k)$, (6.2.12)

где

$C = (c_1, ..., c_m)$ (6.2.13)

— заданные верхние границы весов агрегированного графа Г′, которые определяются выделенными ресурсами.

Таким образом, для поиска оптимальной агрегации необходимо решить следующую задачу минимизации:

$$Q(U, X) \to \min_{U \in S} \Rightarrow U^*(X), \quad (6.2,14)$$

где ограничения S имеют вид

$S : A'(U, X) \leq C$. (6.2.15)

Решение этой задачи U^* естественно назвать оптимальной агрегацией. Она должна изменяться во времени ввиду изменения свойств среды X, влияющей на веса вершин и ребер (6.2.4) ис-ходногр графа.

Задача адаптации заключается в поддержании агрегации в оптимальном состоянии, зависящем от изменяющихся свойств среды X, т. е.

$$U^* = U^*(X). \qquad (6.2.16)$$

Если свойства среды неизменны, т. е. X=const, то задача адаптации сводится к оптимизационной задаче (6.2.14), которую обычно называют *задачей о разрезании графа.*

При медленном изменении среды адаптивная агрегация может быть сведена к последовательному повторению решения задачи о разрезании графа. Именно поэтому задаче адаптивной агрегации графа предшествует задача о его разрезании. Рассмотрим ее специфику.

Прежде всего следует отметить неоднозначность решения U^* данной задачи. Действительно, решением является любая перестановка полученной оптимальной агрегации (6.2.10), так как оптимальность не зависит от нумерации вершин агрегированного графа. Это сразу значительно увеличивает число оптимальных вариантов агрегации, что очень важно при использовании алгоритмов случайного поиска.

Далее, рассматриваемая задача является типичной задачей структурной адаптации, поскольку полностью отсутствуют непрерывные составляющие в решении U^*.

И наконец, это типичная задача эволюционной адаптации (см. § 1.5), так как при незначительном изменении агрегации (например, при «перебросе» одной вершины) минимизируемая функция изменится также незначительно, что позволяет воспользоваться поисковым методом при отыскании оптимальной агрегации.

Рассмотренная постановка задачи имеет детерминированный характер, так как изменение состояния среды X однозначно определяет параметры (6.2.4) задачи.

Однако часто имеет место стохастическая среда, состояние которой изменяется определенным, но известным стохастическим образом $p(X)$, где $p(\cdot)$ — плотность распределения измеряемых состояний среды. При этом естественно осреднить критерий (6.2.11) и ограничения (6.2.12) по X:

$$\bar{Q}(U) = \int Q(U, X)\, p(X)\, dX. \qquad (6.2.17)$$

Теперь задача синтеза оптимальной агрегации U^* принимает вид

$$\bar{Q}(U) \to \min_{U \in S} \Rightarrow U^*, \qquad (6.2.18)$$

где ограничение S может иметь различный вид. Например: ограничение путем осреднения

$$\bar{S}: \bar{A}'(U) \leqslant C, \qquad (6.2.19)$$

где

$$\bar{A}'(U) = (\bar{a}'_1(U), \ldots, \bar{a}'_m(U)) \tag{6.2.20}$$

и

(6.2.21)

ограничение по максимуму

$$\bar{S}: \bar{A}'(U) \leqslant C, \tag{6.2.22}$$

где

$$\bar{a}'_l(U) = \int a'_l(U, X)\, p(X)\, dX \qquad (l = 1, \ldots, m); \tag{6.2.23}$$

$$\hat{A}(U) = (\hat{a}'_1(U), \ldots, \hat{a}'_m(U))$$

и

$$\hat{a}'_l(U) = \max_X a'_l(U, X) \qquad (l = 1, \ldots, m), \tag{6.2.24}$$

и т. д. — в зависимости от содержательной постановки решаемой задачи.

Задачу (6.2.18) называют задачей стохастического программирования: ее минимизируемый критерий $\bar{Q}$ и ограничения $\bar{S}$ являются стохастическими характеристиками, зависящими от свойств среды, т. е. от $p(X)$. Неопределенность свойств среды исключает использование методов математического программирования и заставляет искать другие пути решения задачи (6.2.18). Адаптация является, пожалуй, единственным надежным способом решения этой сложной задачи.

6.2.2. Примеры практических задач агрегации графа

6.2.2.1. Компоновка электронной аппаратуры

Эта техническая задача элементарно сводится к задаче о разрезании неориентированного графа в детерминированной постановке. Вершинами графа здесь являются элементы схемы (емкости, резисторы, большие интегральные схемы и т. д.), а ребрами — провода, соединяющие эти элементы. Весами вершин (6.2.2) являются площади или объемы элементов схемы (в зависимости от целей агрегации), а веса (6.2.3) ребер агрегируемого графа обычно равны единице, если связь имеется, и нулю — при отсутствии электрической

связи между соответствующими элементами. Матрица (6.2.3) здесь симметрична. Веса вершин (6.2.13) агрегированного графа Г′ назначаются исходя из имеющихся ресурсов эксплуатационных объемов и площа-

дей. Например, при компоновке схемы в процессе ее разделения на блоки значения c_i (i = 1, ..., m) определяются допустимыми размерами блоков, плат, сборок, и т. д.

6.2.2.2. Адаптивная сегментация программного обеспечения вычислительной системы*

В этом случае следует воспользоваться стохастической постановкой задачи. Здесь агрегированными вершинами a'_i являются сегменты матобеспечения, расположенные во внешней памяти ВС. Размеры этих сегментов ограничены объемом оперативной памяти процессора. В этом случае $c_l = c$ (l = 1, ..., m), где c — часть объема памяти, выделяемой для информации, поступающей из внешней памяти, и ограничения имеют детерминированный характер.

Вероятностные свойства среды X здесь определяют обращаемость к переходу от одного блока информации к другому. Эти свойства в простейшем случае определяются стохастической матрицей

$$P = || p_{lk} ||_{m \times m} \qquad (6.2.25)$$

где p_{lk} — вероятность перехода от l-го блока информации к k-му. Располагая этой информацией, естественно считать, что $B = P$, т. е. матрица связей блоков информации совпадает со стохастической матрицей (6.2.25), и нет необходимости производить осреднение (6.2.17), так как оно уже сделано при определении матрицы (6.2.25). Тогда при известной матрице переходов (6.2.25) задача оптимальной сегментации принимает детерминированный характер. Решение этой задачи методами адаптации будет рассмотрено ниже.

6.2.2.3. Адаптация расположения блоков информации на магнитных дисках**

Эта задача очень близка к предыдущей, так как рассматривает процесс минимизации времени на обращение к внешней памяти. Минимизируется здесь среднее время движения магнитных головок записи-считывания.

Пусть m — число цилиндров магнитного *барабана;* a_l — информация, располагаемая на l-м цилиндре ($a_l < c$, где c — информационный объем цилиндра); p_l — вероятность обращения

* См. подраздел 2.2.5.1.
** См. подраздел 2.2.5.2.

к информации, расположенной на l-м цилиндре. Среднее время перемещения головки от одного цилиндра к другому

$$Q = \sum_{l,k=1} p_l p_k \varphi(l, k) \quad (6.2.26)$$

где $\varphi(l,k)$ — время перемещения головки от l-го цилиндра к k-му (например, в простейшем случае $\varphi(l, k) = w\ |l—k|$, где w — коэффициент размерности).

Обозначим буквой U вариант расположения информации, образующийся из исходного с помощью перестановки:

$$\begin{pmatrix} 1 & 2 & \dots & m \\ u_1 & u_2 & \dots & u_m \end{pmatrix}, \quad (6.2.27)$$

где $u_l \in \{1, \dots, m\}$ и $u_l \neq u_k$ $(l \neq k)$.

Выражение (6.2.27) означает, что информация a_l с l-го цилиндра переводится на k-й. Таким образом, адаптация как процесс управления в данном случае однозначно определяется следующим целочисленный вектором:

$$U = (u_1, \dots, u_m) \quad (6.2.28)$$

с различными компонентами из отрезка натурального ряда От 1 до т. Задачу синтеза оптимального расположения информации на магнитных цилиндрах теперь можно сформулировать как задачу целочисленной минимизации:

$$Q(U, P) = \sum_{l,k=1}^{m} p_{u_l} p_{u_k} \varphi(u_l, u_k) \to \min_{U \in S} \Rightarrow U^*, \quad (6.2.29)$$

где

$$P = (p_1, \dots, p_m) \quad (6.2.30)$$

— вектор вероятностей обращения к блокам информации $a_l, \dots, a_m$ $\left(\sum_{l=1}^{m} p_l = 1\right)$,

$$S: \begin{cases} u_l \in \{1, \dots, m\} & (l = 1, \dots, m); \\ u_l \neq u_k \quad \text{при} \quad l \neq k. \end{cases} \quad (6.2.31)$$

Очевидно, что оптимальное расположение U^* зависит от состояния среды — вектора P:

$$U^* = U^*(P). \quad (6.2.32)$$

При известном векторе P задача адаптации, как видно, сводится к задаче оптимизации (6.2.29), т. е. является детерминированной.

При неизвестном или изменяющемся P приходится пользоваться его оценкой

$$\hat{P} = (\hat{p}_1, \ldots, \hat{p}_m) \quad (6.2.33)$$

на базе конечного времени наблюдения за процессом работы магнитного диска. Процесс адаптации должен быть построен так, чтобы последовательность получаемых при этом перестановок

$$U_1, \ldots, U_N, \ldots, \quad (6.2.34)$$

каждая из которых использует последнюю оценку (6.2.33), отражала оптимальное расположение информации на дисках по критерию (6.2.29) в каждый конкретный момент времени.

Легко заметить, что полученная задача не является задачей о разрезании графа — тут ничего не «разрезается» и не агрегируется. Эта задача ближе к так называемой задаче о назначениях (см., например, работу [126], с. 158—164). Ее можно сформулировать следующим образом.

Введем двоичную переменную u_{lk} такого рода: $u_{lk}=1$ обозначает расположение l-го блока информации a_l на k-м цилиндре ($l, k = 1, \ldots, m$) и $u_{lk} = 0$ в противном случае. Тогда минимизируемый функционал (6.2.26) можно записать в форме

$$Q(U) = \sum_{i,j,l,k} u_{lk} u_{ij} p_i p_l \varphi(l, k). \quad (6.2.35)$$

При этом должны выполняться следующие очевидные ограничения:

$$S: \begin{cases} \sum_{l=1}^{m} u_{lk} = 1; \\ \sum_{k=1}^{m} u_{lk} = 1; \\ u_{lk} \in \{0; 1\} \qquad (k, l = 1, \ldots, m). \end{cases} \quad (6.2.36)$$

Первое из них выражает очевидное требование, чтобы на каждом k-м цилиндре располагался только один информационный блок. Второе ограничение связано с тем, что каждый l-й блок должен располагаться только на одном цилиндре, а третье определяет бинарность искомого вектора

$$U = (u_{11}, u_{12}, \ldots, u_{mm}), \quad (6.2.37)$$

содержащего m^2 компонент.

Задача адаптации в этом случае имеет вид

$$\sum_{i,j,l,k} u_{lk}u_{ij}p_i(X)\,p_l(X)\varphi(l,k) \to \min_{U\in S} \Rightarrow U^*(X), \tag{6.2.38}$$

где отмечена зависимость вероятностей от состояния среды X.

6.2.2.4. Обобщение предыдущей задачи

Это случай, когда на каждом цилиндре магнитного диска располагается не один блок информации, а много [277]. Тогда задача оптимального расположения блоков информации на магнитных дисках становится эквивалентной задаче об агрегации графа (6.2.1) — с той лишь разницей, что критерий оптимальности записывается не в форме (6.2.11), а в виде

$$Q(U,X)=\sum_{l,k=1}^{m} b'_{lk}(U,X)\varphi(l,k). \tag{6.2.39}$$

Здесь U — агрегация в виде (6.2.10), указывающая на расположение информационных блоков a_i (i=1,...,m). В процессе адаптации решается следующая задача:

$$\sum_{l,k=1}^{m} b'_{lk}(U,X)\varphi(l,k) \to \min_{U\in S} \tag{6.2.40}$$

где

$$S:\ \begin{cases} a'_l < c & (l=1,\ldots,m); \\ I_k \cap I_l = 0 & (k\neq l). \end{cases} \tag{6.2.41}$$

Последнее условие в выражении (6.2.41) означает, что при агрегации каждый информационный блок попадает только на один цилиндр. Вообще говоря, это условие может быть снято, но тогда решаемая задача перестанет быть задачей о разрезании графа, так как некоторые вершины как бы «расщепляются» в процессе агрегации.

6.2.3. Адаптивная агрегация графа методами случайного поиска

6.2.3.1. Методы агрегации (разрезания) графа

Существует целый ряд методов агрегации (разрезания) графа; их подробный обзор приводится в работе [111]. Эти методы подразделяются на детерминированные и стохастические (типа случайного поиска).

Детерминированные методы используют для разрезания графа известный метод ветвей и границ [27, 66, 201], а также различного рода эвристики, которые позволяют эффективно решать поставленную задачу [1, 2, 38, 106, 129, 147, 224]. Однако такие методы не позволяют решать глобальные задачи большой размерности, т. е. определяют лишь одно локальное решение. Именно это обстоятельство заставляет искать стохастические подходы к решению задачи о разрезании графа большой размерности.

Стохастический подход к разрезанию графа связан с намеренным введением элемента случайности в процесс разрезания. Это может быть сделано по-разному. Простейшей схемой является рандомизация перебора [37, 53, 97].

Рассмотрим стохастические методы, решающие задачу разрезания графа методом случайного поиска, т. е. путем введения случайного шага в процесс разрезания с последующей оценкой его эффективности.

6.2.3.2. Операции преобразования агрегации

Введем операции преобразования агрегации U:

$$U_{N+1} = \psi_{N+1}(U_N), \tag{6.2.42}$$

где U_N — агрегация на N-м этапе, а ψ_{N+1} — оператор ее изменения на следующем, (N+1)-м этапе. Задача агрегации графа заключается в определении оператора изменения агрегации.

Назовем *операцией первого рода* перенос одной (i-й) вершины из одного (k-го) сегмента в другой (l-й). Здесь сегментами названы вершины агрегированного графа Г′. Эту операцию, символически изображенную на рис. 6.2.1, будем обозначать тройкой

$$\psi = \langle kil \rangle, \tag{6.2.43}$$

причем $i \in I_k$ и $i \notin I_l$.

Эффект от такой операции легко вычислить. Очевидно, что в агрегированном графе Г′ произойдут следующие изменения:

$$\begin{aligned} a'_k &\to a'_k - a'_i; \\ a'_l &\to a'_l + a'_i, \end{aligned} \tag{6.2.44}$$

выражающие изменение сегментов, а также

$$\begin{aligned} b'_{kl} &\to b'_{kl} + \sum_{j \in I_k} b_{ij} - \sum_{j \in I_l} b_{ij}; \\ b'_{lk} &\to b'_{lk} + \sum_{j \in I_k} b_{ji} - \sum_{j \in I_l} b_{ji}, \end{aligned} \tag{6.2.45}$$

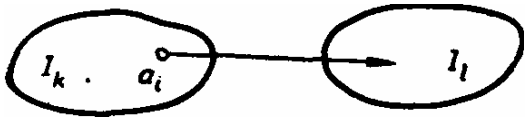


Рис. 6.2.1. Операция первого рода.

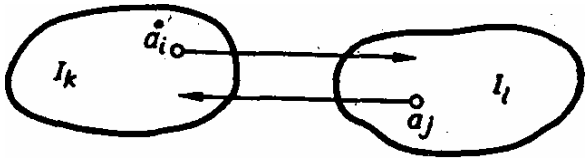


Рис. 6.2.2. Операция второго рода.

выражающие изменение связей между сегментами. Изменение критерия (6.2.11) имеет вид

$$\Delta Q = \sum_{j \in I_k} (b_{ij} + b_{ji}) - \sum_{j \in I_l} (b_{ij} + b_{ji}). \quad (6.2.46)$$

Знак этого приращения определяет, очевидно, эффективность операции (разумеется, при соблюдении ограничений (6.2.12)). *Операция второго рода* связана с обменом вершинами между двумя сегментами (рис. 6.2.2). Обозначим ее четверкой

$$\psi = \langle kilj \rangle, \quad (6.2.47)$$

где, очевидно, $i \in I_k$ и $j \in I_l$

При этом происходят следующие изменения в агрегированном графе Γ':

$$a'_k \to a'_k - a_i + a_j;$$
$$a'_l \to a'_l + a_i - a_j; \quad (6.2.48)$$

$$b'_{kl} \to b'_{kl} + \sum_{j \in I_k}' b_{ij} - \sum_{j \in I_l}' b_{ij} + \sum_{i \in I_l}' b_{ji} - \sum_{i \in I_k}' b_{ji};$$

$$b'_{lk} \to b'_{lk} + \sum_{j \in I_k} b_{ji} - \sum_{j \in I_l} b_{ji} + \sum_{i \in I_l} b_{ij} - \sum_{i \in I_k} b_{ji}.$$

Приращение критерия Q при этом равно

$$\Delta Q_{ij} = \sum_{j \in I_k} (b_{ij} + b_{ji}) - \sum_{j \in I_l} (b_{ij} + b_{ji}) +$$
$$+ \sum_{i \in I_l} (b_{ji} + b_{ij}) - \sum_{i \in I_k} (b_{ji} + b_{ij}). \quad (6.2.49)$$

Легко видеть, что результат операции второго рода может быть представлен как последовательность выполнения двух операций первого рода:

$$\langle kilj \rangle = \langle kil \rangle \langle ljk \rangle. \quad (6.2.50)$$

Произведение (6.2.50) коммутативно, так как результат не зависит от порядка проведения операций первого рода. Это хорошо видно из выражения (6.2.49), где первые два члена определяют результат операции ‹ *kil* ›, а вторые два — операции ‹ *ljk* ›.

Алгоритм агрегации должен указывать, какую операцию следует использовать в том или ином случае и каким образом определять вершины и сегменты, участвующие в этой операции. Будем исходить из какой-то (например, случайной) исходной сегментации графа.

6.2.3.3. Алгоритм устранения нарушения ограничений

Так как соблюдение ограничений S на объем сегментов (6.2.12) имеет больший приоритет, чем оптимизация значений критерия, то в этом случае критерием выбора операции и сегмента является только нарушение ограничений. Алгоритм устранения нарушений S состоит в следующем:

1. Определяется номер сегмента, в котором нарушается ог раничение:

$$k : a'_k > c_k. \qquad (6.2.51)$$

Если таких сегментов несколько, то все они, кроме одного (k-го), исключаются из рассмотрения до тех пор, пока не будет устранено нарушение (6.2.51).

2. Среди оставшихся сегментов, удовлетворяющих ограни чению, ищется сегмент с максимальным «запасом»:

$$\Delta a'_l = c_l - a'_l = \max_z (c_z - a'_z). \qquad (6.2.52)$$

3. В сегменте I_k определяется вершина, вес которой макси мально приближается снизу к запасу (6.2.52), т. е. решается задача

$$a_i = \max_{j \in I_k} (a_j | a_j < a'_l). \qquad (6.2.53)$$

Здесь возможны два варианта. Если такая вершина находится в I_k, то с ней производится операция первого рода (6.2.43), а далее — переход к п. 1. Если же такой вершины не найдется, то следует обращаться к операции второго рода.

4. Для этого из всех оставшихся сегментов случайно выби рается один (l-й) с вероятностью

$$p_l = \frac{\Delta a'_l}{\sum_{z \neq k} \Delta a'_z}. \qquad (6.2.54)$$

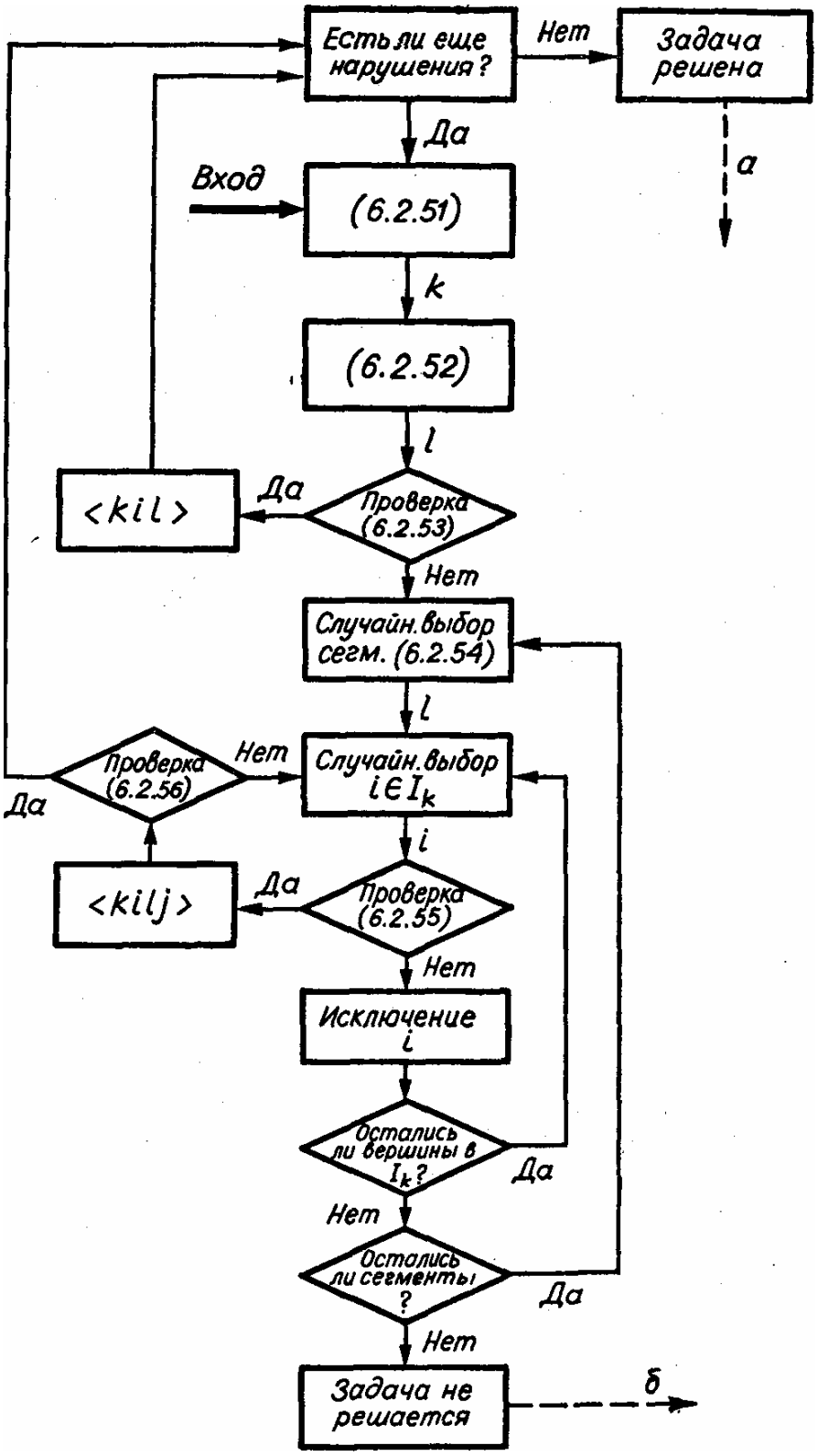


Рис. 6.2.3. Блок-схема алгоритма, устраняющего нарушение ограничения. Пунктирные стрелки *а* и *б* — для случая адаптивного разрезания (см. п. 6.2.3.6).

5. В сегменте I_k случайно выбирается одна вершина a_i, а в сегменте I_l ищется (j-я) вершина, обладающая свойством

$$a_i - \Delta a'_l \le a_j < a_i. \qquad (6.2.55)$$

Если такая j-я вершина в I_l найдется, то производится операция второго рода вида (6.2.47). При этом в случае

$$a_i - a_j \ge a'_k - c_k \qquad (6.2.56)$$

нарушение в I_k устраняется полностью, далее — по п. 1. В противном случае нарушение уменьшается и следует обратиться к п. 5. При этом новую вершину a_j целесообразно исключить из случайного «розыгрыша».

Если же в I_l не найдется вершины, удовлетворяющей условию (6.2.55), то следует обратиться к п. 4, исключив из рассмотрения a_i.

Указанное исключение вершин в конце концов исчерпывает сегмент I_l, что заставляет обращаться к п. 4, т. е. к другому сегменту, исключая l-й из дальнейшей «игры».

Блок-схема этого алгоритма показана на рис. 6.2.3.

6.2.3.4. Локальная минимизация критерия при неизменной среде [115]

Пусть ограничения S выполнены (априори или с помощью описанного выше алгоритма). Тогда задача минимизации критерия (6.2.11) X=const решается следующим алгоритмом.

1. Случайным образом выбирается пара сегментов (k-й и l-й) с вероятностью

$$p_{kl} = \frac{b'_{kl} + b'_{lk}}{\sum_{i \ne j = 1}^{m} (b'_{ij} + b'_{ji})} \qquad (6.2.57)$$

(тем самым стохастически выбирается максимально связанная пара сегментов).

2. Для каждого из сегментов пары определяется «запас»:

$$\Delta a'_l = c_l - a'_l; \qquad (6.2.58)$$
$$\Delta a'_k = c_k - a'_k$$

и выбирается сегмент с наименьшим запасом (пусть это k-й сегмент, т. е. $\Delta a'_k < \Delta a'_l$).

3. Строится множество $\{Z\}$ номеров элементов k-го сегмента, удовлетворяющих условию

$$a_z \le \Delta a'_l \quad (z \in I_k), \qquad (6.2.59)$$

т. е. потенциальных «претендентов» на перевод в l-й сегмент. Если это множество не пусто, то для каждого его элемента определяется прирост критерия при переводе этого элемента в I_l:

$$\Delta Q_z^l = -\sum_{j\in I_l}(b_{zj}+b_{jz}) + \sum_{j\in I_k}(b_{zj}+b_{jz}) \quad (6.2.60)$$

Если наименьшее приращение отрицательно, то оно и определяет номер элемента

$$i = \arg\min_z (\Delta Q_z^l \mid \Delta Q_z^l < 0), \quad (6.2.61)$$

с которым следует произвести операцию первого рода ‹*kil*›, т. е. перенести его из I_k в I_l.

Если же такого элемента не найдется, т. е. не удовлетворяются условия (6.2.59) или $\Delta Q_z > 0$, то следует образовать множество $\{Z\} \in I_l$, удовлетворяющее условию

$$a_z \leqslant \Delta a'_k \qquad (z \in I_l), \quad (6.2.62)$$

из которого выбрать элемент с номером

$$j = \arg\min_z (\Delta Q_z^k \mid \Delta Q_z^k < 0), \quad (6.2.63)$$

где

$$\Delta Q_z^k = -\sum_{i\in I_k}(b_{zi}+b_{iz}) + \sum_{i\in I_l}(b_{zi}+b_{iz}), \quad (6.2.64)$$

и произвести операцию первого рода ‹*ljk*›.

Если же элемента, обладающего свойствами (6.2.62) и $\Delta Q_z^k < 0$, не найдется, то следует обратиться к операции второго рода.

4. Для этого организуется случайный перебор всех возможных обменов между k-м и l-м сегментами. Пусть i — номер случайно выбранного элемента из I_k, а j — номер случайно выбранного элемента из I_l. Тогда при одновременном выполнении условий

$$\begin{aligned} &a_i \le \Delta a'_l + a_j; \\ &a_j \le \Delta a'_k + a_i; \\ &\Delta Q_{ij} < 0 \end{aligned} \quad (6.2.65)$$

(выведенных с использованием выражения (6.2.49)) производится операция второго рода ‹*kilj*›, после чего снова стохастически выбирается пара наиболее связанных сегментов (см. п. 1). Естественно, что в процессе такого случайного перебора следует отбрасывать уже проверенные варианты пар элементов,

не удовлетворяющие (6.2.65). После завершения случайного перебора следует обратиться к случайному выбору наиболее связанной, но другой случайной пары сегментов (см. п. 1).

Если ни одна из $m(m—1)/2$ пар сегментов, проверяемых подряд, не дала улучшения критерия, то задачу локальной оптимизации можно считать решенной (точнее, локально не улучшаемой).

Так как задача об оптимальном разрезании графа, вообще говоря, имеет многоэкстремальный характер, то найденное оптимальное решение локально и зависит от первоначального разрезания. Поэтому, варьируя исходные агрегации, можно улучшить первоначальное локальное решение.

Описанный алгоритм представлен в виде блок-схемы на рис. 6.2.4. Его сходимость исследована в работе [271].

Детерминированный аналог этого алгоритма получается в случае, когда оператор *1* случайного выбора заменяется на оператор выбора пары сегментов с максимальной связностью, т. е.

(6.2.66)

$$k, l: b'_{kl}+b'_{lk}=\max_{i,j}(b'_{ij}+b'_{ji}),$$

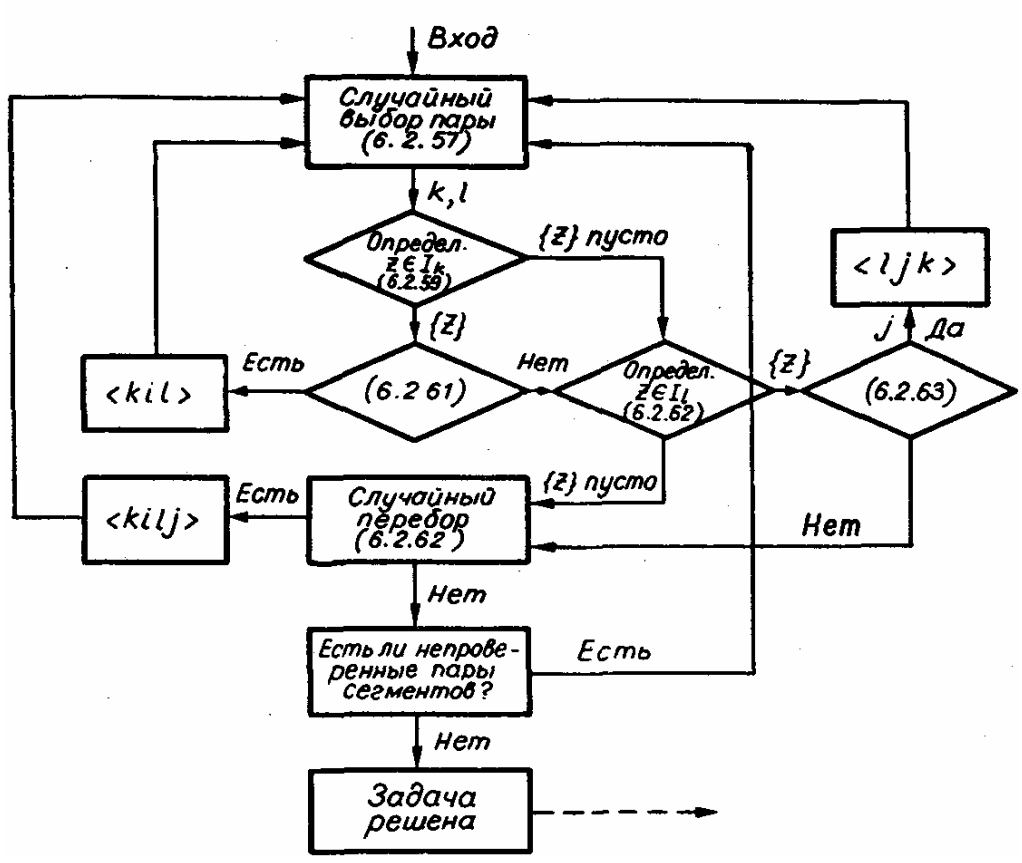


Рис. 6.2.4. Блок-схема алгоритма поиска оптимального разрезания. Пунктирная стрелка — для случая адаптивного разрезания (см. п. 6.2.3.6).

а оператор *2* случайного перебора — на выбор пары вершин, обеспечивающих наиболее интенсивное уменьшение критерия:

$$i, j: \Delta Q_{ij} = \min_{z \in I_k, w \in I_l} (\Delta Q_{zw} | \Delta Q_{zw} < 0), \quad (6.2.67)$$

где ΔQ_{zw} определено выше (6.2.49).

Такой детерминированный алгоритм рассмотрен в работе [224]. Преимущество случайного перебора перед полным (6.2.67) в данном случае очевидно, так как условия (6.2.65) при случайном переборе выполняются в среднем раньше, чем закончится полный перебор (6.2.67).

Приведем примеры работы этого алгоритма [115].

Пример 1. Рассмотрим граф с восемью вершинами (n=8) из работы [27] (рис. 6.2.5). Матрица весов этого графа имеет вид

$$B = \begin{Vmatrix} 0 & 1 & 2 & 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Vmatrix} \quad (6.2.68)$$

Веса вершин графа одинаковы и равны единице (a_i=l). Требуется разрезать граф на два подграфа (m = 2) по четыре вершины в каждом (c_j = 4).

Начальное разрезание выбирается случайно, например:

$$U_0 = \begin{cases} I_1 = \{1, 2, 3, 4\}; \\ I_2 = \{5, 6, 7, 8\}, \end{cases} \quad (6.2.69)$$

и т. д.

Полученное с помощью описанного алгоритма оптимальное разрезание имеет вид

$$U^* = \begin{cases} I_1 = \{1, 3, 4, 6\}; \\ I_2 = \{2, 5, 7, 8\} \end{cases} \quad (6.2.70)$$

и совпадает с полученным в работе [27], где эта задача решена методом ветвей и границ (см. рис. 6.2.5).

Пример 2. В качестве другого примера рассмотрим задачу определения оптимального сечения для устройства сопряжения двух ЭВМ одного типа с несимметричным стыком [112]. Эта задача сводится к задаче разрезания графа. Устройство

сопряжения представлено в виде неориентированного графа со взвешенными ребрами (рис. 6.2.6). Вершина *19* — первая ЭВМ, вершина *20* — вторая ЭВМ. Число вершин графа n=22. Веса вершин одинаковы и равны единице. Матрица весов B графа имеет следующий вид:

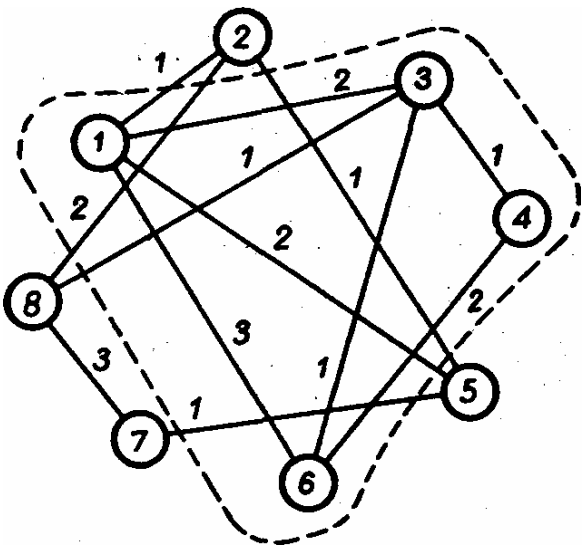
$$B=\begin{Vmatrix}
0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&3&0&0&0\\
&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
&&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
&&&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
&&&&0&1&0&0&0&0&0&0&0&0&0&0&0&2&0&0&0\\
&&&&&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
&&&&&&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
&&&&&&&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
&&&&&&&&0&0&0&1&0&0&0&0&0&2&0&0&0\\
&&&&&&&&&0&0&0&0&0&0&0&0&0&3&0&0\\
&&&&&&&&&&0&1&0&0&0&0&0&0&2&0&0\\
&&&&&&&&&&&0&0&0&0&0&0&0&0&1&0\\
&&&&&&&&&&&&0&1&0&0&0&0&0&0&0\\
&&&&&&&&&&&&&0&1&0&0&0&0&0&0\\
&&&&&&&&&&&&&&0&1&0&0&0&0&0\\
&&&&&&&&0&&&&&&&0&0&0&0&1&0&0\\
&&&&&&&&&&&&&&&0&1&0&1&0&0\\
&&&&&&&&&&&&&&&&0&0&1&0&0\\
&&&&&&&&&&&&&&&&&0&0&0&0\\
&&&&&&&&&&&&&&&&&&0&0&1\\
&&&&&&&&&&&&&&&&&&&0&1\\
&&&&&&&&&&&&&&&&&&&&0
\end{Vmatrix}$$

Требуется разрезать граф на два подграфа (m=2) так, чтобы сумма весов ребер между подграфами была минимальна, а число вершин в каждом подграфе не превышало c=21. Кроме того, вершины *19* и *20* должны находиться в разных подграфах. Начальное разрезание выбирается случайным (рис. 6.2.6):

$$U_0=\begin{cases} I_1=\{4,5,7,8,9,19\};\\ I_2=\{1,2,3,6,10,11,12,13,14,15,16,17,18,20,21,22\};\end{cases}$$

$$Q(U_0)=7. \qquad (6.2.71)$$

Условие «неподвижности» вершин *19* и *20* учитывается тем, что они не входят в число вершин, претендующих на перенос.



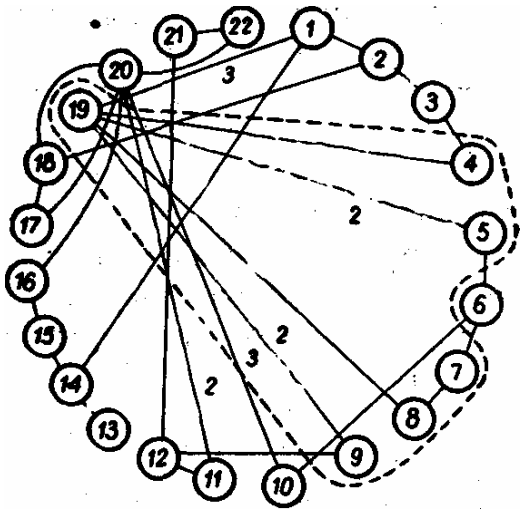


Рис. 6.2.5. Исходный граф примера 1. Пунктиром показано оптимальное разрезание.

Рис. 6.2.6. Граф устройства сопряжения двух ЭВМ. Единичные веса связей не обозначены. Пунктиром показано исходное разрезание. Оптимальное разрезание получается включением в «пунктирную» зону вершин *1, 2, 3* и *6*.

Первая структура, полученная предложенным алгоритмом, имеет вид

$$U^*_1 = \begin{cases} I_1 = \{1, 4, 5, 6, 7, 8, 9, 19\}; \\ I_2 = \{2, 3, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22\}; \end{cases} \quad (6.2.72)$$

$$Q(U^*_1) = 5.$$

Содержательный анализ этой задачи показал, что критерий качества может иметь «площадки». Для их преодоления в выражения (6.2.61), (6.2.63) и (6.2.65) алгоритма были добавлены равенства, что обеспечило возможность движения вдоль площадок минимизируемой функции $Q(U)$. Это дало возможность получить лучшее решение, чем предыдущее:

$$U^*_2 = \begin{cases} I_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 19\}; \\ I_2 = \{10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22\}; \end{cases} \quad (6.2.73)$$

$Q(U^*_2) = 4.$ Таким образом, предложенный алгоритм эффективно решает задачу об оптимальной агрегации (разрезании) графа.

Отметим, что описанный алгоритм является локальным в силу релаксации на каждом шаге, т. е. неухудшения критерия $Q(U)$. Это обеспечивает надежность отыскания локального минимума минимизируемого критерия. Однако задача об оптимальном разрезании достаточно сложного графа имеет ярко

выраженный многоэкстремальный характер. Существующие точные методы ее решения типа «ветвей и границ» требуют очень больших затрат машинного времени, а эвристические детерминированные методы позволяют находить лишь локальное решение данной задачи. Поэтому необходима разработка таких методов, которые совмещают в себе быстродействие эвристических процедур с возможностью нахождения глобального экстремума. Это обстоятельство заставляет рассмотреть глобальные алгоритмы случайного поиска для решения задачи агрегации графа.

6.2.3.5. Алгоритм глобальной минимизации критерия оптимальности $Q(U)$ [114]

Будем «глобализировать» алгоритм, описанный в подразделе 6.2.3.4. Это можно сделать двумя способами. Прежде всего, применим случайный выбор исходных разрезаний:

$$U_1^0, \ldots, U_N^0 \in S, \quad (6.2.74)$$

удовлетворяющих ограничениям 5 на размер сегментов. Локальный спуск с помощью описанного выше алгоритма из каждой начальной точки (6.2.74) дает локальные оптимумы

$$U^*_1, \ldots, U^*_N. \quad (6.2.75)$$

Естественно глобальный оптимум оценивать лучшим из локальных:

$$U^{**} = \arg\min_{i=1,\ldots,N} Q(U^*_i).$$

Другой мерой, «глобализирующей» алгоритм, является введение возможности ухудшения критерия. Для этого достаточно формулы (6.2.61), (6.2.63) и (6.2.65) записать соответственно в виде

$$\begin{aligned} i &= \operatorname*{argmin}_z (\Delta Q_z^l \mid \Delta Q_z^l < \alpha); \\ j &= \operatorname*{argmin}_z (\Delta Q_z^k \mid \Delta Q_z^k < \alpha); \\ \Delta Q_{ij} &< \alpha, \end{aligned} \quad (6.2.76)$$

где в ограничениях вместо нуля введен параметр α. При α>0 возможно увеличение минимизируемого критерия и образуются предпосылки «перевала» в зону глобального экстремума. Очевидно, что на эту меру следует идти лишь попав в локальный экстремум.

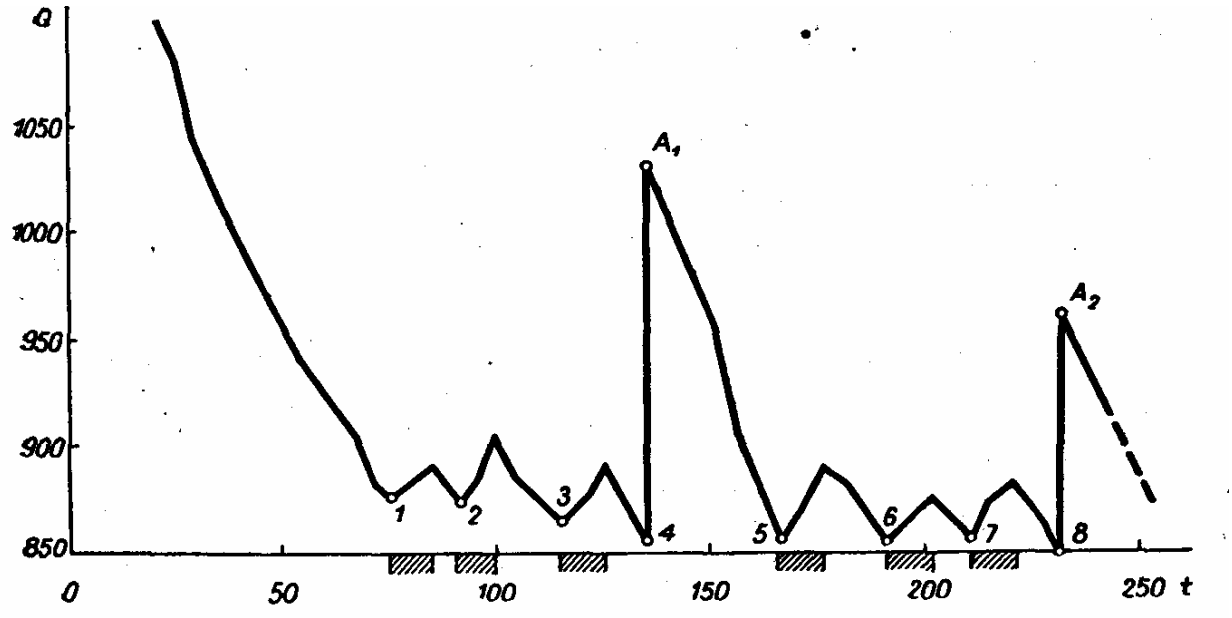


Рис. *6.2.7.* Поведение связности графа в процессе адаптации. Штриховкой отмечены интервалы времени, когда $\alpha > 0$.

Алгоритм глобального разрезания в этом случае представляется таким образом. При $\alpha = 0$ достигается локальный экстремум U^*_1, который запоминается вместе с $Q^*_1 = Q(U^*_1)$. Далее параметр а на некоторое число шагов увеличивается и затем снова $\alpha = 0$. Полученное локальное значение критерия $Q^*_2 = Q(U^*_2)$ сравнивается с Q^*_1. Если $Q^*_2 < Q^*_1$, то вместо Q^*_1 и U^*_1 запоминаются как лучшее решение Q^*_2 и U^*_2, в противном случае поиск продолжается, и т. д. Легко заметить, что в результате работы алгоритма шансы найти глобальный экстремум со временем увеличиваются.

Приведем пример работы этого алгоритма [114]. Рассматривается граф с $n = 60$, который необходимо разрезать на семь подграфов, т. е. $m = 7$ при ограничениях с = 9. Веса всех вершин графа одинаковы и равны единице ($a_i = 1$). Значения функции качества в локальных минимумах U^*_k были следующими: *i* 1 2 3 4 5 6 7 8 9 10 11 12

$Q(U^*_1)$ 875 874 863 853 855 853 857 848 868 871 870 877

Начальное значение функции качества $Q(U_0) = 1242$. В качестве глобального решения получено разрезание, при котором функция качества имеет значение $Q(U^*_8) = 848$.

На рис. 6.2.7 показана динамика поведения критерия при поиске. В моменты A_1 и A_2 вводились рандомизированные изменения начальных условий, что вызвало резкое увеличение критерия.

Таким образом, разумное оперирование обоими средствами глобализации позволяет эффективно и без лишних затрат оценивать глобальный экстремум в задаче о разрезании.

6.2.3.6. Адаптивная агрегация графа с переменными свойствами в детерминированной среде

Рассмотрим случай, когда параметры (6.2.4) агрегируемого графа Г изменяются во времени или под действием среды:

$$a_i = a_i\,(X, t); \qquad (6.2.77)$$
$$b_{ij} = b_{ij}\,(X, t),$$

причем эти зависимости детерминированны и среда изменяется достаточно медленно, т. е. производная $X(t)$ мала по модулю.

Тогда оптимальная агрегация (разрезание) U^* (6.2.16) также изменяется и возникает задача ее отслеживания, т. е. адаптации к изменяющимся параметрам графа. Это обычная задача адаптации к новым условиям, причем ограничения S могут нарушаться. Алгоритм адаптивной агрегации должен иметь возможность как устранять возникающие нарушения ограничений S, так и минимизировать критерий.

Воспользуемся описанным выше алгоритмом устранения нарушения ограничений S (см. п. 6.2.3.3) и алгоритмом поиска оптимального разрезания (см. п. 6.2.3.4). Связь этих алгоритмов показана на рис. 6.2.3 и 6.2.4 пунктирными стрелками. В алгоритме устранения нарушения ограничений S (см. рис. 6.2.3) таких стрелок две: *а* и *б*. Стрелка а обозначает переход при устранении ограничений, а стрелка *б* — при невозможности это сделать. В последнем случае для решения задачи необходимо расширение ресурсов C (6.2.13).

Блок-схема адаптации, объединяющая оба рассмотренных выше алгоритма, приведена на рис. 6.2.8. Здесь стрелки а и б соответствуют пунктирным стрелкам а и б на рис. 6.2.3.

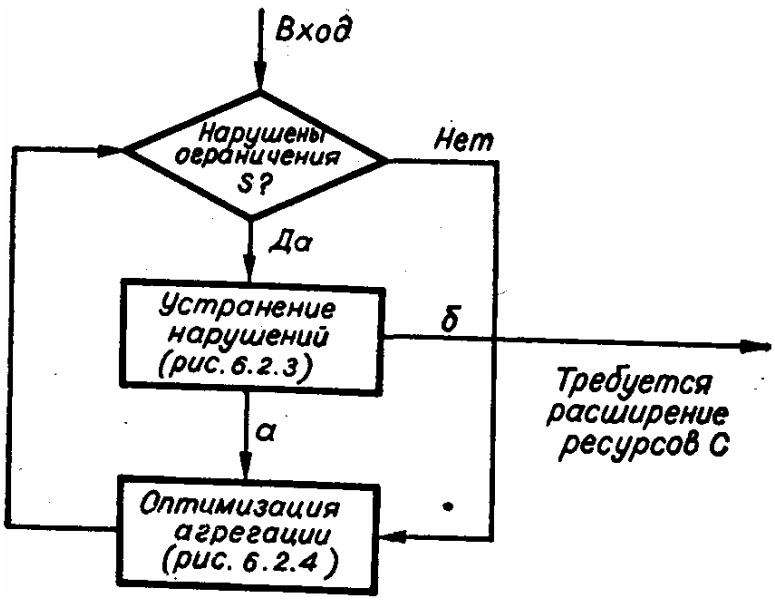


Рис. 6.2.8. Блок-схема адаптивной агрегации изменяющегося графа.

Как видно, алгоритм адаптации работает следующим образом. После устранения нарушения ограничений (или проверки их выполнения) производится локальная оптимизация агрегации графа, за которой снова следует проверка выполнения ограничений, и т. д. Эффективность работы такого алгоритма гарантируется медленностью изменения параметров (6.2.77) графа. Поэтому оптимальную агрегацию $U^*(X,t)$ удается отследить описанным образом.

Однако часто среда X изменяется стохастически в соответствии со своими неизвестными, но определенными статистическими свойствами, которые удобно описывать плотностью распределения состояний среды. Рассмотрим этот случай.

6.2.3.7. Стохастическая задача агрегации графа в соответствии с решением (6.2.18)

Рассмотрим для простоты следующую задачу:

$$\bar{Q}(U) \to \min_{U \in S}, \quad (6.2.78)$$

отличающуюся от (6.2.18) детерминированным характером ограничений S, имеющих вид (6.2.15). В этом случае можно считать, что среда влияет лишь на матрицу связей графа, т. е.

A = const;

$$B = B(X). \quad (6.2.79)$$

Будем предполагать, что состояния среды $X_1,...,X_N$ являются случайными независимыми реализациями неизвестного, но определенного случайного процесса, имеющего плотность распределения $p(X)$, которая, вообще говоря, может изменяться во времени. Случайные состояния $\{X_k\}$ среды определяют, в соответствии с (6.2.79), случайные графы $\{\Gamma_k\} = \{A, B(X_k)\}$, которые и являются исходным материалом для агрегации.

Необходимо решить (6.2.78), располагая только наблюдениями X_1, ..., X_N. Это обычная адаптация, которая состоит в том, что на каждом k-м этапе решается задача

$$Q(U, X_k) \to \min_{U \in S} \Rightarrow U^*_k. \quad (2.6.80)$$

Процедура решения должна быть организована так, чтобы последовательность $U^*_1,...,U^*_N$ сходилась к оптимальной агрегации U^*. Рассмотрим две модели образования случайных графов Γ_k (k=1,...,N). Пусть для простоты

$$B = X \quad (6.2.81)$$

(более общий случай выводится аналогично). Здесь X — случайная матрица $n \times n$, характеризующая конкретное состояние среды

$$X = \| x_{ij} \|_{n \times n} \qquad (6.2.82)$$

Бернуллиевская модель предполагает независимость весов b_{ij} графа. Матрица X в этом случае имеет независимые случайные элементы, стохастические свойства которых $p_{ij}(x_{ij})$ должны быть заданы при моделировании. Генерирование случайных графов; Γ_k здесь не вызывает трудностей.

Марковская модель графа Γ_k опирается на интерпретацию случайной матрицы X как конечной реализации дискретного марковского процесса с конечным числом (n) состояний. В этом случае элемент x_{ij} матрицы X определяется как число обращений к переходу от i-го состояния к j-му за время реализации.

Таким образом, задавая стохастическую матрицу

$$P = \| p_{ij} \|_{n \times n} \qquad (6.2.83)$$

марковской цепи, можно легко генерировать случайные матрицы X, характеризующие случайные реализации графа Г. Легко заметить, что матрица X является смещенной оценкой стохастической матрицы P. Действительно,

$$\hat{p}_{ij} = \frac{x_{ij}}{e_i}, \qquad (6.2.84)$$

где

$$e_i = \sum_{j=1}^{n} x_{ij} \qquad (6.2.85)$$

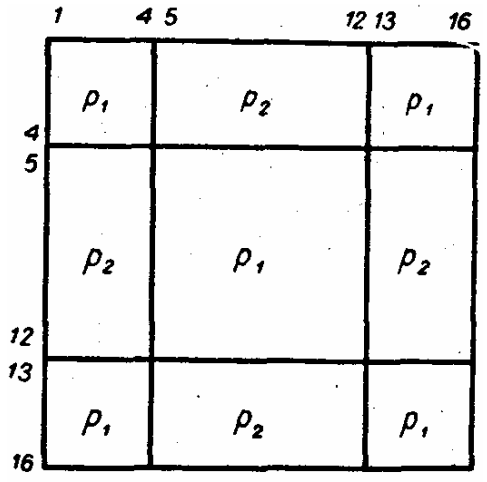


Рис. 6.2.9. Структура стохастической матрицы взаимосвязей вершин графа в экспериментах (примеры 1—3 п. 6.2.3.7).

Такая модель позволяет удобно анализировать процесс получения оптимального разрезания графа адаптивным методом, опирающимся на наблюдения X_k (k = 1, ..., N). Применяя в данном случае на каждом этапе алгоритм, рассмотренный в предыдущем подразделе (см. рис. 6.2.8), можно быть уверенным, что этот процесс сойдется к оптимальному разрезанию, решающему задачу (6.2.78).

Проиллюстрируем сказанное на нескольких экспериментальных примерах.

Пусть для простоты веса вершин одинаковы и равны единице, т. е. $a_i = 1$. Граф имеет $n = 16$ вершин, связанных между собой в соответствии со стохастической матрицей P, состоящей из двух элементов p_1 и p_2, значительно отличающихся друг от друга (p1>>p2). На рис. 6.2.9 представлена структура этой матрицы.

Оптимальная агрегация U^* априори очевидна:

$$U^*: \begin{cases} I_1 = \{1, 2, 3, 4, 13, 14, 15, 16\}; \\ I_2 = \{5, 6, 7, 8, 9, 10, 11, 12\}. \end{cases} \quad (6.2.86)$$

Для оценки эффективности получаемой агрегации U^* естественно ввести коэффициент близости к оптимальной агрегации на N-м. этапе адаптации:

$$k_N = \frac{n_N}{n}, \quad (6.2.87)$$

где n_N — число вершин графа, которое на N-м шаге совпадает с оптимальной агрегацией U^*.

В экспериментах ресурсы C не ограничивались и объем T составлял 128, т. е. матрица (6.2.82) строилась на базе 128 элементов марковской цепи.

Пример 1. $p_1 = 1 / 8$; $p_2 = 0$. Исходная агрегация имела три сегмента:

$$U_0: \begin{cases} I_1 = (1, 2, 3, 4, 5); \\ I_2 = (6, 7, 8, 9, 10); \\ I_3 = (11, 12, 13, 14, 15, 16), \end{cases} \quad (6.2.88)$$

что соответствовало $k_0 = 0{,}56$.

Результаты работы алгоритма адаптации приведены в табл. 6.2.1, из которой видно, что оптимальная агрегация получена на седьмом ($N = 7$) этапе адаптации и далее не изменяется. При этом исходная неоптимальная трехсегментная агрегация ($m_0 = 3$) автоматически перестроилась в оптимальную двухсегментную ($m_7 = 2$).

Пример 2. $p_1 = 7 / 64$; $p_2 = 1 / 64$. Начальная агрегация, как и в предыдущем примере, была трехсегментной (6.2.87).

Результаты работы алгоритма адаптации (табл. 6.2.2) интересны тем, что оптимальная агрегация была получена уже на третьем этапе $U_3 = U^*$, а на пятом произошло ухудшение агрегации. Дело в том, что случайные реализации графа B могут достаточно сильно отклоняться от среднего значения и тем самым вносить существенные искажения в результаты, полученные -ранее. Чтобы избежать этого, можно воспользоваться стан-

Таблица 6.2.1

Этап адаптации N	Агрегация U_N на N-м этапе адаптации	Коэф. близости к оптим. агрегации k_N	Критерий качества Q_N
0	Согласно (6.2.88)	0,56	72
1	I_1= (1, 2, 3, 4, 5) I_2= (7, 8, 9, 10, 11, 12) I_3= (6, 13, 14, 15, 16)	0.6	61
2	I_1= (1, 2, 3, 4) I_2= (5, 7, 8, 9, 10, 11, 12) I_3= (6, 13, 14, 15, 16)	0.7	32
3	I_1= (1, 2, 3, 4) I_2= (5, 6, 7, 8, 9, 10, 11, 12) I_3= (13, 14, 15, 16)	0.75	0
4	То же, что при N=3	0.75	0
5	То же, что при N=4	0.75	0
8	I_1= (1, 2, 4, 13, 14, 15, 16) I_2= (5, 6, 7, 8, 9, 10, 11, 12) I_3=(3)	0.9	37
7	I_1= (1, 2, 3, 4, 13, 14, 15, 16) I_2= (5, 6,7, 8, 9, 10, 11, 12)	1	0
8	То же, что при N=7	1	0
9	То же, что при N=8	1	0
10	То же, что при N=9	1	0

Таблица 6.2.2*

N	U_N	k_N	Q_N
0	Согласно (6.2.88)	0,56	75
1	I_1 = (1, 2, 3, 4, 16) I_2 = (9, 7, 10) I_3 =(5, 6, 8, 11, 12, 13, 14)	0.6	52
2	I_1= (1, 2, 3, 4, 13, 14, 16) I_2= (7) I_3= (5, 6, 8, 9, 10, 11, 12, 15)	0.9	33
3	I_1= (1, 2, 3, 4, 13, 14, 15, 16) I_2= (5, 6, 7, 8, 9, 10, 11, 12)	1	12
4	То же, что при N=3	1	11
5	I_1= (1, 2, 4, 13, 14, 15, 16) I_2= (3, 5, 6, 7, 8, 9, 10, 11, 12)	0.9	22
6	I_1= (1, 2, 3, 4, 13, 14, 15, 16) I_2= (5, 6, 7, 8, 9, 10, 11, 12)	1	15
7	То же, что при N=6	1	18
8	То же, что при N=7	1	15
9	То же, что при N=8	1	17
10	То же, что при N=9	1	20

* Обозначения те же, что в табл. 6.2.1.

*Таблица 6.2.3**

N	U_N	k_N	Q_N
0	Согласно (6.2.88)	0,56	78
1	$I_1 = (1, 2, 3, 4)$ $I_2 = (5, 6, 7, 8, 9, 10, 11, 12)$ $I_3 = (13, 14, 15, 16)$	0.75	56
2	$I_1 = (2, 3, 8)$ $I_2 = (1, 4, 5, 6, 7, 9, 10, 11, 12)$ $I_3 = (13, 14, 15, 16)$	0.75	45
3	$I_1 = (3, 8)$ $I_2 = (4, 5, 6, 7, 9, 10, 11, 12, 14)$ $I_3 = (1, 2, 13, 15, 16)$	0.75	35
4	$I_1 = (2)$ $I_2 = (4, 5, 6, 7, 8, 10, 11, 12, 13)$ $I_3 = (1, 3, 9, 14, 15, 16)$	0.75	50
5	$I_1 = (1, 3, 9, 13, 14, 15, 16)$ $I_2 = (2, 4, 5, 6, 7, 8, 10, 11, 12)$	0.8	45
6	$I_1 = (1, 2, 3, 4, 10, 13, 16)$ $I_2 = (5, 6, 7, 8, 9, 11, 12, 14, 15)$	0.8	37
7	$I_1 = (1, 2, 3, 4, 11, 13, 15, 16)$ $I_2 = (5, 6, 7, 8, 9, 10, 12, 14)$	0.9	36
8	$I_1 = (1, 2, 3, 4, 13, 14, 15, 16)$ $I_2 = (5, 6, 7, 8, 9, 10, 11, 12)$	1	39
9	То же, что при N=8	1	40
10	То же, что при N=9	1	37

* Обозначения те же, что в табл. 6.2.1 и 6.2.2.

дартным приемом, используемым в общей теории адаптации — не «обрабатывать до конца» полученную информацию. Например, можно остановить алгоритм адаптации не по критерию исчерпывания случайного перебора, а раньше — по критерию заданного числа шагов.

Пример 3. $p1 = 6 / 64$; $p_2 = 2 / 64$. Начальные условия (6.2.88) сохраняются. Результаты работы алгоритма адаптации приведены в табл. 6.2.3. Любопытно, что критерий качества агрегации изменялся незначительно ввиду большой связности оптимальной сегментации, которая, тем не менее, была найдена за восемь этапов адаптации.

Таким образом, описанные эксперименты убедительно доказывают эффективность предложенного алгоритма адаптивной агрегации при работе со стохастическими графами.

Теперь рассмотрим случай эволюционирующего стохастического графа, когда плотность распределения состояний X среды изменяется во времени, т. е. имеет вид

$$p\,(X, t). \qquad (6.2.89)$$

6.2.3.8. Адаптивная агрегация графа, функционирующего в нестационарной среде

Здесь оптимальная агрегация U^* изменяется во времени, т. е. агрегированный граф эволюционирует вместе со средой:

$$U^* = U^*(t). \qquad (6.2.90)$$

Алгоритм адаптивной агрегации (см. рис. 6.2.8) должен «отслеживать» этот дрейф оптимальной агрегации.

Пример. Дрейф матрицы (n=16) переходных вероятностей моделировался путем изменения границ вероятностей p_1=7/64 и p_2=1/64 (рис. 6.2.10) таким образом, что каждый сегмент при оптимальном разрезании содержал всегда восемь смежных вершин (первая и шестнадцатая вершины смежные). На рис. 6.2.10 пунктиром показано очередное состояние матрицы, а стрелкой — направление дрейфа границ. Буквой i обозначен промежуток между i-й и (i+1)-й строкой (столбцом). Дрейф матрицы представляется следующей формулой:

$$\begin{aligned} &i \to i + l \quad \text{для } i=1,...,16-l; \\ &i \to i + l - 16 \quad \text{для } i>16-l. \end{aligned} \qquad (6.2.91)$$

Скорость этого дрейфа была выбрана равной l вершинам, меняющим принадлежность к сегменту за один этап работы локального алгоритма агрегации. Матрица реализации строилась на базе 128 шагов марковской цепи при равновероятно выбранной начальной вершине. Ресурсы C не ограничивались.

Результаты проведенных экспериментов (табл. 6.2.4) показывают, что оптимальная агрегация находилась на каждом этапе работы алгоритма. Здесь k_N' — исходное значение коэффициента соответствия с оптимальной агрегацией на N-м этапе, а k_N — значение этого коэффициента после агрегации (оно всегда было равно 1). Стрелками показаны переходы состояний графа.

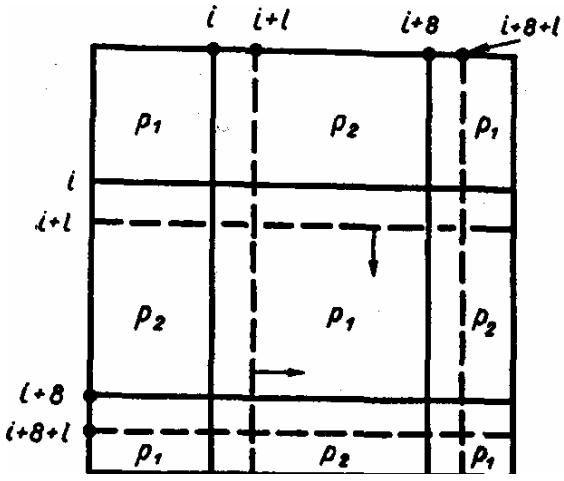


Рис. 6.2.10. Схема дрейфа матрицы переходов марковской цепи (пример л. 6.2.3.8).

На рис. 6.2.11 показано поведение минимизируемого критерия в процессе агрегации. Здесь j — моменты пересегментации (т. е. использования одной из операций ψ), N — моменты скачкообразного дрейфа матрицы на величину l после полной обработки алгоритма

Таблица 6.2.4

Этап адаптации N	l	Исходная агрегация	k'_N	Конечная агрегация	k_N
1	0	$I_1=(1, 2, 3, 4, 5)$	—	$I_1=(1, 2, 3, 4, 13, 14, 15, 16)$	1
		$I_2=(6, 7, 8, 9, 10)$		$I_2=(5, 6, 7, 8, 9, 10, 11, 12)$	
		$I_3=(11, 12, 13, 14, 15, 16)$			
2	1	$I_1=(1, 2, 3, 4, 13, 14, 15, 16)$	$\frac{7}{8}$	$I_1=(1, 2, 3, 4, 5, 14, 15, 16)$	1
		$I_2=(5, 6, 7, 8, 9, 10, 11, 12)$		$I_2=(6, 7, 8, 9, 10, 11, 12, 13)$	
3	1	$I_1=(1, 2, 3, 4, 5, 14, 15, 16)$	$\frac{7}{8}$	$I_1=(1, 2, 3, 4, 5, 6, 15, 16)$	1
		$I_2=(6, 7, 8, 9, 10, 11, 12, 13)$		$I_2=(7, 8, 9, 10, 11, 12, 13, 14)$	
4	2	$I_1=(1, 2, 3, 4, 5, 6, 15, 16)$	$\frac{6}{8}$	$I_1=(1, 2, 3, 4, 5, 6, 7, 8)$	1
		$I_2=(7, 8, 9, 11, 12, 13, 14)$		$I_2=(9, 10, 11, 12, 13, 14, 15, 16)$	
5	4	$I_1=(1, 2, 3, 4, 5, 6, 7, 8)$	$\frac{1}{2}$	$I_1=(1, 2, 3, 4, 13, 14, 15, 16)$	1
		$I_2=(9, 10, 11, 12, 13, 14, 15, 16)$		$I_2=(5, 6, 7, 8, 9, 10, 11, 12)$	

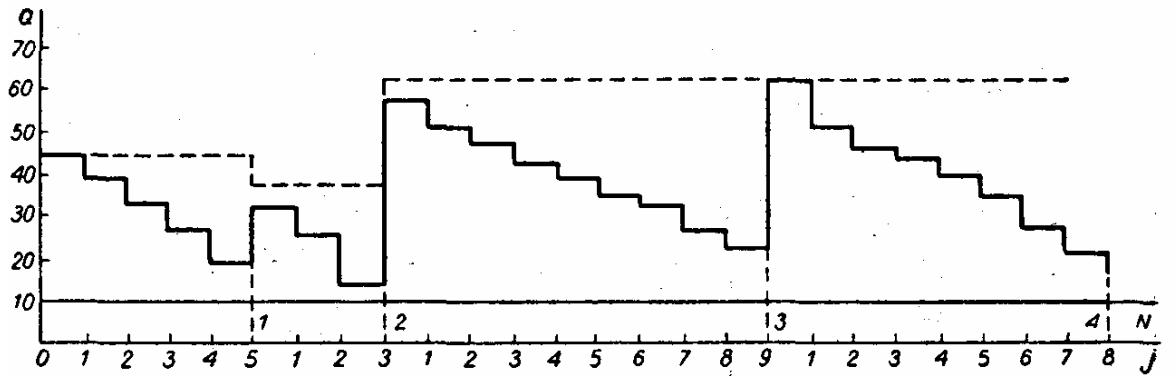


Рис. 6.2.11. Поведение критерия качества агрегации в процессе дрейфа при адаптивной агрегации.

агрегации. Пунктиром показано поведение критерия Q без адаптации.

Эксперименты наглядно показали, что описанный алгоритм эффективно отслеживает изменяющуюся агрегацию графа и поэтому может быть рекомендован для адаптации системы, функционирующей в изменяющейся среде, структура которой описывается графом.

6.2.4. Некоторые обобщения задачи об агрегации графа

Постановку задачи об агрегации графа, рассмотренную в подразделе 6.2.1, можно обобщить следующим образом. Введем функцию отношения между l-м и k-м сегментами $\varphi(l,k)$. Частный случай такой функции уже рассматривался в п. 6.2.2.2 и 6.2.2.4, где она выражала расстояние между l-м и k-м сегментами.

В общем случае функция $\varphi(l,k)$ выражает степень увеличения потерь от расположения вершин графа при агрегации в разных сегментах. Так, для стандартной задачи агрегации (см. подраздел 6.2.1) эта функция имеет вид характеристической:

$$\varphi(l,k) = \begin{cases} 1 & \text{при} \quad l \neq k; \\ 0 & \text{при} \quad l = k, \end{cases} \quad (6.2.92)$$

а для задачи оптимального расположения блоков информации на цилиндрах магнитного диска (см. п. 6.2.2.3) она принимает вид

$$\varphi(l,k) = |l - k|. \quad (6.2.93)$$

Критерия оптимальности агрегации графа теперь легко записать в форме

$$Q(U) = \sum_{i,j=1}^{n} b_{ij}\varphi(l_i, k_j), \quad (6.2.94)$$

где l_i — номер сегмента, содержащего i-ю вершину графа ($i \in I_l$), а k_j — номер сегмента, содержащего j-ю вершину ($j \in I_k$). Очевидно, что эта принадлежность определяется агрегацией U (6.2.10), т. е.

$$l_i = \zeta(U, j); \qquad (6.2.95)$$
$$k_j = \zeta(U, j),$$

где ζ — процедура определения принадлежности данной вершины одному из сегментов, или, точнее, — процедура определения номера сегмента, содержащего данную вершину при имеющейся агрегации U. Реализация такой процедуры, очевидно, не вызывает трудностей.

Задача, которую следует решать при оптимальной агрегации графа, теперь принимает вид

$$Q(U) = \sum_{i,j=1}^{n} b_{ij}\varphi[\zeta(U, i)\zeta(U, j)] \to \min_{U \in S}. \qquad (6.2.96)$$

Преимущество записи критерия в виде (6.2.94) заключается в том, что он определен явно на исходном графе Г, что упрощает организацию операций по перенесению вершин из одного сегмента в другой. Рассмотрим условия осуществления указанных операций преобразования агрегации.

Операция первого рода (см. п. 6.2.3.2) преобразует агрегацию согласно (6.2.42) и (6.2.43) таким образом, что критерий качества изменяется на величину

$$\Delta Q = Q(\langle kil \rangle U) - Q(U) =$$
$$= \sum_{j=1}^{n} (b_{ij} + b_{ji}) \{\varphi[\zeta(\langle kil \rangle U, i), \zeta(\langle kil \rangle U, j)] -$$
$$- \varphi[\zeta(U, i), \zeta(U, j)]\}. \qquad (6.2.97)$$

Легко видеть, что при (6.2.92) это выражение вырождается в (6:2.46). При функции φ вида (6.2.93) получаем

$$\Delta Q = \sum_{j=1}^{n} (b_{ij} + b_{ji}) \{|\zeta(\langle kil \rangle U, i) - \zeta(\langle kil \rangle U, j)| -$$
$$- |\zeta(U, i) - \zeta(U, j)|\}. \qquad (6.2.98)$$

Очевидно, что при $\Delta Q < 0$ операцию следует считать успешной, а в противном случае — неудачной.

Теперь рассмотрим операцию второго рода (6.2.47). Она изменяет критерий качества следующим образом (воспользуемся здесь коммутативностью (6.2.50)):

$$\Delta Q = Q(\langle kilj\rangle U) - Q(U) =$$
$$= \sum_{j=1}^{n} (b_{ij} - b_{ji}) \{\varphi[\zeta(\langle kil\rangle U, i), \zeta(\langle kil\rangle U, j)] -$$
$$- \varphi[\zeta(U, i), \zeta(U, j)]\} +$$
$$+ \sum_{i=1}^{n} (b_{ji} + b_{ij}) \{\varphi[\zeta(\langle ljk\rangle U, i), \zeta(\langle ljk\rangle U, j)] -$$
$$- \varphi[\zeta(U, i), \zeta(U, j)]\}. \qquad (6.2.99)$$

При φ, имеющей вид характеристической функции (6.2.92), это выражение совпадает с (6.2.49). Аналогично можно записать его для модульного представления (6.2.93). Учет полученных соотношений, определяющих успех или неуспех той или иной операции, в алгоритмах адаптивной агрегации п. 6.2.3.4 не представляет труда. Этим и определяется алгоритм поиска оптимальной агрегации графа с произвольным видом соотношения $\varphi(k, l)$ между сегментами.

Примеры адаптивной агрегации стохастического графа (n = = 16) с соотношением φ вида (6.2.93) между сегментами и марковской моделью состояния среды X приведены в работе [116].

§ 6.3. Адаптация процессов распределения памяти в вычислительной сети

6.3.1. Формулировка задачи

Рассмотрим вычислительную сеть, состоящую из n ЭВМ, каждая из которых имеет собственную память объемом c_i (i = 1, ..., n). Кроме того, в сети имеется k ЭВМ, играющих роль банков данных, в которых сосредоточена вся информация, необходимая для решения задач, поступающих в сеть. Пусть эта информация представляет собой множество блоков

$$\{a_j\} \quad (j = 1, ..., m), \qquad (6.3.1)$$

где a_j — объем (например, в битах) j-го блока информации.

Каждая ЭВМ может обратиться в любой банк данных за любым числом блоков из множества (6.3.1) и получить необходи-

мые данные, затратив на это определенное время, зависящее от длины очереди, объема требуемой информации и расстояния между этой ЭВМ и соответствующим банком или числа пунктов коммутации между ними.

Помимо этого, каждая ЭВМ в пределах своего объема памяти c_i может иметь некоторое количество информации, обращение к которой не требует дополнительного времени.

Заявки на обслуживание, поступающие в сеть, содержат перечень той информации из банков данных, которую необходимо использовать при решении каждой задачи-заявки. Будем характеризовать j-ю заявку множеством I_j номеров информационных блоков банков данных, используемых при обслуживании этой заявки в сети.

Дисциплину обслуживания в такой сети естественно связать со средним временем решения задач, что реализуется путем решения задачи на той ЭВМ, в памяти которой имеется необходимая для этой задачи информация.

Таким образом, цель состоит в том, чтобы определить два правила:

1) дисциплину обслуживания, т. е. правило направление оче редной заявки на одну из ЭВМ сети, руководствуясь при этом сведениями о требуемых блоках информации, состоянии памяти всех ЭВМ сети и их загрузке;

2) правило формирования памяти ЭВМ сети.

6.3.2. Сведение задачи распределения памяти к задаче математического программирования

Формализуем задачу. Пусть u_{ij} — двоичная переменная, выражающая наличие (u_{ij}=1) или отсутствие (u_{ij}=0) в памяти i-й ЭВМ j-го блока информации. Очевидно, что имеет место следующее ограничение, связанное с ограниченностью объема памяти каждой ЭВМ:

$$\sum_{j=1}^{m} a_j u_{ij} \leqslant c_i \qquad (i=1, \ldots, n). \quad (6.3.2)$$

Пусть поток задач, решаемых сетью, образуется из k различных потоков, каждый из которых характеризуется определенным распределением вероятностей использования блоков информации (6.3.1):

$$P_l = (p_{1l}, \ldots, p_{ml}) \quad (l = 1, \ldots, k), \ldots. \quad (6.3.3)$$

где p_{jl} — вероятность того, что при решении задачи l-го потока требуется j-й блок информации. Будем для удобства считать эти вероятности нормированными, т. е.

$$\sum_{j=1}^{m} p_{jl} = 1 \quad (6.3.4)$$

(хотя это и необязательно, так как для решения задачи обычно необходимо несколько блоков).

Определим вероятность попадания задачи l-го потока на i-ю ЭВМ при следующей дисциплине обслуживания: «направлять задачу туда, где больше необходимой информации». Эта вероятность

(6.3.5)

$$p'_{li}(U) = \frac{\sum_{j=1}^{m} u_{ij} p_{lj}}{\sum_{i,j} u_{ij} p_{lj}}.$$

Аналогично можно определить вероятность того, что задача-l-го потока, направленная на i-ю ЭВМ, найдет там необходимую для решения информацию, т. е. не придется обращаться к банку данных. Пусть эта вероятность

$$p''_{li} = p_{li}(U). \quad (6.3.6)$$

Теперь, располагая интенсивностями потоков решаемых задач (естественно их считать пуассоновскими)

$$\lambda_1, \ldots, \lambda_k \quad (6.3.7)$$

и интенсивностями решения задач каждой ЭВМ (их можно считать экспоненциальными)

$$\mu_1, \ldots, \mu_n, \quad (6.3.8)$$

можно сформулировать простейшую задачу определения оптимального расположения блоков информации (6.3.1) на ЭВМ сети, минимизирующего обращаемость к банку данных в процессе функционирования сети:

$$Q(U) = \sum_{i=1}^{n} \sum_{l=1}^{k} \lambda_l [1 - p''_{li}(U)] p'_{li}(U) \to \min_{U \in S} \Rightarrow U^*, \quad (6.3.9)$$

где

$$S:\begin{cases}\sum_{l=1}^{k}\lambda_l p_{li}(U)<\mu_i; & (6.3.10)\\ \sum_{i=1}^{m}a_i u_{ij}\leqslant c_i \quad (i=1,\ldots,n) & \text{Здесь } U \text{ — двоичный вектор, характеризующий распределение блоков памяти по сети:}\end{cases}$$

$$U=(u_{11},\ldots,u_{n1},\ldots,u_{nm});\ldots\ldots(6.3.11)$$

$Q(U)$ — интенсивность обращения к банку данных всех ЭВМ сети; первые ограничения (6.3.10) связаны с ограниченностью пропускной способности каждой ЭВМ: при нарушении хотя бы одного из них образуется бесконечная очередь заявок; вторые ограничения в (6.3.10) совпадают с (6.3.2).

Полученная задача (6.3.9) является задачей стохастического программирования с булевыми переменными (6.3.11) большой размерности nm. Для ее решения прежде всего необходимо иметь информацию о вероятной структуре (6.3.3) потоков решаемых задач, интенсивности (6.3.7) этих потоков и интенсивности (6.3.8) решения задач в сети. Но именно эту информацию труднее всего выявить. Более того — и поток задач, и интенсивности обслуживания могут изменяться во времени непредсказуемым образом [274].

Следовательно, задачу распределения памяти по сети следует решать адаптивным способом. Воспользуемся для этого методами эволюционной адаптации.

6.3.3. Адаптивное распределение памяти в сети ЭВМ

Естественной адаптивной мерой образования памяти ЭВМ в сети является сохранение наиболее часто используемых блоков информации. Сделать это можно следующим образом.

Пусть первоначальное распределение блоков по ЭВМ было произвольным (например, случайным):

$$I_{10},\ldots,I_{n0}, \qquad (6.3.12)$$

где I_{i0} — начальное множество номеров блоков информации, расположенное в памяти i-й ЭВМ. На это множество наложено ограничение по объему (6.3.2) в виде

$$\sum_{j\in I_{i0}} a_j\leqslant c_i \quad (i=1,\ldots,n). \qquad (6.3.13)$$

Заявка (задача) представляет собой множество V номеров блоков информации, требуемых для ее обслуживания (решения). Представим I и V *в* виде двоичных n-мерных векторов:

$$I_j = (i_{1j}, ..., i_{mj}) \quad (j = 1, ..., n); \tag{6.3.14}$$
$$V_l = (v_{1l}, ..., v_{ml}) \quad (l = 1, ...).$$

где

$i_{kj} = \begin{cases} 1, & \text{если } k\text{-й блок имеется в памяти } j\text{-й ЭВМ;} \\ 0 & \text{противном случае;} \end{cases}$. в

$v_{kl} = \begin{cases} 1, & \text{если } l\text{-й заявке необходим } k\text{-й блок информации; в} \\ 0 & \text{противном случае.} \end{cases}$

На двоичных векторах (6.3.14) введем функцию, определяющую число запрошенных l-й заявкой блоков информации, не обеспеченных содержимым памяти j-й ЭВМ:

$$\delta_{jl} = \sum_{k=1}^{m} v_{kl}\bar{i}_{kj}, \tag{6.3.15}$$

где верхней чертой обозначено логическое отрицание. Каждая l-я заявка сопровождается вектором

$$\Delta = (\delta_{1l}, ..., \delta_{ml}) \tag{6.3.16}$$

и направляется на обслуживание в ту (j-ю) ЭВМ, для которой δ_{il} минимально, т. е. требует минимальных затрат обращения к банку данных:

$$\delta_{jl} = \min_{i=1,...,m} \delta_{il}. \tag{6.3.17}$$

Если у j-й ЭВМ очередь превышает величину v_i, то заявка переправляется к другой машине, для которой δ_{il} минимально (исключая $i=j$). Там заявка принимается на обслуживание, если очередь к ней не превышает v_i. В противном случае заявка направляется к другой ЭВМ и т. д.

Таким образом, в дисциплину обслуживания входит, вектор предельных длин очередей

$$N = (v_1, ..., v_m), \tag{6.3.18}$$

который также может адаптироваться (но уже параметрическим образом).

Адаптация памяти каждой ЭВМ происходит после обслуживания очередной заявки. Здесь задача состоит в том, чтобы определить новое состояние памяти i-й ЭВМ $I_{i,N+1}$ по предыдущему

$I_{i,N}$ и множеству новых номеров Z_{N+1} блоков, полученных в данный момент из банка данных. Очевидно, что

$$I_{i,N} \cap Z_{N+1} = 0, \qquad (6.3.19)$$

т. е. эти множества не пересекаются. При этом $I_{i,N+1}$ должно удовлетворять ограничению

$$\sum_{j \in I_{i,N+1}} a_j \leqslant c_i \quad (i = 1, \ldots, n). \qquad (6.3.20)$$

Следовательно, работу алгоритма адаптивного изменения памяти можно представить в виде преобразования:

$$I_{i,N+1} = \varphi(I_{i,N}, Z_{N+1}), \qquad (6.3.21)$$

где φ — алгоритм адаптации.

В качестве такого алгоритма могут быть использованы известные алгоритмы замещения [3], предложенные для обмена между основной и вспомогательной памятью ЭВМ:

1. Алгоритм случайного замещения формирует из двух мно жеств $I_{i,N}$ и Z_{N+1} одно случайное, удовлетворяющее условию (6.3.20), таким образом, чтобы не осталось ни одного блока, ко торый можно было бы разместить в памяти, не нарушив условия (6.3.20).

2. Алгоритм «первый пришел — первый обслужен», в соответ ствии с которым из памяти удаляются (замещаются) те блоки, которые дольше находились в ней.

3. Алгоритм замещения наименее используемых блоков. В этом случае из памяти удаляются блоки, которые дольше всех не запрашивались при работе данной ЭВМ.

4. Многоуровневые алгоритмы замещения [3], которые сво дятся к организации на каждом шаге приоритетных списков и принятию решения на их базе. Эти алгоритмы обобщают преды дущие.

Хотя перечисленные алгоритмы вполне работоспособны, они не используют идей адаптации.

Предложим новый рандомизированный алгоритм φ. Пусть каждому j-му блоку информации, находящемуся в памяти любой ЭВМ сети, соответствует число $h_{j,N}$, выражающее уровень приоритета этого j-го блока в момент N (индекс i номера ЭВМ здесь можно снять, так как алгоритм одинаков для всех ЭВМ сети).

Введем монотонно возрастающую функцию $\psi(h)$, такую, что

$$0 \leq \psi(h_{j,N}) \leq 1 \quad (j = 1,..., m'), \qquad (6.3.22)$$

где m' — число блоков, претендующих на сохранение в памяти ЭВМ. Значение этой функции можно рассматривать как вероятность сохранения соответствующего блока в памяти.

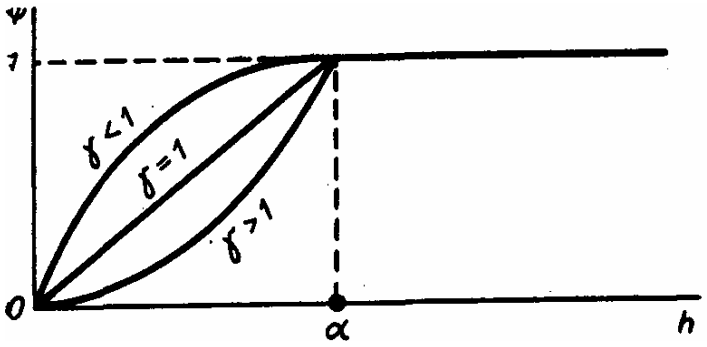


Рис. 6.3.1. График функции $\psi(h)$.

Итак, задача адаптации сводится к формированию такой функции ψ и преобразованию значения весов h_j. Естественно воспользоваться следующей рекуррентной формулой:

$$h_{j,N+1} = kh_{j,N} + \lambda\mu_{j,N}, \qquad (6.3.23)$$

где $0<k<1$ и $\lambda>0$ — параметры;

$\mu_{jN} = \begin{cases} 1, & \text{если } j\text{-й блок использовался в момент } N; \text{ в} \\ 0 & \text{противном случае.} \end{cases}$

Видно, что при постоянном использовании j-го блока $h_{iN} \to \lambda / (1 — k)$, а при неиспользовании $h_{iN} \to 0$.

Функцию $\psi(h)$ (рис. 6.3.1) удобно представить в виде

$$\psi(h) = \begin{cases} (h/\alpha)^{\gamma} & \text{при} \quad h<\alpha; \\ 1 & \text{при} \quad h\geqslant\alpha. \end{cases} \qquad (6.3.24)$$

Ее параметрами являются $0 < \alpha < \lambda / (1 — k)$ и $\gamma>0$. Завершает определение алгоритма φ задание начального значения параметра h для веса новых блоков:

$$h_{j0} = \delta \qquad (j = 1, ..., m). \qquad (6.3.25)$$

Таким образом, алгоритм φ адаптации памяти ЭВМ в сети рандомизирован и однозначно определяется набором из пяти неотрицательных параметров

$$C = (k, \lambda, \alpha, \delta), \qquad (6.3.26)$$

изменяемых в очевидных пределах

$$S: \lambda>1-k, \quad 0<\alpha\leqslant\frac{\lambda}{1-k}. \qquad (6.3.27)$$

С помощью этих параметров оптимизируется работа ЭВМ: минимизируется ее обращаемость к банку данных в процессе решения задач.

Как видно, описанный алгоритм адаптивного распределения памяти по сети ЭВМ имеет ярко выраженную иерархическую

структуру. На первом (нижнем) уровне иерархии производится адаптивное образование памяти каждой ЭВМ и адаптация этого процесса

$$Q(C, \nu_i) \to \min_{C \in S} \quad (6.3.28)$$

по критерию $Q(C, \upsilon_i)$ — обращаемости данной i-й ЭВМ к банку данных. На втором уровне производится адаптация параметров N (6.3.18) — порогов каждой ЭВМ, т. е. решается задача

$$\tilde{Q}(N) \to \min_{N \in \tilde{S}}, \quad (6.3.29)$$

где $\tilde{Q}$ — выбранный критерий функционирования всей сети, например среднее время решения одной задачи в сети и т. д., а $\tilde{S}$ — множество допустимых порогов:

$$\tilde{S}: \nu_i > 0 \quad (i = 1, \ldots, n).$$

Таким образом, эффективность алгоритма эволюционной адаптации памяти каждой ЭВМ сети и сети в целом зависит от набора параметров (6.3.26) и (6.3.18), которые следует адаптировать параметрическим образом.

Эксперименты по эволюционной адаптации такого рода показали работоспособность и эффективность данного подхода даже без параметрической адаптации [143, 144, 211, 282—284].

§ 6.4. Эволюционная адаптация структуры решающих правил

Проблема принятия оптимального решения Y в ситуации X сводится к синтезу решающего правила R, связывающего ситуацию и решение:

$$Y = R(X). \quad (6.4.1)$$

Простейшей задачей принятия решений является задача о распознавании (классификации) двух образов, в которой $Y \in \{0; 1\}$, где 0 и 1 являются именами различаемых образов (классов). Рассмотрим эту проблему с точки зрения эволюционной адаптации [62].

Задача обучения распознаванию образов заключается в построении оптимального решающего правила R^* из некоторого априорно выбранного класса функций, определяющего структуру решающего правила. Если структура выбрана удачно, то в результате обучения возможно построить хорошее решающее правило. Однако обычно имеющейся априорной информации

оказывается недостаточно для правильного выбора структуры решающего правила, поэтому построенное таким образом решающее правило может значительно отличаться от требуемого, рационального (не говоря уже об оптимальном).

В связи с этим возникает важная проблема адаптации структуры решающего правила к конкретной задаче распознавания. Для ее решения необходимо в процесс обучения ввести процедуру поиска оптимальной структуры решающего правила, т. е. сочетать процесс обучения решающего правила с оптимизацией его структуры.

6.4.1. Постановка задачи

Задача обучения классификации объектов, как известно, заключается в построении в процессе обучения решающего правила вида

$$R(X) = \begin{cases} 1, & \text{если} \quad X \in B; \\ 0, & \text{если} \quad X \in C, \end{cases} \quad (6.4.2)$$

где X — n-мерный вектор, описывающий распознаваемый объект; B и C — множества, соответствующие двум различаемым классам объектов (объекты класса B имеют имя «1», а класса C — имя «0»).

Выражение (6.4.2) реализует в n-мерном пространстве признаков $\{X\}$ разделяющую поверхность для всех объектов множества $B \bigcup C$. Однако для определения решающего правила $R(X)$ используется только некоторое подмножество объектов из B и С, составляющее конечную обучающую последовательность длины $L<\infty$. Поэтому задача обучения классификации обычно представляется как задача аппроксимации заранее неизвестной разделяющей поверхности другой поверхностью, достаточно близкой к первой, но построенной на базе конечной обучающей последовательности.

Один из наиболее распространенных подходов к решению этой задачи заключается в представлении искомой разделяющей функции в виде разложения в ряд по некоторой системе функций $\Phi(X)$, выбираемой однократно из априорно заданного класса $\mathfrak{L}_\Phi$, т. е. $\Phi(X) \in \mathfrak{L}_\Phi$. Искомое решающее правило в этом случае описывается выражением

$$Y = R(X) = \operatorname{sg} \sum_{i=1}^{m} v_i \varphi_i(X) = \operatorname{sg} [V, \Phi(X)], \quad (6.4.3)$$

где

$$\operatorname{sg} z = \begin{cases} 1 & \text{при} \quad z \geqslant 0; \\ 0 & \text{при} \quad z < 0. \end{cases} \tag{6.4.4}$$

Система функций $\Phi(X) = \{\varphi_1(X), \ldots, \varphi_m(X)\}$ определяет структуру правила $R(X)$, $V = (v_1, \ldots, v_m)$ — вектор его весов, а через $[V, \Phi(X)]$ обозначено скалярное произведение векторов V и $\Phi(X)$.

Задача построения оптимального решающего правила R заключается в определении параметров V таким образом, чтобы обеспечить наилучшую аппроксимацию действительной разделяющей поверхности, т. е. эффективно приблизить искомое правило (6.4.3) к действительному (6.4.2).

Обозначим через $P(V,\Phi)$ эмпирический риск (число ошибок на обучающей последовательности) искомого решающего правила R (6,4.3). Тогда задачу наилучшей аппроксимации можно рассматривать как задачу минимизации эмпирического риска по параметрам V в процессе обучения при неизменной, априорно выбранной структуре Ф, т. е.

$$P(V, \Phi) \to \min_{V \in R^m} \Rightarrow V^*, \tag{6.4.5}$$

где R^m — евклидово пространство размерности m, *а* V^* — результат решения задачи. Решение (6.4.5) и есть обучение.

Для решения этой задачи разработаны многочисленные алгоритмы обучения. Общая блок-схема такого обучения приведена на рис. 6.4.1. В зависимости от выбора системы функций $\Phi(X)$ схема может описывать метод потенциальных функций [13], классический перцептрон [199] и др.

Однако во многих конкретных задачах имеющиеся априорные соображения либо оказываются недостаточными для правильного выбора системы функций $\Phi(X)$, либо отсутствуют. В таких случаях естественно адаптировать структуру Ф решающего правила R в процессе обучения. Тогда оптимальное решающее правило, обеспечивающее наилучшую аппроксимацию, будет определяться из условия минимума функционала эмпирического риска при вариации структуры решающего правила:

$$P(V, \Phi) \to \min_{\Phi \in \mathcal{L}_\Phi} \min_{V \in R^m} \Rightarrow \Phi^*, V^*. \tag{6.4.6}$$

Блок-схема этой процедуры приведена на рис. 6.4.2.

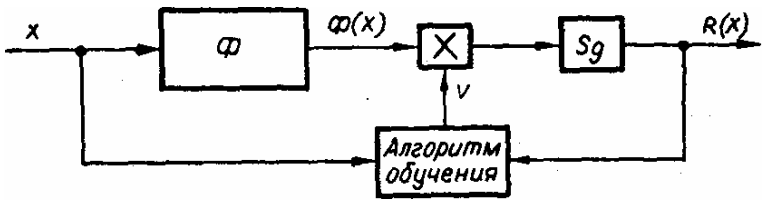


Рис. 6.4.1. Блок-схема обучения решающего правила при заданной структуре.

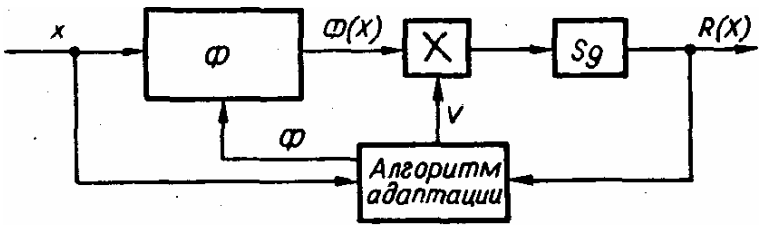


Рис. 6.4.2. Блок-схема обучения с адаптацией структуры решающего правила.

Для решения такой задачи удобно представить систему функции Ф в виде

$$\Phi = \Phi'(X, S),$$

где Ф′ — заданная система функций, а S — вектор, кодирующий структуру этой системы. Тогда задачу адаптации структуры решающего правила можно сформулировать как задачу отыскания оптимальной структуры S^* в процессе обучения:

$$P(V, \Phi) \to \min_{S \in \Omega} \min_{V \in R^m} \Rightarrow S^*, V^*, \tag{6.4.7}$$

где Ω — множество допустимых структур. Ввиду обилия допустимых структур S решение задачи адаптации целесообразно производить методами случайного поиска. (Заметим, что возможна и параметризация структуры перцептрона, которая была успешно использована в работе [176], где для этого применялась процедура многомерной линейной экстраполяции [186].)

6.4.2. Адаптация структуры перцептрона

Адаптация структуры решающего правила исследовалась в классе перцептронных решающих правил, определяемых выражением (6.4.3), в котором v_i — вес i-го *А элемента*, а $\varphi_i(X,S)$ — пороговая функция, реализуемая i-м A-элементом перцептрона:

$$\varphi_i(X, S) = \begin{cases} 1, & \text{если } \sum_{j=1}^{n} s_{ji} x_j \geqslant \theta_i; \\ 0, & \text{если } \sum_{j=1}^{n} s_{ji} x_j < \theta_i, \end{cases} \tag{6.4.8}$$

где x_j — выходной сигнал j-го элемента сетчатки ($x_j \in \{0; 1\}$, $j = 1, \ldots, n$); θ_i — порог j-го A-элемента; s_{ji} — коэффициент связи между j-м элементом сетчатки и i-м A-элементом ($s_{ji} \in \{-1; 0; 1\}$ в зависимости от наличия и знака связи, $i = 1, \ldots, m$).

Матрица $S = \| s_{ij} \|_{n \times n}$, составленная из коэффициентов связи с учетом ограничения

$$\sum_{i=1}^{n} |s_{ij}| = l \qquad (j = 1, \ldots, m), \quad (6.4.9)$$

где l — число входов каждого элемента, полностью определяет структуру решающего правила. В обычном перцептроне эта матрица определялась однократно и случайно, причем процедура обучения (6.4.5) сводилась к формированию вектора V весов A-элементов таким образом, чтобы на заданной обучающей последовательности L удовлетворялись соотношения

$$\sum_{j=1}^{m} v_j \varphi_j(X_h, S) = \begin{cases} \geqslant 0 & \text{для } X_h \in B; \\ < 0 & \text{для } X_h \in C \end{cases} (h = 1, \ldots, L). \quad (6.4.10)$$

Для обучения использовалась α-система подкрепления с коррекцией ошибок [199]. В случае, если величина подкрепления принимается равной ± 1, этот алгоритм может быть записан в следующем виде:

$$v_j^{h+1} = \begin{cases} v_j^h & \text{при правильном ответе перцептрона;} \\ v_i^h + (1 - 2Y_h)\varphi_j(X_h, S) & \text{при ошибке,} \end{cases} \quad (6.4.11)$$

где h — номер предъявляемого объекта (или такт обучения), а Y_h — выход перцептрона, определяемый соотношением (6.4.3) для X_h.

Реализуемое перцептроном решающее правило, таким образом, может быть представлено в виде $Y = R(X, V, S)$, где матрица S определяет структуру, а вектор V — параметры.

В работе [199] показано, что для случая непересекающихся классов объектов элементарный перцептрон с исходной матрицей S и бесконечным числом A-элементов всегда позволяет построить точную разделяющую поверхность. Однако при ограниченном числе A-элементов (что всегда бывает в реальных системах) построенное из условия (6.4.5) решающее правило может значительно отличаться от оптимального, т. е. соотношение (6.4.10) может не выполняться для непересекающихся классов. Это, например, будет иметь место тогда, когда непересекающиеся в исходном n-мерном пространстве рецепторов $\{X\}$ перцептрона классы оказываются пересекающимися в m-мерном пространстве A-элементов (т. е. когда объектам разных классов соответствует одинаковый бинарный код, образованный выходами A-элементов) .

В таких случаях в процессе обучения необходима адаптация структуры связей между сетчаткой и *A*-элементами к конкретной задаче, т. е. к заданной обучающей последовательности объектов. Для этого в перцептрон должен быть дополнительно введен процесс адаптации — поиск оптимальной матрицы *S*, минимизирующей эмпирический риск (6.4.6), причем ввиду многомерности структуры связей *A*-элементов с сетчаткой для ее оптимизации наиболее целесообразно использовать методы случайного поиска (см. главу 3). Здесь можно применить следующие алгоритмы случайного поиска.

1. Алгоритм случайного поиска структуры связей всех A-элементов с сетчаткой. Этот алгоритм представляет собой случайный перебор матриц *S*:

$$S_N = \Xi, \tag{6.4.12}$$

где S_N — матрица *S* на *N*-м шаге случайного поиска; Ξ — случайная матрица $n \times m$, элементы которой случайны и равны $s_{ij} \in \{-1;\ 0;\ 1\}$ с учетом ограничения (6.4.9).

Целенаправленность перебора (6.4.11) достигается благодаря специальной организации памяти в процессе поиска. Рекуррентные выражения для содержимого памяти имеют вид

$$S^*_{N+1} = \begin{cases} S^*_N & \text{при} \quad Q(S_{N+1}) \geqslant Q^*_N; \\ S_{N+1} & \text{при} \quad Q(S_{N+1}) < Q^*_N; \end{cases}$$

$$Q^*_{N+1} = \begin{cases} Q^*_N & \text{при} \quad Q(S_{N+1}) \geqslant Q^*_N; \\ Q(S_{N+1}) & \text{при} \quad Q(S_{N+1}) < Q^*_N \end{cases} \tag{6.4.13}$$

где $Q(S_N)$ — значение минимизируемого показателя качества — числа неправильных реакций перцептрона на обучающую последовательность для матрицы связей S_N; Q^*_N — наименьшее хранимое в памяти значение показателя качества за *N* шагов поиска; S^*_N — оптимальная матрица за *N* шагов поиска, обеспечивающая наименьшее значение критерия:

$$Q^*_N = Q(S^*_N) = \min_{t=1,\ldots,N} Q(S_t). \tag{6.4.14}$$

2. Алгоритм случайного поиска оптимальной матрицы S с отбором A-элементов. Здесь показатель качества вычисляется отдельно для каждого *A*-элемента и поиск строится на тех элементах, которые дают значения показателя качества, большие заданного порога *q*.

Представив матрицу *S* в виде $S = \| s_1, \ldots, s_m \|$, где s_i — вектор-столбец этой матрицы, соответствующий соединениям входов

j-го A-элемента с сетчаткой (j=1,...,m), этот алгоритм можно записать как

$$S_{N+1} = \begin{cases} S_N, & \text{если } Q_j(s_{jN}) \le q \text{ для всех } j = 1, ..., m; \\ \Xi', & \text{если } Q_j(s_{jN}) > q \text{ хотя бы для одного } j = 1, ..., m. \end{cases} \quad (6.4.15)$$

Здесь $Q_i(s_{iN})$ — значение минимизируемого показателя качества — числа неправильных реакций j-го A-элемента перцептрона на объекты обучающей последовательности на N-м шаге случайного поиска; q — некоторый заданный порог, общий для всех A-элементов; Ξ' — случайная матрица ($n \times m$) с вектор-столбцами ξ_j, т. е. $\Xi = (\xi_1, ..., \xi_m)$, где

$$\xi_j = \begin{cases} s_{jN}, & \text{если } Q_j(s_{jN}) \le q; \\ \xi, & \text{если } Q_j(s_{jN}) > q. \end{cases} \quad (6.4.16)$$

Здесь ξ — случайный вектор, координаты которого равновероятно принимают значения из {-1; 0; 1} с учетом ограничения (6.4.9).

6.4.3. Оценка вероятности образования оптимальной структуры перцептрона в процессе адаптации

В тех случаях, когда классы предъявляемых объектов пересекаются, принципиально невозможно построить в процессе обучения такое решающее правило, которое обеспечивало бы безошибочное распознавание объектов на экзамене. Но при этом может быть найдено некоторое оптимальное решающее правило, которое обеспечивает минимальную вероятность ошибки распознавания.

Вероятность построения такого оптимального решающего правила в элементарном перцептроне можно увеличить двумя способами: 1) увеличением общего числа A-элементов; 2) адаптацией структуры перцептрона в процессе обучения, т. е. введением поиска оптимальной матрицы S. Такая адаптация структуры S позволяет избежать дополнительного увеличения числа A-элементов за счет увеличения числа шагов поиска структуры.

Вероятность образования оптимального решающего правила, минимизирующего ошибку классификации, исследовалась для двух пересекающихся классов объектов, построенных следующим образом. Эталонный объект каждого класса представлялся n-мерным вектором, координаты которого

$$x_i^{\ni} = \begin{cases} 1 & \text{с вероятностью } p_1; \\ 0 & \text{с вероятностью } 1 - p_1 \end{cases} \quad (i=1,...,n), \quad (6.4.17)$$

а координаты объектов каждого класса отличались от соответствующей координаты эталонного объекта этого класса с вероятностью P2< 1/2, т. е.

$$x_i^t = \begin{cases} \bar{x}_i^э \text{ с вероятностью } p_2; \\ x_i^э \text{ с вероятностью } 1 - p_2 \end{cases} \quad (i = 1, \ldots, n), \tag{6.4.18}$$

где t — номер объекта; вероятности p_1 и p_2 заданы. В работах [58, 61] показано, что для минимизации ошибки распознавания необходимо максимизировать вероятность образования оптимальной структуры S^* перцептрона. В соответствии с этим минимальное число A-элементов перцептрона, образующее оптимальную структуру, может быть подсчитано как

$$q = \begin{cases}]k_1/l_1[, & \text{если} \quad k_1/l_1 \geqslant k_2/l_2, \\ ]k_2/l_2[, & \text{если} \quad k_1/l_1 < k_2/l_2. \end{cases} \tag{6.4.19}$$

Здесь k_1 — число признаков эталона класса B (k_2 — класса C), значения которых равны 1 и которые не совпадают с соответствующими координатами эталона класса C (класса B) ($k_1 + k_2 = k$, где k — расстояние по Хэммингу между эталонными объектами); l_1 (l_2) — число входов для возбуждающих (тормозящих) связей A-элемента ($l_1 + l_2 = 1$), а символ][означает округление в большую сторону до ближайшего целого. В этом случае за оптимальную структуру S^* принимается такая структура соединений A-.элементов с сетчаткой, в которой все k_1 и k_2 рецепторов соединяются соответствующими связями с A-элементами и A-элементы возбуждаются только объектами класса B. При фиксированных единичных весах A-элементов в перцептроне с q A-элементами оптимальное решающее правило может быть построено без дополнительного обучения только за счет образования такой оптимальной структуры.

Вероятность образования такой структуры S^* при однократной случайной реализации связей A-элементов с сетчаткой определяется выражением [58]

$$P = \sum_{\nu=0}^{k} (-1)^\nu \binom{k_1}{\nu} (n-\nu)^{l_1 q} \sum_{\nu=1}^{k_2} (-1)^\nu \binom{k_2}{\nu} (n-\nu)^{l_2 q} n^{-lq}. \tag{6.4.20}$$

При неограниченном увеличении количества A-элементов m вероятность будет стремиться к единице. Когда число A-элементов равно минимальному (6.4.19) и неограниченно увеличивается число шагов случайного поиска структуры, то вероятность ее отыскания также стремится к единице. Поэтому достичь заданной вероятности ошибки распознавания можно за счет увеличе-

ния числа шагов адаптации без дополнительных затрат, вызванных увеличением количества *A*-элементов. При исходном числе *A*-элементов, равном *q,* соотношение между числом *N* шагов поиска и количеством *m A*-элементов, которые необходимы для достижения одинаковой вероятности образования оптимальной структуры, будет определяться как $N \leqslant \binom{m}{q}$. Это означает, что при исходном минимальном числе *A*-элементов в среднем следует сделать не более $\binom{m}{q}$ шагов адаптации методом случайного поиска структуры, для того чтобы получить ту же вероятность образования оптимальной структуры, что и при количестве *A*-элементов, равном *m.* В случае, если исходное число *q*' *A*-элементов взято несколько большим минимального, необходимое число шагов случайного поиска уменьшается:

$$N \approx \binom{m}{q} \Big/ \binom{m}{q'},$$

что позволяет повысить эффективность адаптации. Но при этом уже необходимо дополнительное обучение перцептрона, заключающееся в сведении к нулю весов тех *A*-элементов, которые возбуждаются объектами обоих классов и не участвуют в образовании оптимальной структуры.

В реальных системах с ограниченным числом *A*-элементов вероятность образования такой оптимальной структуры довольно мала, а число шагов адаптации для ее оптимизации также ограниченно. Поэтому оказывается целесообразным поставить задачу минимизации ошибки распознавания как задачу максимизации вероятности образования такой структуры, которая реализует наличие хотя бы двух *A*-элементов, каждый из которых реагирует только на объекты своего класса [61]. В процессе обучения веса этих *A*-элементов сформируются с противоположными знаками, а веса *A*-элементов, реагирующих как на объекты одного, так и другого класса, будут, сведены к нулю. В этом случае для образования оптимальной структуры достаточно, чтобы возбуждающие или тормозящие связи *A*-элементов просто попадали в интервал, соответствующий величине k_1 или k_2 на сетчатке, независимо от того, как эти связи распределяются внутри интервала (т. е. здесь достаточно, чтобы хотя бы один из рецепторов интервала k_1 или k_2 был соединен соответствующими связями с *A*-элементами [61]).

Допустим, что в процессе обучения перцептрона веса *A*-элементов, реагирующих только на объекты класса *B* или только класса *C*, принимают значения +1 или -1 соответственно, а веса *A*-элементов, реагирующих на объекты обоих классов, при-

нимают значения 0. Тогда значения координат вектора V весов A-элементов будут определяться следующим образом:

$$v_j = \begin{cases} +1 \text{ с вероятностью } P_B; \\ 0 \text{ с вероятностью } 1 - P_B - P_C; \\ -1 \text{ с вероятностью } P_C \end{cases} \quad (j = 1, ..., m), \tag{6.4.22}$$

где P_B (P_C) — вероятность того, что A-элемент возбуждается только объектами класса B (класса C) [61].

В этом случае вероятность построения оптимальной классификации будет определяться вероятностью образования такого вектора весов V^*, у которого будут хотя бы две координаты, из которых одна имеет значение -1, а другая +1. Эта вероятность может быть представлена в следующем виде:

$$P_m(V^*) = \sum_{m_1=1}^{m-1} \sum_{m_2=1}^{m-m_1} \frac{m!}{m_1! m_2! (m - m_1 - m_2)!} \times \\ \times P_B^{m_1} P_C^{m_2} (1 - P_B - P_C)^{m - m_1 - m_2}, \tag{6.4.23}$$

где m — общее число A-элементов перцептрона; m_1 (m_2) — количество A-элементов, имеющих положительный (отрицательный) вес.

Вероятность $P_m(V^*)$ стремится к единице как при неограниченном увеличении m, так и при фиксированном $m=m'$ и неограниченном увеличении числа шагов поиска структуры связей между сетчаткой и A-элементами. Соотношение между числом шагов N поиска и количеством m A-элементов в этом случае определяется выражением [61]:

$$N \approx \frac{(1 - P_B)^m + (1 - P_C)^m - (1 - P_B - P_C)^m - 1}{[(1 - P_B)^{m'} + (1 - P_C)^{m'} + (1 - P_B - P_C)^{m'} - 1](m - m')}, \tag{6.4.24}$$

которое показывает, сколько в среднем следует сделать шагов случайного поиска структуры S при заданном числе m' A-элементов, чтобы получить ту же вероятность построения оптимальной классификации, что и при увеличении числа A-элементов на единицу.

6.4.4. Модельные эксперименты

Адаптация структуры перцептрона методом случайного поиска исследовалась на примере распознавания двух классов объектов, образованных следующим образом [60].

В соответствии с выражениями (6.4.17) и (6.4.18), в которых $n = 64$; $p_1 = 0{,}5$; $p_2 = 0{,}3$; $t = 50$, задавались два эталонных объекта, и для каждого из них был образован свой класс объектов. Полученные таким образом 100 объектов были затем дополнительно разбиты случайным образом на два класса по 50 объектов.

Для решения данной задачи на ЭВМ моделировался трехслойный перцептрон с сетчаткой из 64 рецепторов ($n = 64$) и числом A-элементов, равным 100 и 25. Каждый A-элемент имел пять входов: два входа для тормозящих и три — для возбуждающих связей с сетчаткой. Порог θ каждого A-элемента был принят равным 1. Исходная структура связей выбиралась случайным образом.

Полученная случайная обучающая последовательность имела случайные признаки, и для ее распознавания было проведено семь независимых экспериментов. В каждом эксперименте объекты этой обучающей последовательности заново разбивались на два класса случайным образом и проводилось обучение перцептронов с 25 и 100 A-элементами по двум алгоритмам: 1) алгоритм I: α-система подкрепления с коррекцией ошибок (6.4.11); 2) алгоритм II: α-система подкрепления с коррекцией ошибок в комбинации с описанным алгоритмом адаптации структуры (6.4.13).

В качестве показателя качества каждого A-элемента принималось число неправильных реакций A-элемента на объекты обучающей последовательности, и для каждой последовательности задавался ряд значений порога q.

В случае перцептрона со 100 A-элементами оказалось достаточным применить только первый алгоритм обучения, для того чтобы по окончании процесса обучения безошибочно распознать объекты обучающей последовательности во всех семи экспериментах, т. е. для решения задачи распознавания использовалась только процедура (6.4.11).

В случае перцептрона с 25 A-элементами использования для обучения только α-системы подкрепления с коррекцией ошибок оказалось недостаточным для обеспечения безошибочной классификации объектов обучающей последовательности. Здесь обучение проводилось по второму алгоритму, т. е. использовалась процедура (6.4.6). Результаты обучения, осредненные по семи экспериментам, а также доверительные интервалы, соответствующие доверительной вероятности 0,92, приведены на рис. 6.4.3. Видно, что полученные при решении такой модельной задачи экспериментальные результаты хорошо согласуются с приведенными выше теоретическими.

Кроме того, проводилось сравнение перцептрона с адаптивной структурой и алгоритма «Кора» [42] на примере решения

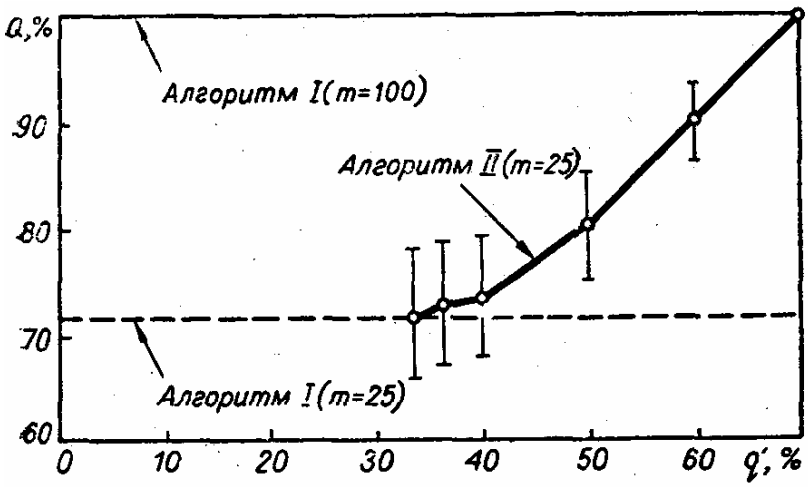


Рис. 6.4.3. Результаты экспериментов по обучению обычного перцептрона (алгоритм I) и перцептрона с адаптивной структурой (алгоритм II).
Q — надежность классификации (доля правильных ответов перцептрона, %); q' — доля (%) правильных реакций A-элемента на объекты обучающей последовательности.

следующей задачи. В соответствии с выражениями (6.4.17) и (6.4.18), в которых принималось: n=32; p_1=0,5; p_2=0,2 и t=48, задавались два класса объектов. Из общего числа 96 объектов случайным образом выбирались четыре различные обучающие последовательности, в которые входило по 32 объекта из каждого класса, а оставшиеся 32 объекта использовались только для экзамена.

Для решения этой задачи на ЭВМ «Минск-32» моделировались алгоритм «Кора» и трехслойный перцептрон с адаптивной структурой связей A-элементов с сетчаткой. Число A-элементов m в перцептроне было принято равным 100 и 25, каждый A-элемент имел два входа для тормозящих и три — для возбуждающих связей с сетчаткой (n=32), порог в каждого A-элемента был принят равным единице. В качестве признаков в алгоритме «Кора» рассматривались конъюнкции не выше третьего ранга. Количество отбираемых признаков принималось равным 10 и 30, а значения заданного минимального числа объектов одного класса в обучающей последовательности, для которых данная конъюнкция истинна, задавались равными 7 и 12 соответственно.

Результаты экзамена, осредненные по четырем независимым экспериментам, а также время обучения приведены в табл. 6.4.1.

Таблица 6.4.1

Алгоритм	Количество		Порог	Надежность классификации; %	Время обучения, мин
	признаков	A-элементов			
«Кора»	10	—	7	71,8	36
	30	—	12	91.8	214
Перцептрон	—	25		73,4	30
	—	100	—	89,1	8

Эти результаты показывают, что введение в перцептрон процедуры адаптации структуры методами случайного поиска позволяет достичь того же эффекта гораздо быстрее, чем с помощью алгоритма «Кора», использующего метод перебора признаков (как видно из таблицы, в одном случае имеем выигрыш времени более чем на порядок).

6.4.5. Применение перцептрона с адаптивной структурой для решения некоторых практических задач распознавания

Перцептрон с адаптивной структурой связей *A*-элементов с сетчаткой использовался для решения ряда практических задач распознавания. Рассмотрим эти задачи.

6.4.5.1. Классификация объектов промышленного производства

Перцептрон с адаптивной структурой использовался для классификации деталей в зависимости от их конструктивно-технологических признаков и характера производства [56].

Для минимизации технологических затрат при изготовлении объектов производства могут быть использованы стандартные (групповые, типовые, нормализованные) технологические процессы, являющиеся оптимальными в данных условиях производства. Выбор таких оптимальных технологических маршрутов проводится на основе классификации объектов производства. Например, распределение деталей по группам (классификация) в зависимости от их конструктивно-технологических признаков и характера производства позволяет однозначно определить конкретные методы и средства изготовления деталей.

Однако, чтобы получить объективную классификацию, минимизирующую затраты на изготовление объектов производства с учетом ограничений, имеющих место для конкретных условий данного предприятия (ограничения на станочный парк, его загрузку и т. д.), необходимо рассчитать стоимость затрат на изготовление для множества различных вариантов технологических процессов. Это очень громоздкая и сложная вычислительная задача, решение которой не всегда возможно из-за отсутствия необходимой информации. Поэтому на практике задача классификации решается экспертным путем. Очевидный интерес представляет имитирование мнения специалиста и выведение формальных решающих правил для целей классификации. Такие правила могут синтезироваться обучаемыми алгоритмами перцептронного типа. Сформулируем задачу классификации.

Пусть имеется некоторое множество объектов, представляющих собой совокупность деталей или изделий промышленного производства. Каждый объект может быть описан некоторым n-мерным вектором X двоичных признаков.

Такими признаками могут быть конструктивные и технологические характеристики данного объекта. Множество объектов в общем случае состоит из определенного числа пересекающихся подмножеств или классов. Здесь под классом объектов подразумевается подмножество деталей, характеризующееся общностью технологического процесса. Например, множество деталей, обрабатываемых на токарных станках, образует один класс, на револьверных — другой и т. д. Поскольку некоторые детали можно с одинаковой эффективностью обработать как на токарных, так и на револьверных станках, то классы таких объектов пересекаются.

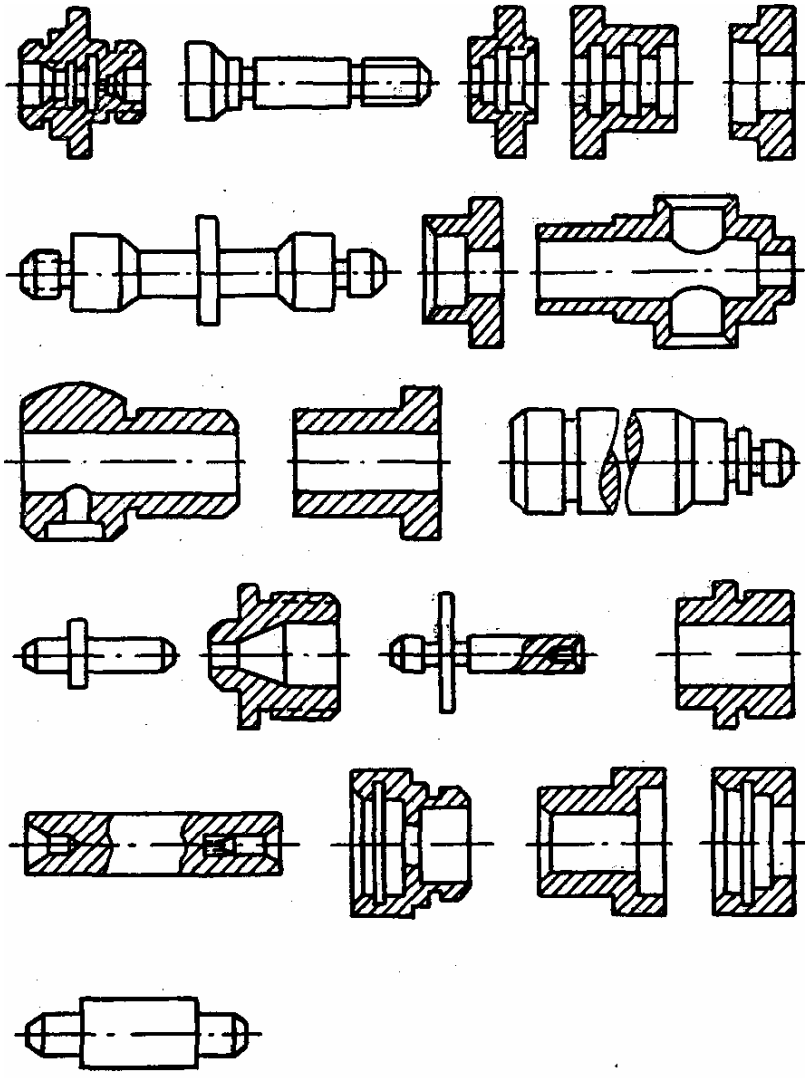


Рис. 6.4.4. Детали, обрабатываемые на токарных станках.

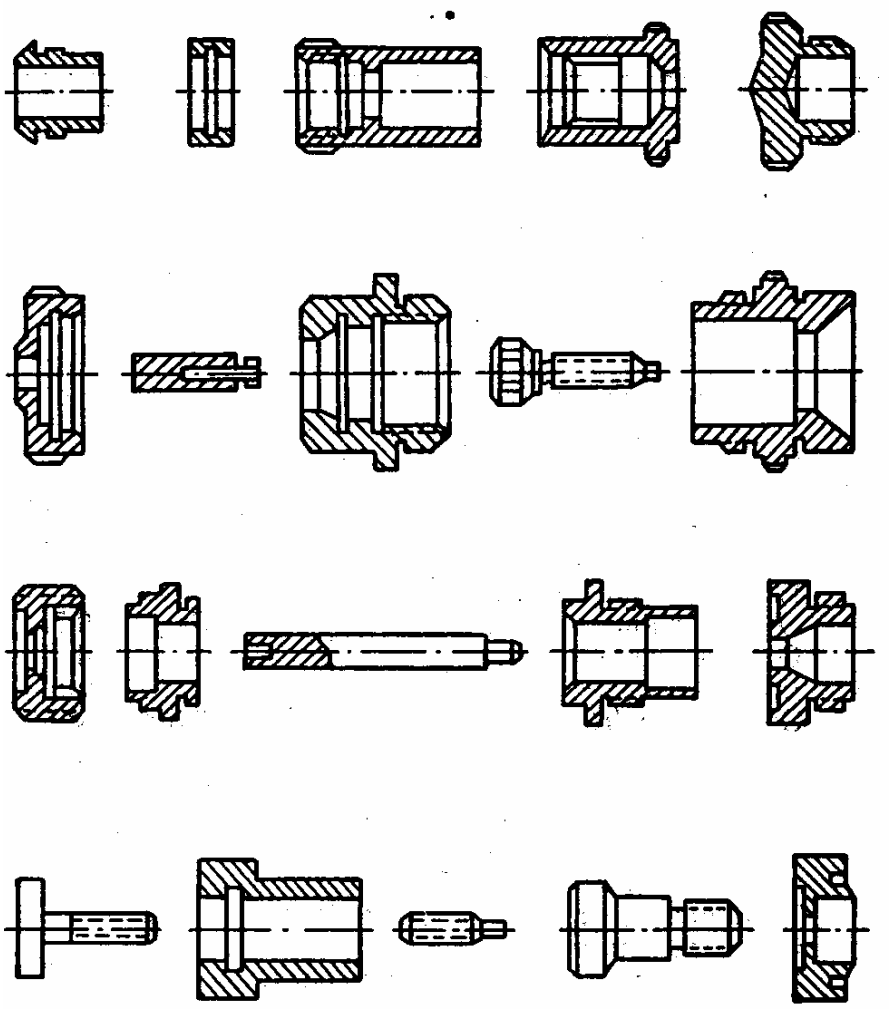


Рис. 6.4.5. Детали, обрабатываемые на револьверных станках.

В случае дихотомии (два класса) задача классификации заключается в построении некоторой решающей функции вида (6.4.2), где B — множество объектов одного класса, C — другого. Обучающая последовательность деталей, разбитых на указанные классы, составляется квалифицированным технологом — экспертом.

Для решения этой задачи может быть использован описанный выше алгоритм трехслойного перцептрона с адаптацией структуры, В качестве примера рассмотрим одну конкретную задачу классификации деталей, обрабатываемых на токарных или револьверных станках. В каждой группе было по 20 объектов (рис. 6.4.4, 6.4.5), каждый из которых описывался 20-мерным вектором двоичных признаков (табл. 6.4.2). На ЭВМ моделировался трехслойный перцептрон с сетчаткой из 20 рецепторов (n = 20) и 25 A-элементами. Каждый A-элемент имел два входа для возбуждающих связей и один — для тормозящей связи с сетчаткой. Порог 9 каждого A-элемента принимался равным единице.

Таблица 6.4.2

№ признака	Характеристика	Значение i-го признака 0	1
1	Наружный диаметр	До 40 мм	40—1000 мм
2	Внутренний диаметр наибольший	До 40 мм	40—1000 мм
3	Длина l	До 200 мм	200—1000 мм
4	Количество ступеней по наружной поверхности с одной стороны	≤3	>3
5	Характер отверстия	Простое	Сложное
6	Количество ступеней в отверстии	<3	>3
7	Высший класс точности наружных поверхностей	≤3	>3
8	Высший класс точности внутренних поверхностей	≤4	>4
9	Наличие особых поверхностей	Нет	Есть
10	Общая длина правых ступеней	Свыше 0,7 l	До 0,7 l
11	Общая длина левых ступеней	Свыше 0,7 l	До 0,7 l
12	Вид заготовки	Из прутка	Штучная
13	Отношение l/D (D — диаметр)	$0.5 < l / D < 5$	Вне указ.
14	Внутренний диаметр наименьший	До 20 мм	Свыше 20 мм
15	Количество ступеней по наружной поверхности с другой стороны	≤3	>3
16	Требования по соосности наружных поверхностей	Нет	Есть
17	Требования по соосности внутренних поверхностей	Нет	Есть
18	Требования по перпендикулярности осей цилиндрических поверхностей к торцам	Нет	Есть
19	Особые требования по точности	Нет	Есть
20	Программа (план)	Свыше 50 шт. в месяц	Менее 50 шт. в месяц

Из общего числа объектов, равного 40, был выбран ряд обучающих последовательностей, и для каждой из них проводилось четыре независимых эксперимента: четыре раза случайным образом выбиралась исходная структура и для каждой структуры проводилось обучение с последующим экзаменом по оставшейся части объектов, не вошедшей в обучающую последовательность. Для обучения использовались два алгоритма: алгоритм I (α-система подкрепления с коррекцией ошибок) и предложенный выше алгоритм II (а-система подкрепления с коррекцией ошибок в комбинации с алгоритмом адаптации структуры).

Результаты экзаменов, осредненные по четырем экспериментам, для обучающей последовательности из 12 объектов приведены на рис. 6.4.6.

Полученные результаты показали, что использование перцептрона с адаптивной структурой для решения задачи классифи-

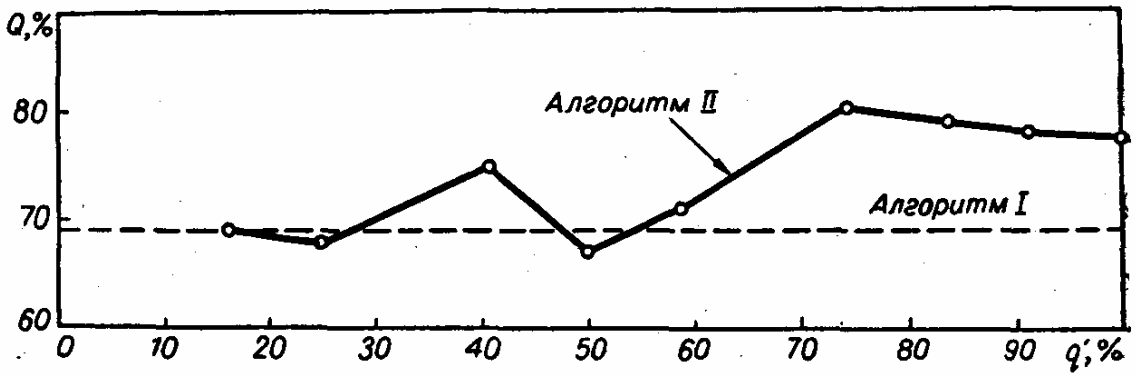


Рис. 6.4.6. Результаты классификации деталей, представленных на рис. 6.4.4—6.4.5. Обозначения те же, что и на рис. 6.2.3.

кации объектов промышленного производства позволяет повысить надежность классификации в среднем на 10—15% по сравнению с обычным перцептроном; надежность классификации при этом достигает 80%.

6.4.5.2. Классификация в задачах технологического проектирования [63]

Одна из основных задач технологического проектирования состоит в создании наиболее целесообразных технологических процессов, т. е. в определении последовательности технологических операций и выборе средств производства, позволяющих при наличии определенных ресурсов изготавливать требуемые объекты производства с минимальными затратами. Характерной особенностью этой задачи является наличие множества возможных технологических процессов и средств изготовления одинаковых объектов, что создает многовариантность задачи техно-логического проектирования. Ее решение обусловлено различием видов производства, уровнем развития технологии, состоянием ресурсов, спецификой предприятий и другими факторами. Формализация этой задачи приводит либо к значительным трудностям при составлении формальных описаний процедур проектирования технологических процессов, либо — при наличии таких описаний — к большим вычислительным затратам, которые в ряде случаев могут стать непреодолимыми ввиду большого количества ограничений, многоэкстремальности и многокритериальности указанной задачи и т. п.

Именно поэтому обычно достаточно сложные задачи технологического проектирования не решаются путем формализации. Эти трудноформализуемые задачи на современном этапе развития моделирования и вычислительной техники может решить.

лишь эксперт. Стандартный путь решения таких задач и заключается в использовании решений экспертов-технологов. Принимаемое ими решение носит, естественно, субъективный характер и зависит от индивидуального опыта эксперта, его способностей к экстраполяции, обилия и структуры ассоциаций и других сугубо психологических факторов.

Специфика современного производства заключается в том, что ввиду наличия большого разнообразия методов обработки сырья и деталей и множества уже разработанных технологических процессов при подготовке производства новых объектов в большинстве случаев целесообразнее выбрать и использовать уже разработанный и действующий технологический процесс, чем создавать новый.

Именно поэтому одним из основных элементов технологического проектирования является классификация (объектов производства, технологических операций и т. д.), позволяющая для объектов одного класса принимать соответствующие типовые решения. Например, разделение деталей на группы по конструктивно-технологическим признакам позволяет связать их конструкцию с технологией изготовления, а для каждой такой группы устанавливается типовое (стандартное) технологическое решение.

Указанные особенности задач технологического проектирования и привели к. использованию для их решения методов распознавания образов, позволяющих имитировать процесс принятия решения специалистами-технологами. Выше (см. п. 6.4.5.1) было описано применение методов распознавания образов для классификации деталей, обрабатываемых на токарных или револьверных станках. В работе [43] методы распознавания образов использовались для определения необходимости правки штампованных поковок и для назначения напуска на отверстия в деталях. Формирование таблицы, позволяющей синтезировать алгоритмы классификации простейших плоских штампованных деталей, рассматривалось в работе [83]. Вопрос выбора комплексного технологического процесса этими методами ставился также в работе [206]. Здесь в процессе обучения необходимо было выполнение двух условий: гипотезы компактности и некоррелированности признаков. В этом случае операции технологического процесса определялись объединением идентичных операций.

В практических задачах определения технологических процессов указанные условия могут не выполняться, разделяющие поверхности между классами могут быть достаточно сложными, поэтому для решения таких задач необходимо иметь адаптивное решающее правило, позволяющее решать широкий класс задач. Ниже рассматривается синтез такого правила на примере задачи определения последовательности технологических операций.

Пусть, как и выше, имеется множество объектов промышленного производства, причем каждый из них описывается n-мерным двоичным вектором признаков X. Такими признаками могут быть конструктивные и технологические характеристики объекта — например, геометрия, размеры, материал, точность обработки и т. д. Для изготовления этих объектов в общем случае может быть использовано k различных технологических процессов, каждый из которых представляет собой определенную последовательность из r операций:

$$Z_i = (z_1^i, ..., .z_r^i) \quad (i = 1, ..., k), ...(6.4.25)$$

где элемент z_i^i обозначает j-ю технологическую операцию ($j = = 1, ..., r$) в i-м процессе, причем в последовательности (6.4.25) операции могут повторяться (например, сверление отверстий может повторяться после пробивки фасонного отверстия). Тогда исходное множество объектов может быть разбито на k пересекающихся подмножеств или классов, характеризующихся общностью процесса. Каждому технологическому процессу Z_i соответствует свое подмножество объектов, которые целесообразно обрабатывать этим способом.

Задача синтеза решающего правила, позволяющего определять для каждого объекта технологический процесс (6.4.25), в котором возможны различные сочетания технологических операций, в общем случае является достаточно сложной. Поэтому решающее правило строят на более низком иерархическом уровне: в последовательности (6.4.25) априорно определяется место каждой из операций и задача сводится к решению вопроса, использовать эту операцию или нет. В этом случае комплексный технологический процесс может быть закодирован в виде бинарного вектора

$$Y_i = (y_1^i, ..., y_t^i) \quad (i = 1, ..., k), \qquad (6.4.26)$$

где y_j^i принимает значение 1, если в i-м технологическом процессе Z_i соответствующая технологическая операция на j-м месте присутствует, и значение 0 — в противном случае ($i = 1, ..., k$; $j = 1, ..., t$).

Например, если такой комплексный процесс состоит из пяти технологических операций:

$$Y = (y_1, y_2, y_3, y_4, y_5), \qquad (6.4.27)$$

где y_1 — операция «отрезать заготовку», y_2 — «вырезать наружный контур», y_3 — «пробить отверстие», y_4 — «нарезать резьбу», y_5 — «выгнуть деталь», то технологические процессы «отрезать заготовку, пробить отверстие, выгнуть деталь», «отрезать заготовку, вырезать наружный контур, пробить отверстие, нарезать

резьбу» и «отрезать заготовку, выгнуть деталь» будут соответственно описываться кодами

$$Y_1 = (1, 0, 1, 0, 1); \quad Y_2 = (1, 1, 1, 1, 0); \quad Y_3 = (1, 0, 0, 0, 1). \qquad (6.4.28)$$

Тогда задачу обучения классификации технологических процессов можно свести к построению в процессе обучения t решающих функций перцептронного типа:

$$y_j = R_j(X) \quad (j = 1, ..., t) \qquad (6.4.29)$$

на базе обучающей последовательности объектов, в которой для каждого объекта X известен технологический процесс Z.

Рассмотрим пример решения одной конкретной задачи синтеза комплексного технологического процесса штамповочного производства. Код детали для решения поставленной задачи описывался 32-мерным вектором бинарных признаков X, который формировался из исходного описания объектов, представляющих собой штамповочные детали. Компоненты этого вектора определяли наличие и значения соответствующих конструктивно-технологических признаков, влияющих на выбор варианта технологического процесса штамповки.

Так, компоненты x_1 и x_2 вектора X в зависимости от длины детали l принимали значения, указанные в табл. 6.4.3.

Таблица 6.4.3

Значени	компоне нт	Длина детали l, мм
0	0	$l \le 80$
0	1	$80 < l \le 200$
1	0	$200 < l < 500$
1	1	$500 < l$

Компоненты x_3 и x_4 учитывали марку и толщину S материала, из которого изготавливалась деталь:

$$x_3 = \begin{cases} 0 & \text{для алюминия;} \\ 1 & \text{для стали;} \end{cases}$$

$$x_4 = \begin{cases} 0, & \text{если } S<2 \text{ мм;} \\ 1, & \text{если } S\ge 2 \text{ мм.} \end{cases} \qquad (6.4.30)$$

Другие конструктивные признаки учитывались остальными компонентами вектора X аналогичным образом.

В качестве комплексного технологического процесса рассматривалась последовательность из четырнадцати технологических

операций (t=14), причем в него были включены именно те операции, выбор которых не мог однозначно определяться различными их комбинациями. Эти операции технологического маршрута представлялись в следующем порядке:

z_1 — разрезать полосу на заготовки (один рабочий);
z_2 — разрезать полосу на заготовки (два рабочих);
z_3 — пробить круглое отверстие;
z_4 — пробить фасонное отверстие;
z_5 — пробить группу отверстий;
z_6 — просверлить отверстие;
z_7 — снять фаски в отверстиях под резьбу;
z_8 — нарезать резьбу;
z_9 — выгравировать надписи;
z_{10} — выгнуть деталь на ручном прессе;
z_{11} — выгнуть деталь на механическом прессе;
z_{12} — сварить деталь газоэлектрической сваркой;
z_{13} — сварить деталь газовой сваркой;
z_{14} — изготовить шаблон.

В зависимости от наличия или отсутствия тех или других . операций данный маршрут позволял получать технологические процессы изготовления различных объектов в производстве.

Для решения этой задачи на ЭВМ [63] моделировался перцептрон с сетчаткой из 32 рецепторов (n = 32) и с числом A-элементов m, равным 25 и 200.

Каждый A-элемент имел пять входов, из них три входа для возбуждающих и два — для тормозящих связей с сетчаткой. Порог каждого A-элемента выбирался равным единице.

Для обучения перцептрона использовались три различных алгоритма:

— алгоритм I: α-система подкрепления с коррекцией ошибок (6.4.11) в комбинации с алгоритмом случайного поиска опти мальной структуры связей всех A-элементов с сетчаткой (6.4.12), (6.4.13);

— алгоритм II: α-система подкрепления с коррекцией оши бок (6.4.11) в комбинации с алгоритмом случайного поиска оп тимальной структуры с отбором A-элементов (6.4.15);

— алгоритм III: α-система подкрепления с коррекцией оши бок (6.4.11) в комбинации с алгоритмом случайного поиска с обучением.

В первых двух алгоритмах не учитывалась предыстория поиска, поскольку на каждом новом шаге поиска для A-элементов могла равновероятно образоваться любая из возможных структур их связей с сетчаткой. Так, для любого j-го A-элемента

(j = 1, ..., m) число различных возможных структур его соединений может быть в общем случае подсчитано как

$$n_k = \frac{A_n^{l}}{P_{l_1} P_{l_2}}, \quad (6.4.31)$$

где A_n^{l} — число размещений из n элементов по l; P_{l_i} — число перестановок из l_i элементов (i = 1, 2); l_1 (l_2) — число входов A-элемента для возбуждающих (тормозящих) связей; n — размерность сетчатки.

Вероятность образования структуры связей для j-го A-эле-мента может быть представлена в виде n_k-мерного вектора:

$$P_j = (p_1, \ldots, p_{n_k}), \quad (6.4.32)$$

где $\sum_{i=1}^{n_k} p_i = 1$. При этом в алгоритмах I и II все координаты вектора p_i равны $1/n_k$. В результате обучения на N-м шаге поиска увеличивается вероятность p_i образования j-м A-элементом такой i-й структуры, для которой выполняется условие

$$Q_j(S_{jN}, n_i) \le q. \quad (6.4.33)$$

Здесь $Q_j(S_{jN}, n_i)$ — значение минимизируемого показателя качества (например, числа неправильных реакций j-го A-элемента на объекты обучающей выборки) для i-й структуры, определяемой вектор-столбцом S_{iN}; q — некоторый заданный порог. Вероятности образования остальных структур n_t ($t \neq i$) уменьшаются за счет нормирования вероятностей. Этот алгоритм обучения до нормирования может быть записан в виде

$$p_i^{N+1} = \begin{cases} p_i^N + \delta, & \text{если } Q_j(S_{jN}, n_i) \le q; \\ p_i^N, & \text{если } Q_j(S_{jN}, n_i) > q, \end{cases}$$

где величина δ определяет интенсивность обучения.

Все 43 объекта были включены в обучающую последовательность, на которой исследовалась возможность ее безошибочной классификации алгоритмами I—III. При этом по некоторым операциям (R-элементам) не удалось добиться безошибочного распознавания (это были R_4, R_7, R_{11} и R_{14}), несмотря на значительные затраты машинного времени. Эти результаты свидетельствовали о пересечении классов в выбранном пространстве.

Для проверки работоспособности и сравнения предложенных алгоритмов из общего количества объектов случайным образом выбиралось 30 для составления обучающий выборки, а оставшиеся 13 объектов использовались для экзамена. Таким образом, были выбраны три различные обучающие выборки и для каждой из них было проведено обучение по всем трем алгоритмам (I— III). Осредненные результаты экзамена приведены в табл. 6.4.4.

Таблица 6.4.4

Номер операции (*R*-элемента)	Доля (%) правильных ответов на		
	Алгоритм I (*m*=200)	Алгоритм II (*m*=200)	Алгоритм III (*m*=25)
1	92,3	89,4	88,5
2	96.1	87.2	84.6
3	84,6	87,2	69,2
4	49.9	56.4	50.0
5	92.3	87.2	69.2
6	76.9	72.4	69.2
7	60.4	64.1	69.2
8	80.6	82.0	88.5
9	84.6	84.6	88.5
10	84.6	79.4	80.8
11	57.7	59.0	57.7
12	84.6	84.6	84.6
13	96.1	94.9	94.9
14	38,5	62,2	53,8

Полученные результаты показывают примерно одинаковую эффективность предложенных алгоритмов с точки зрения надежности классификации. Однако число шагов поиска для алгоритма II значительно меньше, чем для остальных алгоритмов, что позволяет при его использовании сократить затраты машинного времени. На рис. 6.4.7 показана динамика обучения перцептрона. Сравнительно низкая надежность классификации, полученная по

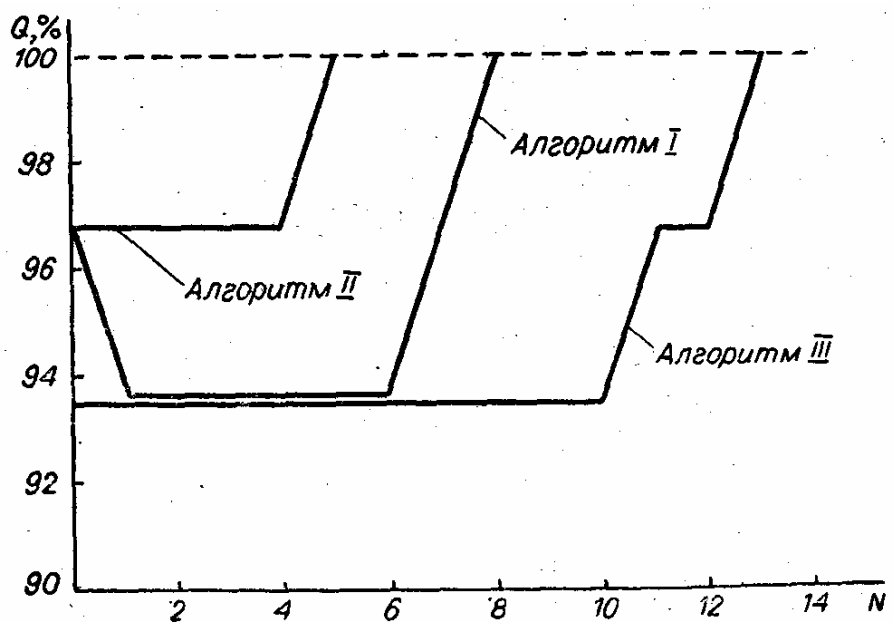


Рис. 6.4.7. Динамика обучения перцептрона с 25 *A*-элементами для R_3 («пробивка отверстия»). Обучающая выборка *L*=32.

операциям 4, 7, 11, 14, подтвердила предположение о пересечении классов в выбранном пространстве, что и было установлено последующим анализом описаний объектов.

Из полученных результатов следует принципиальная возможность использования методов распознавания образов для решения трудноформализуемых задач технологического проектирования.

6.4.5.3. Прогнозирование активности химических соединений по их структурным формулам

Целенаправленный синтез органических соединений, обладающих заранее заданными свойствами — например, реакционной способностью, фармакологической активностью, — всегда являлся важнейшей задачей органической химии. Углубление знаний о квантовохимических механизмах, отвечающих за проявление того или иного свойства химического вещества, позволяет химикам более успешно решать эту задачу.

Тем не менее в настоящее время взаимосвязь между структурой и свойствами вещества устанавливается в основном чисто эмпирическим путем. Это приводит к огромным затратам времени на синтез и изучение таких веществ, которые затем отбрасываются как неудачные, пока не будет получено вещество с требуемыми полезными свойствами.

Для уменьшения этих затрат стремятся найти корреляции между факторами, связанными со структурой вещества, и его свойствами [93]. Однако подобные попытки пока не принесли серьезных успехов, что объясняется, по-видимому, чрезвычайно сложным характером зависимости «структура—свойство».

В связи с этим представляет значительный интерес применение для предсказания свойств химических веществ методов распознавания образов — в частности, использование описанных выше перцептронных алгоритмов.

Задача прогнозирования активности химических соединений может быть поставлена как задача их классификации по структуре молекулы. Такая задача классификации химических соединений по их свойствам в пределах одного гомологического ряда [59, 158] исследовалась на примере классификации фармакологической активности по структуре молекулы ряда алкил- и алкоксиалкилзамещенных 1,3-диоксанов [28]. Пусть имеется множество химических соединений, представляемых структурными формулами, показанными на рис. 6.4.8, *а*. Молекулы этих соединений имеют неизменное ядро N и отличаются друг от друга только видом радикалов $R_1,...,R_k$, каждый из которых может быть представлен определенным сочетанием из t заранее известных простейших структурных компонентов.

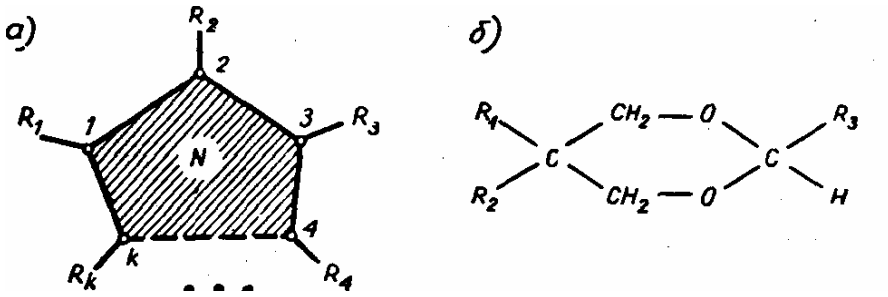


Рис. 6.4.8., Структурная формула химического соединения: *а* — обобщенная структура, *б* — структура тризамещенных 1,3-диоксанов.

В этом случае каждому соединению может быть сопоставлен *n*-мерный вектор двоичных признаков *X,* координаты которого определяют составные структурные компоненты радикала, а сам вектор учитывает их структурные связи. Каждый вектор *X* может быть отнесен к одному из двух классов в зависимости от физических или химических свойств соединения. Иначе говоря, решается задача дихотомии.

Рассмотрим конкретный класс соединений, структура которого показана на рис. 6.4.8, *б*. Структурные формулы соединений этого класса отличаются друг от друга только видом радикалов R_1, R_2, R_3 и изомерией. Поэтому задача состояла в отнесении предъявляемого соединения в зависимости от вида радикалов и изомерии диоксанового кольца к одному из двух классов — фармакологически активным или неактивным веществам.

Каждое соединение представлялось 64-мерным вектором двоичных признаков. Структурная формула каждого радикала может быть представлена сочетаниями из пяти простейших структурных компонентов: O, H, CH, CH_2, CH_3. Для кодирования каждого из этих компонентов на сетчатке перцептрона отводилось по три рецептора, а для кодирования *цис-* и *транс*-изомерии — по два. Связи между компонентами специально не кодировались, так как они однозначно определялись видом соседних компонентов. На рис. 6.4.9 приведены примеры кодирования радикалов на сетчатке. Как видно, для кодирования радикала R_1 отведено три строки, а для радикалов R_2 и R_3 — по две строки. Однако в Целях экономии элементов сетчатки строки указанных кодов располагались последовательно.

Для решения этой задачи на ЭВМ моделировался перцептрон с числом ассоциативных элементов, равным 100. Каждый *А-элемент* имел три входа для возбуждающих связей и два — для тормозящих. Из 46 приведенных в работе [29] соединений выбирались три обучающие последовательности с числом объектов,

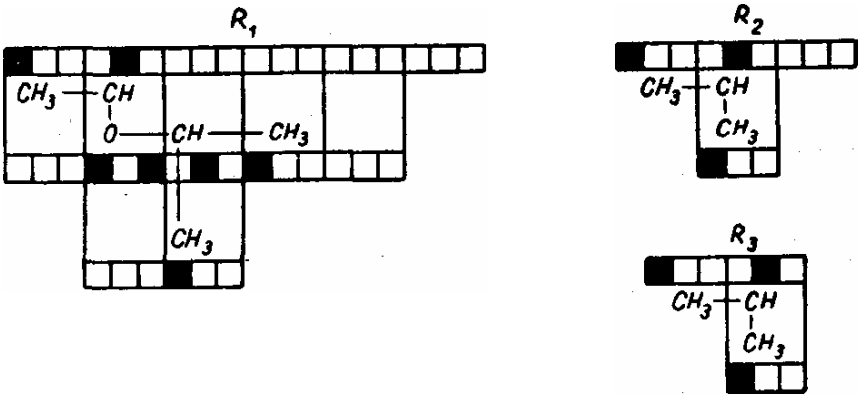


Рис. 6.4.9. Примеры кодирования радикалов на сетчатке перцептрона.

равным 22, 24 и 26. Для каждой последовательности проводи лось десять независимых экспериментов — десять раз случайным образом выбиралась исходная структура. Для каждой структуры проводилось обучение по двум алгоритмам: алгоритм I — α-си стема подкрепления с коррекцией ошибок (6.4.11) и алгоритм II структурной адаптации (6.4.12) — (6.4.13), а оставшаяся часть соединений использовалась только для экзамена. Результаты экзаменов, осредненные по десяти экспериментам, с доверитель ными интервалами, соответствующими вероятности 0,95 для обу чающей последовательности $L = 24$, приведены на рис. 6.4.10. Полученные результаты показали, что использование перцеп трона с адаптивной структурой позволяет достичь надежности классификации, равной 75%. При использовании же обычного перцептрона надежность классификации не превышала 58%, т. е. практически задача не решалась.

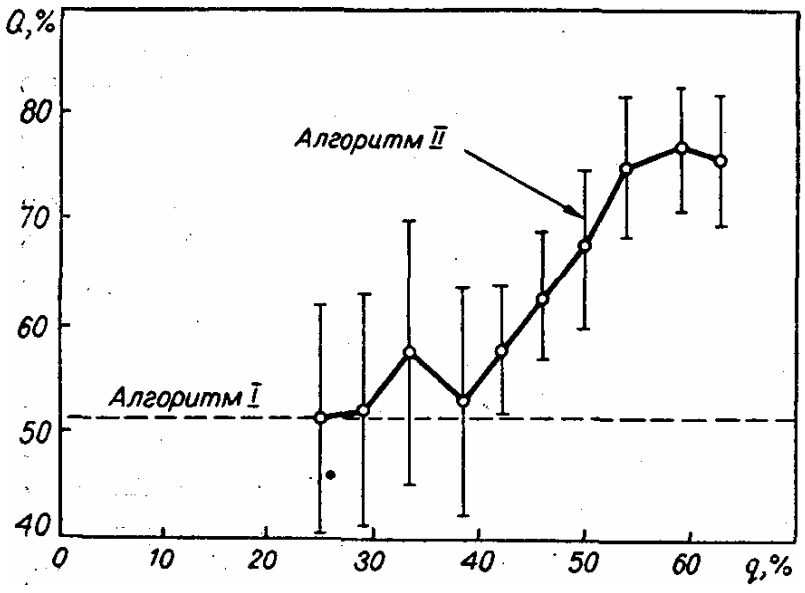


Рис. 6.4.10. Результаты экспериментов по распознаванию активности химических соединений исходя из структурных формул для алгоритмов I и II.

6.4.5.4. Прогнозирование месторождений полезных ископаемых

Возможность использования перцептрона с адаптивной структурой для прогнозирования месторождений полезных ископаемых исследовалась на примере распознавания месторождений и рудопроявлений Атасуйского рудного района [57]. Каждый объект (месторождение или рудопроявление) представлялся 64-мерным вектором определенных признаков; из общего числа 46 объектов 5 объектов составляли класс месторождений, а 41 — класс рудопроявлений. Для решения этой задачи на ЭВМ моделировался трехслойный перцептрон с сетчаткой из 64 рецепторов и 100 *A*-элементами. Каждый *A*-элемент имел три входа для возбуждающих связей и два — для тормозящих, причем исходная структура связей выбиралась случайным образом. Из 46 приведенных в работе [57] объектов случайным образом выбиралось шесть различных обучающих последовательностей с числом объектов 25, 26, 27, 34, 35, 36, для которых проводилось обучение по предложенному алгоритму, а оставшаяся часть составляла экзаменационную последовательность. Результаты экспериментов показали, что надежность классификации достигает 80—85%.

Таким образом, результаты, полученные при решении различных практических задач, показывают, что использование перцептрона с адаптивной структурой позволяет существенно повысить надежность классификации по сравнению с обычным перцептроном [151]. Сложные задачи классификации (например, задача классификации химических соединений по их активности), которые вообще не могут быть решены с помощью обычного перцептрона с ограниченным числом *A*-элементов, весьма успешно решаются после дополнительного введения процедуры адаптации структуры. Кроме того, следует отметить, что предложенный алгоритм адаптации структуры трехслойного перцептрона может быть использован и в многослойных перцептронах, позволяющих решать более сложные задачи (обобщение по подобию и др.), т. е. может рассматриваться как универсальное правило синтеза структуры решающих правил весьма широкого класса. По-видимому, большинство известных решающих правил может быть синтезировано именно таким образом.

§ 6.5. Адаптивный синтез оптимальных факторных планов эксперимента

6.5.1. Постановка задачи

Рассмотрим применение эволюционного адаптивного алгоритма случайного поиска для синтеза оптимальных факторных планов эксперимента [180].

Пусть исследуемое явление зависит от q факторов u_1, ..., u_q и каждый фактор u_i (i = 1, ..., q) имеет N_i варьируемых уровней a_{i1}, ..., a_{iNi}. Тогда факторное пространство $\{U\}$ образуется $N = \prod_q N_i$ возможными точками эксперимента $U^{(j)}$ (j = 1, ...

..., N). Пусть количество планируемых экспериментов равно $m<N$, т. е. план эксперимента O определяется m точками:

$$O = \{U_1, \ldots, U_m\}, \quad (6.5.1)$$

где $U_i \in \{U^{(j)}\}$ (j = 1, ..., N), причем некоторые точки плана могут совпадать. Тогда для точного определения оптимального плана O^* надо произвести перебор из $\binom{m}{N}$ планов, что практически осуществимо лишь при малых m и N. Возникает необходимость в эффективном усечении перебора. Это можно сделать с помощью алгоритма эволюционной адаптации.

Рассмотрим модель объекта, линейную относительно ее параметров:

$$y = \sum_{i=1}^{k} c_i \varphi_i(U), \quad (6.5.2)$$

где $\{\varphi_i(U)\}$ (i = 1, ..., k) — заданная система функций, обычно полиномиальная, т. е. вида

$$\varphi_i(U) = u_1^{\gamma 1(i)} \ldots u_q^{\gamma q(i)}. \quad (6.5.3)$$

Матрица $\|\gamma_i^{(i)}\|$ (j = 1, ..., q; i = 1, ..., k), как видно, определяет выбор системы функций $\{\varphi_i(X)\}$ (i = 1, ..., k), где $\gamma_1^{(i)}$, ..., $\gamma_q^{(i)}$ — целочисленные неотрицательные показатели степени. Критерий эффективности здесь, как и в § 4.4, задается на дисперсионной матрице искомых параметров $C = (c_1, \ldots, c_k)$:

$$K = K(DC), \quad (6.5.4)$$

где DC — дисперсионная матрица (4.4.15) вектора параметров C, зависящая от синтезируемого плана O. Для определенности в качестве критерия был выбран определитель дисперсионной матрицы (обобщенная дисперсия), минимум которого соответствует максимуму определителя информационной матрицы Фишера (4.4.10), т. е. D-оптимальному плану.

6.5.2. Адаптация вероятностных характеристик поиска

Для нахождения оптимального (точнее — квазиоптимального) плана O^*эксперимента используем сначала метод эволюционного случайного поиска, в соответствии с которым вероят-

ность выбора удачных точек плана увеличивается, а вероятность выбора неудачных — уменьшается. На первом этапе планы $U_j^{(1)}$ генерируются исходя из равномерного распределения вероятностей выбора точек плана, т. е. $p_1 = (1/N, \ldots, 1/N)$. Для каждого случайного плана $U_j^{(1)}$ ($j = 1, \ldots, r$) определяется значение максимизируемого показателя D-оптимальности $K_j^{(1)}$).

По результатам r-кратного моделирования определяются

$$K_{\min}^{(1)} = \min_{j=1,\ldots,r} K_j^{(1)} \quad \text{и} \quad K_{\max}^{(1)} = \max_{j=1,\ldots,r} K_j^{(1)} \tag{6.5.5}$$

и запоминаются те планы $U_{\min}^{(1)}$ и $U_{\max}^{(1)}$, при которых получены эти значения показателя. Из точек наилучшего плана делаются систематические попытки улучшить этот план путем «спусков» вдоль тех координат плана, которые позволяют сделать это. Общее число таких возможных спусков на каждый случайный план равно mq. На каждом спуске решается задача

$$K(U_1, \ldots, U_m) \to \max_{u_{ij} \in S_i} \Rightarrow u^*_{ij}, \tag{6.5.6}$$

где $U_j = (u_{1j}, \ldots, u_{ij}, \ldots, u_{qj})$, а S_i — множество уровней i-го фактора, число которых равно N_i.

Если в результате подобных локальных спусков находится такой план $U_{\text{сп}}^{(1)}$, для которого значение критерия оптимальности $K_{\text{СП}}^{(1)}$ больше значения $K_{\max}^{(1)}$, то за оптимальный на данном этапе принимается план $U_{\text{сп}}^{(1)}$. В противном случае за оптимальный принимается исходный план $U_{\max}^{(1)}$, соответствующий значению критерия оптимальности $K_{\max}^{(1)}$.

Далее перестраиваются вероятности выбора точек, т. е. алгоритм поиска адаптируется. Адаптация сводится к изменению вектора вероятностей P на каждом этапе поиска так, чтобы точки плана $U_{\max}$ увеличили свою вероятность, а точки плана U — уменьшили. Коррекция вектора вероятностей на k-м этапе адаптации производится по рекуррентной формуле: для точек плана $U_{\min}$

$$\left.\begin{array}{ll} p_j^{(k)} - h', & \text{если } p_j^{(k)} - h' \geqslant p^- \\ p_j^{(k)} - h' < p^- & \end{array}\right\} \text{при } U_j \in U_{\min}$$

с последующей нормировкой:

$$\left\{\begin{array}{ll} & \\ p_j^k & \text{при } U_j \notin U_{\min} \end{array}\right.$$

$$p_j^{(k+1)} = \frac{q_j^{(k+1)}}{\sum_{j=1}^{N} q_j^{(k+1)}};$$ (6.5.7) (6.5.8)

для точек плана $\tilde{U}_{\max}$

$$q_j^{(k+1)} = \left\{ \begin{array}{lll} p_j^{(k)} + h^+, & \text{если} & p_j^{(k)} + h^+ \leqslant p^+ \\ p^+, & \text{если} & p_j^{(k)} + h^+ > p^+ \\ p_j^{(k)} & \text{при} & U_j \notin \tilde{U}_{\max} \end{array} \right\} \text{при } U_j \in \tilde{U}_{\max} \quad (6.5.9)$$

с такой же последующей нормировкой (6.5.8).

После адаптации вероятностей переходят к следующему, (k+1)-му этапу выбора точек лучшего и худшего плана в соответствии с полученными вероятностями, и т. д. [278].

За оптимальный принимается план $\tilde{U}^*$, наилучший по всем циклам поиска.

Рассмотрим примеры моделирования описанного алгоритма при различных значениях параметров.

Пример 1. $q = 2$; $N_i = 3$ (i = 1, 2), т, е. имеются два фактора u_1 и u_2, варьируемых на трех уровнях: u_1, $u_2 \in \{-1; 0; 1\}$. Возможные точки эксперимента и их нумерация показаны на рис. 6.5.1 (N=9). Структура функции (модели исследуемого явления) имеет вид

$$y(U) = c_1 + c_2u_1 + c_3u_2 + c_4u_1u_2. \quad (6.5.10)$$

Здесь использована система функций

$$\varphi_1 = 1; \quad \varphi_2 = u_1; \quad \varphi_3 = u_2; \quad \varphi_4 = u_1u_2. \quad (6.5.11)$$

Рассмотрим сначала некоторые общие свойства информационной матрицы с системой функций (6.5.11). Обозначим через Φ_i информационную матрицу Фишера для одного эксперимента в точке U_i (i = 1, ..., 9). Учитывая систему функций (6.5.11), получаем

$$\Phi_1 = \begin{Vmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{Vmatrix} \quad \Phi_2 = \begin{Vmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{Vmatrix} \quad \Phi_3 = \begin{Vmatrix} 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 \end{Vmatrix}$$

$$\Phi_4 = \begin{Vmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{Vmatrix} \quad \Phi_5 = \begin{Vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{Vmatrix} \quad \Phi_6 = \begin{Vmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{Vmatrix} \quad (6.5.12)$$

$$\Phi_7 = \begin{Vmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{Vmatrix} \qquad \Phi_8 = \begin{Vmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{Vmatrix}$$

$$\Phi_9 = \begin{Vmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{Vmatrix}$$

Матрица Фишера для плана $\bar{U} = \{Ui_1, ..., Ui_m\}$, где i_j — номера точек, обозначенные на рис. 6.5.1, образуется простой суммой:

$$\Phi(\bar{U}) = \sum_{i=i_1}^{i_m} \Phi_i. \quad (6.5.13)$$

Если количество точек (экспериментов) в плане $m<4$, то значение критерия D-оптимальности плана, очевидно, всегда равно нулю.

В случае $m=4$ наилучшим является план $\bar{U}^* = \{U_1, U_3, U_7, U_9\}$, образуемый угловыми точками. Ему соответствует критерий, равный $|\Phi_1 + \Phi_3 + \Phi_7 + \Phi_9| = 256$. Другие значения критерия неоптимального плана равны 0, 4, 16 и 64.

Если план имеет $m = 5$, то оптимальное расположение также соответствует угловым точкам: $\bar{U}^* = \{U_1, U_3, U_7, U_9, U_i\}$ ($i = 1, 3, 7, 9$), Значение критерия для этого плана равно 512.

Оптимальный план с $m = 6$ имеет вид $\bar{U}^* = \{U_1, U_3, U_7, U_9, U_i, U_j\}$ ($i \neq j = 1, 3, 7, 9$) и значение критерия $K\{\bar{U}^*\} = 1024$.

Совершенно аналогично строятся оптимальные планы, состоящие из 7 и 8 точек; они содержат только угловые и не более чем двухкратные точки. Значения критерия для них соответственно равны 2048 и 4096.

Теперь приведем результаты моделирования описанного алгоритма синтеза для различных m и при следующих значениях параметров алгоритма: $h^- = h^+ =$ 0,1; 0,05; 0,01. Результаты экспериментов представ-

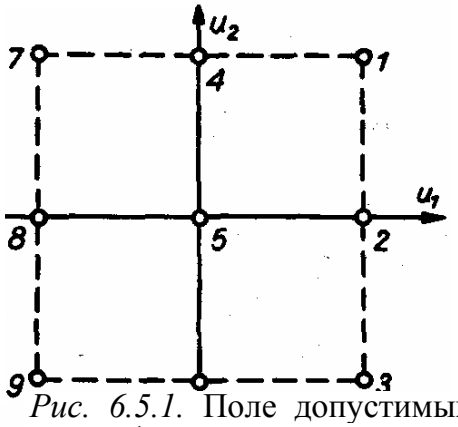


Рис. 6.5.1. Поле допустимых точек факторного плана эксперимента для q=2.

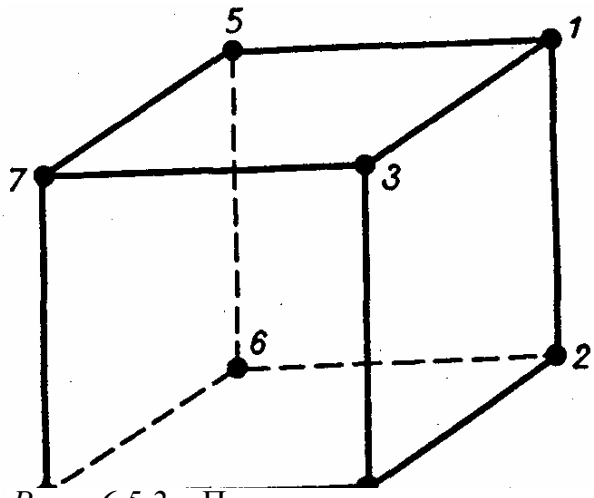


Рис. 6.5.2. Поле допустимых точек факторного плана эксперимента для q=3 и двух уровней.

лены в табл. 6.5.1, где показано число этапов R, необходимое для отыскания оптимального плана. Из таблицы видно, что он находится за сравнительно небольшое количество этапов (R = 3—18), причем уменьшение параметра h незначительно влияет на скорость нахождения оптимального плана. Финальные вероятности в экспериментах хорошо соответствовали оптимальным.

Пример 2. q=3; структура искомой модели линейна:

$$y(U) = c_1 + c_2u_1 + c_2u_2 + c_4u_3; \qquad (6.5.14)$$

каждый фактор u_i (i = l, 2, 3) имеет два уровня, обозначенных 1 и -1 соответственно. Факторное пространство и нумерация точек этого пространства показаны на рис. 6.5.2. Количество точек в плане m=4. Известно, что в этом случае имеются два D-оптимальных плана, состоящих из точек U_1, U_4, U_6, U_7 и U_2, U_3, U_5, U_8. Значение критерия для оптимального плана равно 256. Эксперимент был проведен для следующих значений параметров: r = 100, $h^+ = h^-$ =0,05. Результаты эксперимента представлены в табл. 6.5.2, из которой видно, что уже на первом цикле поиска был найден оптимальный план, а на восьмом вероятности выбора точек плана стали близкими к оптимальным: p_i* = l/4 (i = = 1, 4, 6, 7), p^*_i=0 (i = 2, 3, 5, 8).

Пример 3. q=3; структура искомой функции такова:

$$y(U) = c_1 + c_2u_1 + c_3u_2 + c_4u_3 + c_5{u_3}^2 + c_6{u_3}^3 + c_7{u_3}^4 + c_8{u_3}^5, \qquad (6.5.15)$$

причем факторы u_1 и u_2 имеют два уровня: 0 и 1, а фактор u_3 —

Таблица 6.5.1

Число точек плана m	Рекорд k_{max}	Число этапов R при параметрах поиска			
		h =h$^+$ =0,l; 15	h =h$^+$ =0,05; r=15	h =h$^+$ =0,01; r = r=15	h =h$^+$ =0,05; r=10
4	256	5	9	12	6
5	512	3	3	10	3
6	1024	4	10	18	11
7	2048	7	7	4	11
8	4098	5	8	15	6

Таблица 6.5.2

R	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	Φ_{max}
1	0,175	0,125	0,125	0,175	0,075	0,125	0,175	0,175	256
2	0.225	0.125	0.125	0.075	0.075	0.125	0.225	0.025	256
3	0.225	0.125	0.125	0.125	0.075	0.175	0.120	0.250	256
4	0.125	0.125	0.125	0.175	0.025	0.225	0.175	0.025	256
5	0.125	0.075	0.125	0.225	0.025	0.175	0.225	0.025	256
6	0.171	0.073	0.122	0.220	0.024	0.220	0.170	0	256
7	0.171	0.073	0.122	0.270	0.024	0.120	0.220	0	256
8	0.171	0.073	0.122	0.270	0.024	0.170	0.270	0	256
9	0.171	0.073	0.122	0.270	0.024	0.170	0.270	0	64
10	0.215	0.071	0.021	0.214	0	0.165	0.314	0	256
11	0.215	0.071	0.021	0.214	0	0.215	0.264	0	256
12	0.265	0.071	0.021	0.264	0	0.165	0.214	0	256
13	0.315	0.021	0.021	0.214	0	0.165	0.264	0	256
14	0.215	0.021	0.021	0.264	0	0.215	0.264	0	256
15	0.115	0.021	0.021	0.314	0	0.265	0.264	0	256
16	0.065	0.021	0.021	0.364	0	0.314	0.213	0	256
17	0.065	0.021	0.021	0.364	0	0.265	0.262	0	256
18	0.115	0.021	0.021	0.314	0	0.265	0.263	0	256
19	0.165	0.021	0.021	0.264	0	0.265	0.263	0	256
20	0,215	0,021	0,021	0,214	0	0,215	0,313	0	256

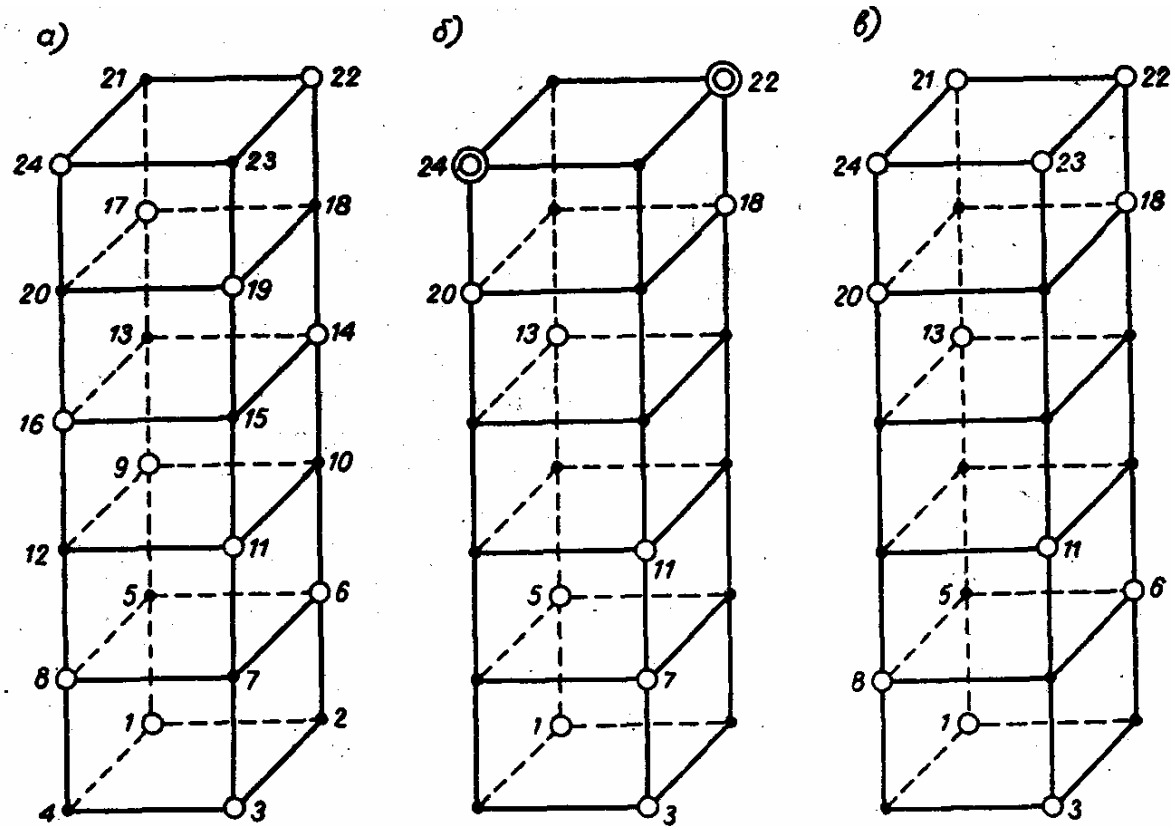


Рис. 6.5.3. Факторные планы (m=12; N_1=N_2=2; N_3=5).
а — план, полученный в работе [65]; *б*, *в* — планы, синтезированные с помощью эволюционной адаптации: *б* — в эксперименте № 2, *в* — в эксперименте № 11 (см. табл. 6.5.3)

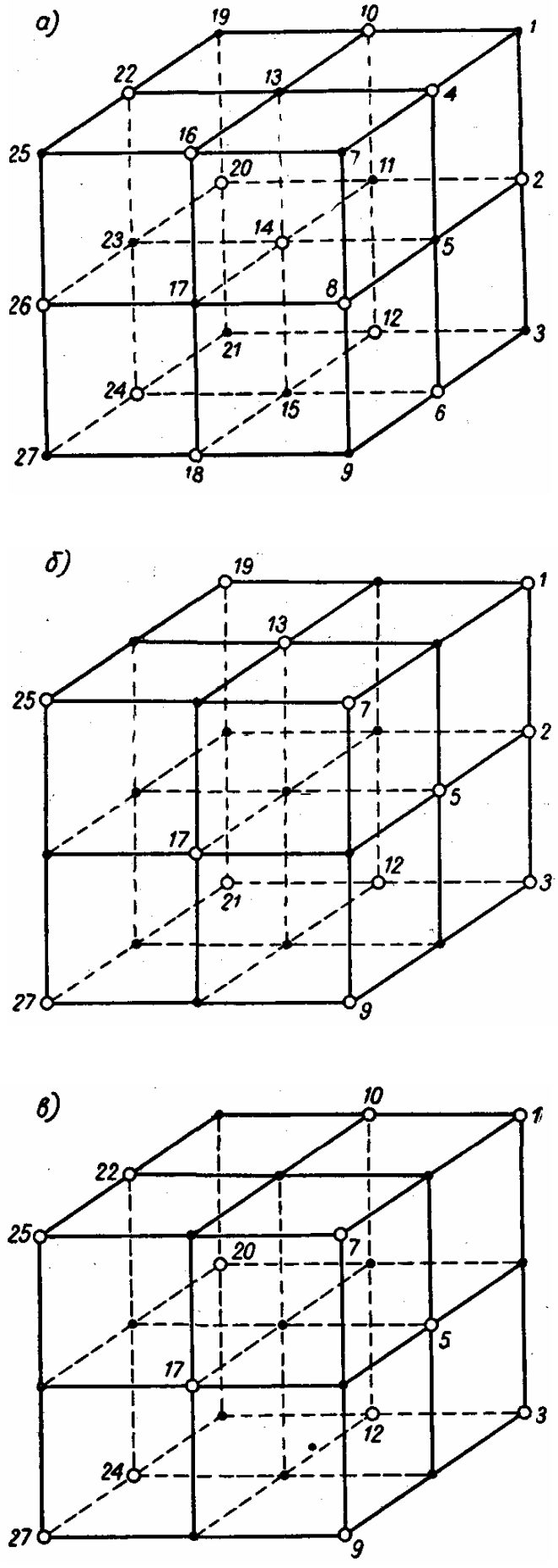


Рис. 6.5.4. Факторные планы (m = = 13; N_1=N_2=N_3=3). *а* — план, полученный в работе [65], *б, в* — планы, синтезированные с помощью эволюционной адаптации.

шесть уровней: 0, 1, 2, 3, 4, 5. Факторное пространство и нумерация его точек показаны на рис. 6.5.3. Количество точек в плане m=12. В этом случае всего имеется

$$\binom{12}{24} = 2\,704\,156$$ планов, из

которых по [34] оптимальными являются лишь два: один состоит из точек U_i (i = 1, 3, 6, 8, 9, 11, 14, 16, 17, 19, 22 и 24), а другой — из остальных точек. Значение нормированного D-критерия для полученного оптимального плана равно $102{,}4 \cdot 10^3$. Этот план, показанный на рис. 6.5.3, а светлыми кружочками, отличается своей симметрией.

Другие квазиоптимальные планы были получены в работе [34] методом, предложенным ранее [261]. Однако описанный выше эволюционный алгоритм адаптации позволил получить планы с большими значениями критерия K [198] (табл. 6.5.3). В среднем лучший случайный план получается на пятом этапе, т. е. лучшие планы встречаются примерно равновероятно. Спуск значительно улучшает план — примерно в два раза по D-критерию (из двенадцати планов только один (№ 6) не удалось улучшить).

Из табл. 6.5.3 видно, что были получены два плана (№2 и № 11) с максимальным значением критерия. Они показаны на рис. 6.5.3, *б, в,* где двойными кружочками обозначены точки повторных экспериментов. Вообще повторяемость экспериментов является характерной чертой полученных планов. Только один из полученных планов (№ 11) не имеет повторяющихся точек.

Пример 4. q=3 (N_1=N_2=N_3=3); искомая функция обладает структурой полной квадратичной формы:

$$y(U) = c_1 + c_2u_1 + c_3u_2 + c_4u_3 + c_5u_1^2 + c_6u_2^2 + c_7u_3^2 + c_8u_1u_2 + c_9u_1u_3 + c_{10}u_2u_3. \quad (6.5.16)$$

Все факторы u_i (i = l, 2, 3) имеют три уровня: 1, 0, —1. Факторное пространство и нумерация его точек показаны на рис.

6.5.4. Количество точек в плане m=13. Всего существует $\binom{13}{27}$= = 20 058 300 планов. Квази-D-оптимальный план для этого случая построен в работе [65] (рис. 6.5.4, *а*).

Предложенным методом были синтезированы планы, представленные в табл. 6.5.4. Из данных таблицы видно, что только два случайных плана (№ 1 и № 10) оказались хуже, чем № 18. Все остальные планы были лучше.

На рис. 6.5.4, *б, в* показаны два наилучших плана (№ 9 и № 16). Здесь заняты угловые точки, что, естественно, не может не улучшить критерий эффективности плана по сравнению с полученным в работе [65].

Любопытно, что, в противоположность примеру 3, не было обнаружено ни одного плана с повторяющимися точками.

Все полученные планы несимметричны, что нельзя считать преимуществом, так как при этом теряется ортогональность, обес печивающая независимость оценок параметров модели (см. рис. 6.5.4, а).

Таким образом, результаты проведенных экспериментальных исследований показывают, что предложенный алгоритм эволюционной адаптации плана эффективно решает задачу построения оптимальных факторных планов эксперимента. Особенно перспективно применение этого алгоритма в случае, когда значения параметров плана m и N велики.

6.5.3. Адаптивный синтез планов эксперимента методами случайного поиска с пересчетом и спуском

Рассмотренные в предыдущем подразделе алгоритмы синтеза оптимальных планов эксперимента мало чем отличались от случайного перебора. Поэтому естественно обратиться к поисковому

№ эксперимента	Объем выборки случайных планов r	Рекорд при $R = 10$ $K_{max}(^1)/10^3$	Номер этапа, на котором был получен рекорд $R(^1)$	Рекорд после спуска $K_{max}(^2)/10^3$	Номер этапа, на котором был получен рекорд после спуска $R(^2)$
1	50	81,9	3	178,6	1
2	50	18.8	6	220.1	6
3	50	85.1	2	177.3	1
4	100	74.2	1	163.8	1
5	100	89.6	5	159.4	8
6	100	168.3	8	168.3	8
7	50	84.1	9	178.6	1
8	50	85.1	7	119.0	1
9	50	68.5	7	138.2	5
10	100	113.3	10	163.8	1
11	100	111.4	1	220.1	10
12	100	106,9	8	163,8	1
13	По данным [65]			102,4	

№ эксперимента	Объем выборки случайных планов г	Рекорд при $R = 10$ $K_{max}(^1)/10^3$	Номер этапа, на котором был получен рекорд $R^{(1)}$	Рекорд после спуска $K_{max}(^2)/10^3$	Номер этапа, на котором был получен рекорд после спуска $R(^2)$
1	50	5,5	9	11,5	1
2	50	7.5	1	27.6	1
3	50	9.9	9	18.1	1
4	75	14.9	1	14.9	1
5	75	15.7	8	27.0	8
б	75	9.2	1	21.8	1
7	100	10.2	1	10.2	1
8	100	22.8	1	22.8	1
9	100	10.3	8	31.9	1
10	50	5.0	1	11.5	1
11	50	9.1	3	22.4	3
12	50	10.8	1	24.3	2
13	75	14.9	1	30.2	4
14	75	7.8	3	24.3	4
15	100	12.4	9	26.9	4
16	100	17.4	1	33.7	4
17	100	14,7	1	14,7	1
18	По каталогу [3]			6,09	

* Обозначения те же, что в табл. 6.5.3.

Таблица 6.5.3

Кол-во вычислений критерия для получения плана	Точки плана $\tilde{U}^*$											
338	1,	3,	5,	7,	11,	13,	16,	19,	21,	22,	22,	24
2028	1.	3.	5.	7.	11.	13.	18.	20.	22.	22.	24.	24
338	1.	з.	6.	8.	9.	15.	17.	19.	21.	21.	22.	24
388	1,	з,	6,	7,	12,	14,	17,	17,	21,	21,	22,	24
3104	3.	5.	6.	11.	14.	19.	20.	21.	21.	22.	22.	24
3104	2.	4.	5.	7.	9.	14.	15.	17.	21.	,22.	22.	24
388	1.	3.	5.	7.	11.	13.	16.	19.	21.	22.	22.	24
388	2.	4.	5.	7.	10.	12.	14.	17.	19.	21.	21.	24
826	1.	3.	6.	8.	11.	13.	13.	19.	22.	23.	23.	24
388	1.	3.	6.	7.	12.	14.	17.	19.	21.	21.	22.	24
1576	1	3	6	8	11	13	18	20	21	22	23	24
388	1,	3,	5,	8,	11,	13,	18,	20,	21,	21,	22,	23
	1,	3,	6,	8,	9,	11,	14,	16,	17,	19,	22,	24

*Таблица 6.5.4**

Кол-во вычислений критерия для получения плана	Точки плава 0^*											
341	1	2,	4,	7,	9,	10,	12,	14,	16,	19,	21,	22,	27
341	1	3.	5.	7.	9.	11.	12.	13.	17.	19.	24.	25.	27
341	1	3.	5.	7.	9.	11.	13.	18.	21.	22.	24.	25.	27
366	1	3.	4.	5.	7.	9.	10.	11.	13.	20.	21.	25.	27
1182	1	3.	4.	7.	9.	10.	14.	16.	20.	21.	22.	26.	27
366	1.	2.	3.	5.	7.	9.	10.	12.	17.	20.	22.	25.	27
391	1	2,	3,	6,	7,	9,	10,	13,	20,	21,	23,	25.	27
391	1	3,	5,	7,	9,	10,	12,	14,	20,	22,	24,	26,	27
391	1	3.	5.	7.	9.	10.	12.	17.	20.	22.	24.	25.	27
341	1.	2.	4.	7.	9.	10.	12.	14.	16.	19.	21.	22.	27
1023	2	3.	4.	6.	7.	9.	10.	16.	19.	21.	23.	25.	27
682	1.	3.	6.	7.	8.	9.	11.	14.	18.	19.	21.	25.	27
146	1.	3.	5.	6.	7.	9.	11.	12.	13.	19.	21:	25.	27
1464	1	3.	6.	7.	8.	9.	13.	14.	18.	19.	21.	,25.	27
1564	1.	з.	5.	7.	9.	10.	12.	13.	17.	20.	21.	25.	27
1564	1.	2.	3.	5.	7.	9.	12.	13.	17.	19.	21.	25.	27
391	1	3,	6,	7,	8,	9,	12,	14,	16,	18,	19,	24,	26
	2,	4,	6,	8,	10,	12,	14,	16,	18,	20,	22,	24,	26

подходу. С этой целью введем аналоги процедур случайного поиска с пересчетом и покоординатного спуска.

Случайный поиск с пересчетом в данном случае будет связан со случайным изменением $l \le m$ случайных точек плана [17], т. е. с реализацией случайного преобразования

$$O_N \xrightarrow{\xi} O_N^{(1)},$$

где планы O_N и $O_N^{(1)}$ имеют лишь $m-l$ совпадающих точек (возможно, но маловероятно случайное совпадение и большего числа точек планов). Если полученный план $O_N^{(1)}$ окажется лучше O_N по заданному критерию, т. е.

$$K(O_N) < K(O_N^{(1)}), \tag{6.5.17}$$

то реализуется рабочий шаг

$$O_{N+1} = O_N^{(1)}. \tag{6.5.18}$$

В противном случае делается следующий случайный шаг

$$O_N \xrightarrow{\xi} O_N^{(2)}, \tag{6.5.19}$$

и т. д.

Алгоритм поиска имеет такой вид:

1. Выбрать равновероятно l случайных точек плана O_N и исключить их.

2. Добавить к полученному усеченному плану l равновероятно выбранных случайных точек плана из N допустимых.

3. Если полученный план не окажется лучше O_N, то вернуться к O_N (это и есть «возврат») и перейти к п. 1 ($i \to i+1$). Если $O_N^{(1)} \succ O_N$, то $O_{N+1} = O_N^{(1)}$ и следует п. 1.

Как видно, при $l=m$ этот алгоритм совпадает с изложенным в подразделе 6.5.1 (но без перестройки вероятностей выбора случайных точек плана).

Для эффективной работы данного алгоритма необходимо, чтобы выполнялось следующее требование: незначительное изменение плана должно приводить к незначительному изменению критерия, определенного на этом плане. При малых l это обычно выполняется, что и обеспечивает эффективность описанного алгоритма.

Алгоритм покоординатного спуска предполагает последовательное изменение (сканирование) координат каждой точки плана, т. е. решение следующей задачи:

$$K(U_N) \to \min_{u_{ij} \in \{a_{i1}, \ldots, a_{iN_i^*}\}}$$

(6.5.20)

$(i = 1, \ldots, q; j = 1, \ldots, m)$,

где j-я точка плана U_N

$$U_j = (u_{1j}, ..., u_{qj}) \qquad (6.5.21)$$

варьируется покоординатно, т. е. задачу (6.5.20) надо решать для всех $i = 1, ..., q$. Решение осуществляется прямым перебором,

для чего достаточно сделать $m \prod_{i=1}^{q} (N_i - 1)$ вычислений критерия K.

При обнаружении плана U'_N, лучшего U_N, немедленно производится перестройка

$$U_{N+1} = U'_N \qquad (6.5.22)$$

и спуск продолжается.

Для исключения влияния порядка перебора уровней (a_{il}, ..., a_{iNi}). координат ($i = 1, ..., q$) и точек плана ($j = 1, ..., m$) необходимо ввести рандомизацию, т. е. последующие уровень, координату и точку выбрать случайно и равновероятно (исключая при этом уже использованные варианты).

Можно рассмотреть полный спуск, при котором после последнего улучшения плана делается полный перебор всех вариантов покоординатной деформации плана. При неулучшении плана в этом случае спуск считается полным и законченным.

Усеченный спуск ограничивается получением первого плана лучше исходного. Его длительность ограничена сверху полным перебором всех покоординатных деформаций исходного плана.

Случайное усечение спуска ограничивается случайно выбранным числом точек плана, варьируемых координат и уровней.

Описанные алгоритмы поиска с возвратом и спуском можно объединять в разных сочетаниях, что позволяет построить различные алгоритмы синтеза оптимальных факторных планов эксперимента [180].

Алгоритмы эволюционной адаптации, описанные в этой главе, по сути своей моделируют эволюцию, а формально являются модификациями случайного перебора. Именно это последнее обстоятельство накладывает существенные ограничения, так как всякий перебор, даже очень усеченный, связан с экспоненциальным ростом затрат. Преодоление указанной трудности дало бы возможность решать этим методом очень сложные задачи адаптации.

Заключение

Адаптация в живой природе является основным механизмом, благодаря которому появились, существуют и прогрессируют живые организмы. Во «второй природе», созданной человеком,— в частности, в науке и технике — адаптация пока не имеет столь важного значения. В подавляющем большинстве случаев ее реализует человек, причем неформальным образом. Такой процесс часто называют настройкой, усовершенствованием, модификацией, реконструкцией и т. д. Формализация этих процессов и порождает адаптацию.

В данной книге рассмотрена лишь одна ветвь мощного древа адаптации, которое сейчас бурно развивается благодаря насущным потребностям науки, техники и народного хозяйства. Эти потребности связаны прежде всего с задачей повышения эффективности функционирования существующих систем, которая и возлагается на алгоритмы адаптации, реализуемые схемно в специализированных устройствах (экстремальных регуляторах и оптимизаторах) или программно в универсальных управляющих ЭВМ.

Однако современная тенденция развития методов и средств адаптации показывает, что она начинает играть новую роль. Эта роль связана с появлением таких объектов, которые без адаптации вообще не могут существовать. И только наличие адаптации позволяет объекту выполнять возлагаемые на него функции.

Проиллюстрируем сказанное на одной задаче — автоматизации управления современным аэропортом. Известно, что для ее решения надо иметь вычислительные мощности не менее 10^{14} (сто тысяч миллиардов!) операций в секунду. Самые мощные современные вычислительные системы — сети ЭВМ — могут обеспечить едва сотую долю процента требуемой мощности. Современные ЭВМ, вычислительные системы и сети, опирающиеся на фоннеймановскую концепцию, не смогут решить эту задачу ни сегодня, ни завтра. Дело в том, что возможности таких вычислительных средств ограниченны в принципе, и ограничение связано прежде всего со стабильностью их структуры. Адапта-

ция этих вычислительных средств ограничена жесткими рамками имеющейся структуры, системы команд, типом памяти и т. д. Адаптироваться здесь можно только в рамках заданной структуры, т. е. только варьируя программу. Однако варьированием программы нельзя изменить саму машину. Безусловно, адаптация программ и программного обеспечения ЭВМ, систем и сетей ЭВМ позволяет улучшить их свойства значительно, но не в решающей степени. Так, адаптируя программу, можно повысить ее эффективность в полтора—два раза, но не на два—три порядка.

Для того чтобы получить очень сильный эффект от введения адаптации, надо варьировать *структуру* вычислительных средств, т. е. нужно иметь возможность изменять сами схемы, из которых составлена вычислительная техника. Таких возможностей современные ЭВМ не предоставляют и не могут предоставить. В этой структурной «окаменелости» и состоит слабость электронных компьютеров, которая в настоящее время приводит к исчерпанию их возможностей.

Новые задачи требуют новых средств, позволяющих адаптировать структуру компьютера. И такие средства есть — это оптоэлектроника. Сочетание оптических и электрических процессов позволяет получить принципиально новые функции, и элементы, недоступные обычной электронике, и в конечном счете — создать вычислительные машины с чрезвычайно гибкой структурой, которую легко адаптировать, т. е. перестраивать на эффективное решение различных задач.

Эти новые компьютеры структурно очень сильно отличаются от «старых» (фоннеймановских) схем. Так, память здесь распределена по всем элементам машины, а не сосредоточена в одном месте, как в стандартной ЭВМ. Каждый элемент может в широких пределах изменять свою функцию, т. е. адаптировать ее к новым задачам. Элементы такой новой машины должны обладать большой связностью, т. е. возможностью непосредственно обмениваться информацией с большим числом других элементов машины. Возможность глубокой адаптации структуры отличает новый компьютер от «старых» ЭВМ. Как известно, изменяя структуру, можно получить *любые* схемы, т. е. любые вычислительные машины. Именно здесь возникает возможность каждый раз для новой задачи строить новую, адекватную ей машину. Осуществить это можно только методами адаптации структуры связей и функций элементов машины.

Таким образом, адаптация становится решающим фактором коренного изменения направления развития современной вычислительной техники. Без нее просто невозможно решать задачи, выдвигаемые народнохозяйственной практикой.

Список литературы

1. *Абрайтис Л. Б., Шимайтис А. П.* Алгоритм узлов и исследование их эффективности. — В кн.: Материалы науч.-техн. конф. Литовской ССР «Вычислительная техника». Каунас, Каунас. политехн, ин-т, 1971, т. 2, с. 66—76.
2. *Абрайтис Л. Б., Шейнаускас Р. И., Жилевичус В. А.* Автоматизация проектирования ЭВМ. М., Сов. радио, 1978. 268 с.
3. *Авен О. И., Коган Я. И.* Управление вычислительным процессом в ЭВМ. М., Энергия, 1978. 240 с.
4. Адаптация в вычислительных системах. Под общ. ред. Л. А. Растригина. Рига, Зинатне, 1978. 174 с.
5. Адаптация в системах обработки информации. Рига, Зинатне, 1977. 94 с.
6. Адаптация в системах со сложной организацией. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1977. 174 с. (Вопр. кибернетики).
7. Адаптивные автоматические системы. М., Сов. радио, 1972. 184 с.
8. Адаптивные системы. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1974. 256 с. (Вопр. кибернетики).
9. Адаптивные системы. М., Научный совет АН СССР по комплексной проб леме «Кибернетика», 1976. 190 с. (Вопр. кибернетики).
10. Адаптивные системы. Под общ. ред. Л. А. Растригина. Рига, Зинатне. Вып. 1. 1972. 154 с.; вып. 2, 1972. 92 с.; вып. 3. 1973. 116 с.; вып. 4. 1974. 136 с.; вып. 5. 1974. 79 с.
11. Адаптивные системы управления. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1977. 214 с. (Вопр. кибернетики).
12. *Адлер Ю. П., Маркова Е. В., Грановский Ю. В.* Планирование эксперимента при поиске оптимальных условий. М., Наука, 1976. 280 с.
13. *Айзерман М. А., Браверман Э. М., Розоноэр Л. И.* Метод потенциальных функций в теории обучения машин. М., Наука, 1970. 384 с.
14. *Алакоз Г. М., Бахарев А. Т.* Специфика синтеза многопороговых схем. — В кн.: Проектирование и проверка дискретных устройств на интегральных схемах. Рига, Зинатне, 1977, с. 5—12.
15. *Алакоз Г. М., Бахарев А. Т.* Аналитическая оценка случайной аппроксимации многозначных логических функций. — Теория конечных автоматов и ее приложения (Рига), 1977, вып. 9, с. 54—57.
16. Алгоритмы оптимизации проектных решений. Под ред. А. И. Половинкина. М., Энергия, 1976. 264 с.
17. *Антамошкин А. Н.* О решении одной задачи целочисленного програм-

мирования методом случайного поиска. — В кн.: Материалы 5-й науч. конф. по математике и механике. Томск, Изд-во ТГУ, 1975, с. 80—89.

18. *Аркадьев А. Г., Браверман Э. М.* Обучение машины классификации объектов. М., Наука, 1971. 192 с.
19. *Аткинсон Р., Бауэр Г., Кратере Э.* Введение в математическую теорию обучения. М., Мир, 1969. 486 с.
20. *Бартон Г., Салливан Д.* Ошибки и контроль ошибок. — В кн.: Системы передачи данных и сети ЭВМ. М., Мир, 1974, с. 34—44.
21. *Батищев Д. И.* Поисковые методы оптимального проектирования. М., Сов. радио, 1975. 216 с.
22. *Бахарев А. Т.* Оптимизация многопороговых моделей. — Пробл. слу чайного поиска (Рига), 1975, вып. 4, с. 209—214.
23. *Бахарев А. Т.* Оценка сложности многопороговой реализации троичных функций двух переменных. — Теория конечных автоматов и ее прило жения (Рига), 1976, вып. 6, с. 25—31.
24. *Бахарев А. Т.* Структура входных сигналов многозначных пороговых схем. — Автоматика и вычисл. техника, 1976, № 3, с. 22.
25. *Бахарев А. Т., Растригин Л. А,* Поисковый метод синтеза многопороговых элементов. — Автоматика и вычисл. техника, 1967, № 6, с. 18—20.
26. *Бахарев А. Т., Растригин Л. А.* Синтез оптимальных многопороговых эле ментов методом случайного поиска. — В кн.: Проблемы статистической оптимизации. Рига, Зинатне, 1968, с. 119—129.
27. *Бернштейн. Л. С., Селянкин В. В.* О минимальном разрезании графов со взвешенными ребрами. — Электронная техника, сер. 9. АСУ, 1976, вып. 4(20), с. 96—106.
28. *Богатский А. В., Вихляев Ю. И.* и др. Стереоизомерия и фармакологиче ская активность алкил- и алкоксиалкилзамещенных 1,3-диоксанов. — Хим.-фармакол. журн., 1968, т. 2, № 9, с. 3—12.
29. *Вожаков Е. С., Вучков И. Н.* Статистически методи за моделиране и оптимизиране на многофакторни обекти. София, Техника, 1973. 530 с.
30. *Бонгард М. М,* Проблема узнавания. М., Наука, 1967. 320 с.
31. *Брайловский В. Л.* Алгоритм распознавания со многими параметрами и его приложения. — Изв. АН СССР. Техн. кибернетика, 1964, № 2, с. 30—39.
32. *Бреэгунова Н. М., Рипа К.. К.* Последовательный синтез оптимальных планов экспериментов методом адаптивного случайного поиска. — Пробл. случайного поиска (Рига), 1980, вып. 8, с. 254—270.
33. *Брезгунова Н. М., Рипа К. К.* Структура диалога в процессе синтеза оптимального плана эксперимента. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 263—284.
34. *Бродский В. З., Бродский И. Л., Малолеткин Г. Н., Мельников Н. Я.* О каталоге факторных планов эксперимента на ЭВМ. — Вопр. киберне тики', вып. 47. Математико-статистические методы анализа и планирова ния эксперимента. М., Научный совет АН СССР по комплексной проб леме «Кибернетика», 1978, с. 6—24.
35. *Бродский В. З.* Введение в факторное планирование эксперимента. М., Наука, 1976. 223 с.
36. *Букатова И. Л.* Эволюционное моделирование и его приложения. М., Наука, 1979. 230 с.
37. *Бурков В. Н., Гроппен В. О.* Решение задачи о минимальном разрезе в бисвязном орграфе алгоритмами типа ветвей и границ. — Автоматика и телемеханика, 1974, № 9, с. 104—110.

38. *Бутыльский Ю. Г., Брунченко А. В.* Алгоритм разрезания двудольного графа для построения цифровых устройств на основе больших интег ральных схем. — Автоматика и вычисл. техника, 1976, № 4, с. 72—76.
39. *Буш Р., Мостеллер Ф.* Стохастические модели обучаемости. М., Физмат-гиз, 1962. 484 с.
40. *Буш Р., Мостеллер Ф.* Сравнение восьми моделей. — В кн.: Математи ческие методы в социальных науках, М., Прогресс, 1973, с. 295—315.

41. Вазан *М.* Стохастическая аппроксимация. М., Мир, 1972. 296 с.
Вайнцвайг М. Н. Алгоритм обучения распознаванию образов «Кора». — В кн.: Алгоритмы обучения распознаванию образов. М., Сов. радио, 1973, с. 110—116,
43. *Вайсбурд Р. А., Симоненко В. Н., Баранов Ю. М., Алиев Ч. А.* Разра ботка алгоритмов проектирования с помощью методов обучения рас познаванию образов. — В кн.: Материалы II Всесоюз. науч.-техн. конф. «Автоматизация технической подготовки производства в машинострое нии». Минск, 1972, ч. 2, с. 99—100.
44. *Вапник В. Н.* Восстановление зависимостей по эмпирическим данным. М., Наука, 1979. 448 с.
45. *Вапник, В. Н., Червоненкис А. Я.* Теория распознавания образов. М., Наука, 1974. 414с.
46. *Варшавский В. Н.* Коллективное поведение автоматов. М., Наука, 1973. 400 с.
47. *Вентцель Е. С.* Теория вероятностей. М., Физматгиз, 1962. 560 с.
48. *Вучков И. Н., Круг Г. К. D*-оптимальные экспериментальные планы. — Тр. МЭИ, 1969, вып. 68, с. 5—19.
49. *Вучков К. Н., Круг Г. К.* Оптимальное планирование экспериментальных исследований. — Автоматика и телемеханика, 1969, № 11, с. 171—177.
50. *Вучков И. Н., Круг Г. К.* Применение метода непрерывного планирования экспериментов для получения *D*-оптимальных планов. — В кн.: Проб лемы планирования эксперимента. М., Наука, 1969, с. 69—78,
51. *Гагарин Ю. И.* Об эффективности некоторых адаптивных способов пере дачи дискретной информации. — В кн.: Адаптация в системах со слож ной организацией. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1977, с. 154—157. (Вопр. кибернетики).
52. *Гарусин М. И.* Построение многоуровневых вероятностных алгоритмов градиентного типа в задачах бивалентного программирования. — Авто матика и телемеханика, 1978, № 2, с. 62—72.
53. *Гарусин М. И., Каплинский А. И.* О формировании адаптивных алго ритмов оптимизации псевдобулевых функций на основе метода локаль ного улучшения. — Автоматика и телемеханика, 1976, № 9, с. 96—104.
54. *Гарусин М. И., Каплинский А. И., Чернышева Г. Д.* О правилах локаль ного улучшения в вероятностных алгоритмах оптимизации псевдобуле вых функций. — В кн.: Вопросы оптимального программирования в производственных задачах. Воронеж, 1974, с. 63—69.
55. *Гельфанд И. М., Цетлин М. Л.* О некоторых способах управления слож ными движениями. — Успехи мат. наук, 1962, т. 17, вып. 1, с. 3—25.
56. *Глаз А. Б., Гончаров В. А., Растригин Л. А.* Применение перцептронных алгоритмов для классификации технологических маршрутов. — Адап тивные системы (Рига), 1972, вып. 1, с. 37—48.
57. *Глаз А. Б., Давиденко В. В.* О возможности применения перцептрона с адаптивной структурой для решения геологических задач. — Адаптив ные системы (Рига), 1972, вып. 2, с. 76—90.

58. *Глаз А. В., Растригин Л. А.* Оценка вероятности образования оптималь ной структуры перцептрона при ее оптимизации методами случайного поиска. — В кн.: Задачи статистической оптимизации. Рига, Зинатне, 1971, с. 131—142.
59. *Глаз А. Б., Растригин Л. А., Розенблит А. Б.* Адаптация структуры пер цептрона на примере классификации химических соединений. — Автома тика и вычисл. техника, 1972, № 1, с. 37—45.
60. *Глаз А. Б., Растригин Л. А.* Применение перцептрона для разделения объектов на классы по случайным признакам. — Автоматика, 1972, № 4, с. 19—26.
61. *Глаз А. Б., Растригин Л. А.* Оценка вероятности ошибки распознавания при оптимизации структуры перцептрона методами случайного поиска. — Пробл. случайного поиска (Рига), 1972, вып. 1, с. 167—182.
62. *Глаз А. Б., Растригин Л. А.* Решающие правила с адаптивной структу рой. — Изв. АН СССР. Техн. кибернетика, 1974, № 6, с. 192—201.
63. *Глаз А. Б., Гончаров В. А., Растригин Л А.* Применение адаптивных методов для решения задач технологического проектирования. — Управ ляющие системы и машины, 1977, № 2, с. 14—19,
64. *Глаз А. Б., Растригин Л. А.* Трехрядный статистический перцептрон со специальным устройством статистической адаптации структуры. — В кн.: Перцептрон — система распознавания образов. Киев, Наукова думка, 1975, с. 334—385.
65. *Голикова Т. И., Панченко Л. А., Фридман М. З.* Каталог планов вто рого порядка. В 2-х т. Т. 1. М., Изд-во МГУ, 1974. 387 с.
66. *Гориштейн Л. Л.* О разрезании графов. — Изв. АН СССР. Техн. кибер нетика, 1969, № 1, с. 79—85.
67. *Горцев А. М., Назаров А. А., Терпугов А. Ф.* Управление и адаптация в системах массового обслуживания. Томск, Изд-во ТГУ, 1978. 208 с.
68. *Гренина О. В., Миронова Е. П.* Пособие по английскому языку для мате матических факультетов педагогических вузов. М., Высшая школа, 1974. 111 с.
69. *Гринченко С. Н.* Синтез и анализ алгоритмов матричного случайного поиска. Автореф. дис. на соиск. учен. степ. канд. техн. наук. Киев, 1975. 24с.
70. *Гринченко С. Н.* О нейробионических алгоритмах матричного случайного поиска. — Вопр. кибернетики, вып. 33. Проблемы случайного поиска. М., Научный Совет АН СССР по комплексной проблеме «Кибернетика», 1978, с. 70—78.
71. *Гринченко С. Н., Загускин С. Л.* Случайный поиск как адекватный ап парат описания механизмов функционирования нервной клетки. — Вопр. кибернетики, вып. 45. Случайный поиск в задачах оптимизации. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1978, с. 124—134.
72. *Гринченко С. Н., Растригин Л. А.* Алгоритм матричного случайного-поиска. — Пробл. случайного поиска (Рига), 1976, вып. 5, с. 167—184.
73. *Гринченко С. Н., Растригин Л. А.* О матричном случайном поиске. — Автоматика и вычисл. техника, 1976, № 1, с. 48—51.
74. *Гуляев В. В., Катханов М. Н., Суханов Ю. Н.* Адаптивный алгоритм поиска оптимальных параметров сложных систем. — В кн.: V Всесоюз. совещ. по проблемам управления (Москва, 1971). Реф. докл. М., 1971, ч. 2, с. 132—134.
75. *Гупал А. М.* Стохастические методы решения негладких экстремальных задач. Киев, Наукова думка, 1979. 152 с.

76. *Дэвис Д., Барбер Д.* Сети связи для вычислительных машин. М., Мир, 1976. 680 с.
77. *Дертоузос М.* Пороговая логика. М., Мир, 1967. 343 с.
78. *Джандиери И. В., Растригин Л. А.* Параметрическая и структурная оптимизация коллектива алгоритмов распознавания. — Автоматика, 1979, № 6, с. 3—8.
79. *Джандиери И. В., Растригин Л. А.* Об одном способе синтеза алгоритма распознавания. — В кн.: Динамика машин. Оптимизация и адаптация. Горький, ГГУ, 1979, с. 168—179.
80. *Джандиери И. В., Растригин Л. А.* Один подход к оптимизации кол лектива непараметрических алгоритмов распознавания. — Приложения случайного поиска (Кемерово), 1981, вып. 1 (в печати).
81. *Джандиери И. В., Растригин Л. А.* Об определении оптимального разно образия коллектива и гибрида непараметрических решающих правил ме тодом случайного поиска. — Приложения случайного поиска (Кемерово), 1982, вып. 2 (в печати).
82. *Джандиери И. В., Растригин Л. А., Эренштейн Р. Х.* Один подход к оптимизации коллектива решающих правил: — В кн.: Структурная адап тация сложных систем управления. Воронеж, ВПИ, 1977, с. 54—63.
83. *Добровольская М. К.* Исследование методов распознавания конфигурации штампованных деталей при автоматизированном проектировании технологии и штампов холодной листовой штамповки. Автореф. дис. на соиск. учен. степ. канд. техн. наук. Свердловск, 1972. 17 с.
84. *Дрожжин Н. И., Клыков С. П., Кудрин В. Г., Селюгин Б. Е.* Адаптивная система поисковой оптимизации. — В кн.: Моделирование и оптимиза-.ция в условиях системы автоматизированного проектирования. Материалы респ. семинара в Ваймела 27.06—2.07 1977. Таллин, НИИ Таллин. электротехн. з-да, 1977, с. 77—83.
85. *Друзета А., Вранешич З., Седра А.* Применение многопороговых элементов для реализации многозначных логических функций, — Экспресс-информация. Вычислит, техника, 1975, № 13, реф. 80, с. 9—13.
86. *Жданок А. И., Растригин Л. А.* Динамика двуальтернативного выбора в задачах адаптации. — Автоматика и вычисл. техника, 1978, №4, с. 51—54.
87. *Жданок А. И., Растригин Л. А.* Формальный подход к созданию адаптивной структуры поисковых алгоритмов. — Пробл. случайного поиска (Рига), 1975, вып. 4, с. 48—53.
88. *Журавлев Ю. И.* Об алгебраическом подходе к решению задач распознавания и классификации. — Пробл. кибернетики (М.), 1978, вып. 33, с. 5—68.
89. *Журавлев Ю. И., Камилов М. М., Туляганов Ш. Е.* Алгоритмы вычисления оценок и их применение. Ташкент, Фан, 1974. 119 с.
90. *Загоруйко Н. Г.* Методы распознавания и их применения. М,; Сов. радио, 1972. 206 с.
91. *Зуев А. К., Растригин Л. А.* Оценка параметров объекта оптимизации. — В кн.: Автоматическое управление. Рига, Зинатне, 1967, с. 107—116.
92. *Ивахненко А. Г.* Системы эвристической самоорганизации в технической кибернетике. Киев, Техника, 1971. 372 с.
93. *Иоффе И. И., Федоров В. С., Мухенберг К. М., Фукс И. С.* Прогнозирование химических реакций методами статистической теории распознавания. — ДАН, 1969, т. 189, № 6, с. 1290.
94. *Калинников Ю. С.,.Лифшиц А. Л.* О некоторых модификациях алгоритма глобального статистического поиска по направляющей сфере. — В кн.: Задачи статистической оптимизации. Рига, Зинатне. 1971. с. 197—202.

95. *Канторович Л. В.* О методе наискорейшего спуска. — ДАН, 1947 т. 56, № 3, с. 233—236.
96. *Каплинский А. И., Краснепкер А. С., Цыпкин Я. З.* Рандомизация и сгла живание в задачах и алгоритмах адаптации. — Автоматика и телеме ханика, 1974, № 6, с. 47—57.
97. *Каплинский А. И., Лимарев А. Е., Чернышева Г. Д.* Построение рандо мизированных алгоритмов оптимизации. — Пробл. случайного поиска (Рига), 1980, вып. 8, с. 63—91.
98. *Каплинский А. И., Чернышева Г. Д.* О построении вероятностных ите ративных алгоритмов решения задачи целочисленного программирова ния. — Тр. НИИ математики ВГУ (Воронеж), 1973, вып. 9, с. 34—38.
99. *Каплинский А. И., Чернышева Г. Д.* Об одном способе построения адап тивных алгоритмов решения задач оптимизации с булевыми перемен ными. — Автоматика и телемеханика, 1976, № 10, с. 66—77.
100. *Карп В. П., Кунин П. Е.* Метод направленного обучения в проблеме узнавания. — В кн.: Автоматизация. Организация. Диагностика. М., Наука, 1971, ч. 1, с. 333—338.
101. *Карп В. П., Кунин П. Е.* Метод направленного обучения в переработан ной схеме М. М. Бонгарда и онкологическая диагностика. — В кн.: Моделирование обучения и поведения. М., Наука, 1975, с. 7—13.
102. *Катковник В. А.* Линейные оценки и стохастические задачи оптимизации (метод параметрических операторов усреднения). М., Наука, 1976. 488 с.
103. *Катковник В. Я., Антонов Г. Е.* Фильтрация и сглаживание функций многих переменных для целей поиска глобального экстремума. — Авто матика и вычисл. техника, 1970, № 4, с. 32—38.
104. *Кендалл М.* Ранговые корреляции. М., Статистика, 1975. 216 с.
105. *Кнут Д.* Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск. М, Мир, 1978. 840 с.
106. *Коган Я. А., Файнштейн И. А., Шейнман М. В.* Исследование и оптими зация систем программного обеспечения. — Автоматика и вычисл. тех ника, 1976, № 2, с. 55—63.
107. *Колосов В. В., Растригин Л. А.* Применение методов случайного поиска при последовательном планировании оптимальных экспериментов. — Зав. лаб., 1974, № 3, с. 302—305.
108. *Кондратьева В. А.* Оптимизация усвоения лексики иностранного языка. М., Высшая школа, 1974. 118 с.
109. *Корбут А. А., Финкельштейн Ю. Ю.* Дискретное программирование. М., Наука, 1969. 366 с.
ПО. *Корнеева А. В., Чавчанидзе В. В.* О применении концептуального подхода к задачам медицинской кибернетики. — В кн.: Материалы IV симпоз. по кибернетике. Тбилиси, Институт кибернетики АН ГССР, 1972, ч. 5, с. 106—109.
111. *Коробков Б. П.* Методы разрезания графов на минимально связные подграфы и их использование в задачах адаптивной обработки инфор мации (обзор). — В кн.: Адаптация в вычислительных системах. Рига, Зинатне, 1978, с. 107—129.
112. *Коробков Б. П., Подкорытов М. П.* Рандомизированная оптимизация си стемы интерфейсных модулей. — Системы автоматизации научных иссле дований (Рига), 1980, вып. 4, с. 6—17.
113. *Коробков Б. П., Растригин Л. А.* Методы структурной адаптации в процессах управления сложным объектом. — В кн.: Адаптация в систе мах обработки информации. Рига, Зинатне, 1977, с. 3—21.

114. *Коробков Б. П., Растригин Л. А.* Глобальный рандомизированный алго ритм разрезания графов. — В кн.: Структурная адаптация многомашин ных систем обработки информации. Рига, Зинатне, 1978, с. 56—62.
115. *Коробков Б. П., Растригин Л. А.* Рандомизированные алгоритмы агрега ции графов. — В кн.: Адаптация в вычислительных системах. Рига, Зинатне, 1978, с. 6—20.
116. *Коробков Б. П., Растригин Л. А.* Агрегация систем с оптимальным вза имным размещением агрегатов. — В кн.: Адаптация многомашинных вычислительных систем. Рига» Зинатне, 1980, с. 66—78.
117. *Коробков Б. П., Растригин Л. А.* Метод адаптивного разрезания графов и его использование в задаче сегментации. — Системы автоматизации научных исследований (Рига), 1980, вып. 4, с. 52—63.
118. *Красовский А. А.* Динамика непрерывных систем экстремального регулирования при случайных сигналах поиска. — Изв. АН СССР. Энер гетика и автоматика, 1960, № 3, с. 37—45.
119. *Круг Г. К., Сосулин Ю. А., Фатуев В. А.* Планирование эксперимента в задачах идентификации и экстраполяции. М., Наука, 1977. 208 с.
120. *Кудрин В. Г.* Универсальная автоматизированная адаптивная система оптимизации. — Автоматика и вычисл. техника, 1977, № 4, с. 54.
121. *Кулешов А. П.* Об одном классе адаптивных алгоритмов управления потоками. — В кн.: VII Всесоюз. конф. по теории кодирования и пере дачи информации. Москва—Вильнюс, 1978, ч. 3, с. 80—84.
122. *Кудрин В. Г.* Универсальная автоматизированная адаптивная система оптимизации. — Автоматика и вычисл. техника, 1977, № 4, с. 54.
123. *Лабутин В. К.* Очерки адаптации в биологии и технике. Л., Энергия, 1970. 160 с.
124. *Лавров С. С., Гончарова Л. И.* Автоматическая обработка данных. М., Наука, 1971. 160 с.
125. *Левин В. И.* Статистический надежностный синтез автоматов. Рига, Зинатне, 1974. 282 с.
126. Линейное и нелинейное программирование. Под ред. И. Н. Ляшенко. Киев, Вища школа, 1975. 370 с.
127. *Лихтенштейн В. Е.* Дискретность и случайность в экономико-математи ческих задачах. М., Наука, 1973. 376 с.
128. *Мак-Каллок У. С., Питтс У.* Логическое исчисление идей, относящихся к нервной активности. — В кн.: Автоматы. М., ИЛ, 1956, с. 362—384.
129. *Максименков А. В., Щерс А. Л.* Алгоритм сегментации машинных про грамм. — Автоматика и телемеханика, 1976, № 5, с. 52—60.
130. *Мартин Д.* Системный анализ передачи данных. В 2-х т. М., Мир, 1975. Т. 1. 256 с.; Т. 2. 430 с.
131. Медицинская информационная система. Сб. статей. Под ред. Н. М. Амо сова. Киев, Наукова думка, 1975.
132. Методы исследования нелинейных систем автоматического, управления. М., Наука, 1975. 448 с.
133. *Миркин Б. Г.* Проблема группового выбора. М., Наука, 1974. 256 с.
134. *Мкртчян С. О.* Проектирование логических устройств ЭВМ на нейрон ных элементах. М., Энергия, 1977. 200 с.
135. *Моцкус И. Б.* Многоэкстремальные задачи в проектировании. М., Наука, 1967. 214 с.
136. *Мухин В. И., Неймарк Ю. И., Ронин Е. И.* Автоматная оптимизация с эволюционной адаптацией. — Пробл. случайного поиска (Рига), 1973, вып. 2, с. 83—97.

137. *Налимов В. В.* Теория эксперимента. М., Наука, 1971. 206 с.
138. *Невельсон М. Б., Хасьминский Р. З.* Стохастическая аппроксимация и рекуррентное оценивание. М., Наука, 1972. 304 с.
139. *Нейман Дж., Моргенштерн О.* Теория игр и экономическое поведение. М., Наука, 1970. 707 с.
140. *Неймарк Ю. И.* Автоматная оптимизация. — Изв. вузов. Радиофизика, 1972, № 7, с. 967—971.
141. *Неймарк Ю. И.* Динамические системы и управляемые процессы. М., Наука, 1978. 336 с.
142. *Никифорова Н. Е.* О выборе оптимальных значений параметров опти мизатора. — Пробл. случайного поиска (Рига), 1974, вып. 3, с. 265—272.
143. *Никифорова Н. Е., Сытенко Л. В.* Исследования адаптивной подсистемы хранения информации в сети ИВЦ путем имитации ее работы с помощью ЭВМ. — Адаптивные системы (Рига), 1974, вып. 5, с. 55—71.
144. *Никифорова Н. Е.* Об одной модели адаптивного диспетчера в вычисли тельной системе. — В кн.: Структурная адаптация многомашинных си стем обработки информации. Рига, Зинатне, 1978, с. 30—35.
145. *Николаев Е. Г.* О скорейшем спуске со случайным выбросом направ лений. — Автоматика и вычисл. техника, 1970, № 5, с. 28—35.
146. *Овчинников В. В., Дшхунян В. Л., Чичерин Ю. В.* Проектирование быст родействующих микроэлектронных цифровых, устройств. М., Сов. радио, 1975. 288 с.
147. *Орлова Г. И., Дорфман Я. Г.* Оптимальное деление графа на несколько подграфов. — Изв. АН СССР. Техн. кибернетика, 1972, № 1, с. 118—121.
148. *Паперное А. И., Подымав В. Я.* Упорядочение массивов информации. М., Наука, 1973. 383 с.
149. *Первозванский А. А.* Поиск. М., Наука, 1970. 264 с.
350. *Перельман И. И.* Адаптивные алгоритмы текущей идентификации. М., Ин-т проблем управления, 1978. 80 с.
151. Перцептрон — система распознавания образов. Под ред. А. Г. Ивах-ненко. Киев, Наукова думка, 1975. 430 с.
152. *Петерсен И.* Статистическая оптимизация посредством сглаживания. — Изв. АН СССР. Техн. кибернетика, 1969, № 2, с. 36—44.
153. *Подвысоцкий Ю. С., Ротанов С. В.* Бит-ориентированный протокол уп равления информационным каналом доступа в сеть с коммутацией паке тов (КНП-2БИТ). Рига, Ин-т электроники и вычисл. техники АН ЛатвССР, 1979. 52 с.
154. *Половинкин А. И.* Метод оптимального проектирования с автоматиче ским поиском схем и структур инженерных конструкций. М., 1970. 162 с. (Сб. тр. ВНИИ транспорт, строительства, вып. 34).
155. *Половинкин А. И.* Алгоритм поиска глобального экстремума при проек тировании инженерных конструкций. — Автоматика и вычисл. техника, 1970, № 2, с. 31—37.
156. *Половинкин А. И.* Оптимальное проектирование с автоматическим поис ком схем инженерных конструкций. — Изв. АН СССР. Техн. киберне тика, 1971, № 5, с. 29—38.
157. *Пугачев В. С.* Теория вероятностей и математическая статистика. М., Наука, 1979. 496 с.
158. Распознавание физиологической активности химических соединений на перцептроне со случайной адаптацией структуры. — ДАН, 1971, № 4, с. 199—201.
159. *Растригин Л. А.* Способ автоматической настройки многопараметриче-

ских систем автоматического управления и регулирования на заданные условия. А. с. СССР № 129701 от 17. VIII 1959. — Бюл. изобрет., 1960, № 13.

160. *Растригин Л. А.* О вероятностных свойствах случайных процессов при их моделировании на ЦВМ. — Автоматика и вычисл. техника (Рига), 1966, вып. 13, с. 117—123.
161. *Растригин Л. А.* Статистические методы поиска. М., Наука, 1968. 376 с.
162. *Растригин Л. А.* Поисковый синтез датчика случайных чисел с заданной автокорреляционной функцией. — Автоматика и вычисл. техника, 1969, № 1, с. 33—38.
163. *Растригин Л. А.* Адаптивная идентификация параметров распределений и ее применение для оценки параметров объекта в процессе случайного поиска. — Автоматика и вычисл. техника, 1970, № 6, с. 39—44.
164. *Растригин Л. А.* Случайный поиск с линейной тактикой. Рига, Зинатне, 1971. 192 с.
165. *Растригин Л. А.* Смешанные алгоритмы случайного поиска. — Пробл. случайного поиска (Рига), 1973, вып. 2, с. 7—17.
166. *Растригин Л. А.* Системы экстремального управления. М., Наука, 1974. 630с.
167. *Растригин Л. А.* Структурная адаптация алгоритмов поисковой опти мизации. — Пробл. случайного поиска (Рига), 1976, вып. 5, с. 5—14.
168. *Растригин Л. А.* Структурная адаптация алгоритмов случайного по иска. — Вопр. кибернетики, вып. 45. Случайный поиск в задачах оптими зации. М., Научный совет АН СССР по комплексной проблеме «Кибер нетика», 1978, с. 5—12.
169. *Растригин Л. А.* Об особенностях учета ограничений в процессах слу чайного поиска. — Пробл. случайного поиска (Рига), 1978, вып. 7, с. 13—21.
170. *Растригин Л. А.* Адаптация сложных систем. — Изв. АН ЛатвССР, 1978, №5 (370), с. 87—97.
171. *Растригин Л. А.* Обучение с моделью. — В кн.: Человеко-машинные обучающие системы. М., Научный совет АН СССР по комплексной проб леме «Кибернетика», 1979, с. 40—49. (Вопр. кибернетики).
172. *Растригин Л. А.* Современные принципы управления сложными объек тами. М., Сов. радио, 1980. 232 с.
173. *Растригин Л. А.* Адаптация вычислительных сетей. — Вопр. киберне тики. Надежность информационного обмена в вычислительных сетях. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1980, с. 88—98.
174. *Растригин Л. А.* Структурная адаптация пакета программ оптимиза ции. — В кн.: Аналитические методы в экономике производства. Горький, ГГУ, 1980, с. 3—9.
175. *Растригин Л. А.* Вычислительные системы и сети как объекты примене ния случайного поиска (вместо предисловия). — Пробл. случайного поиска в вычислительных системах и сетях (Рига), 1982, вып. 10 (в пе чати).
176. *Растригин Л. А., Глаз А. Б., Ямпольская Т. С.* Синтез структуры решаю щего правила методами многомерной линейной экстраполяции. — Авто матика, 1979, № 5, с. 3—7.
177. *Растригин Л. А., Джандиери В. В.* Коллективные решения в задачах распознавания образов (обзор). — Автоматика, 1981, № 6 (в печати).
178. *Растригин Л. А., Мафжаров Н. Е.* Введение в идентификацию объектов управления. М., Энергия, 1977. 214 с.
179. *Растригин Л. А., Рипа К. К.* Алгоритм синтеза оптимальных планов

экспериментов с качественными и количественными факторами и его применение при исследовании ВС. — Пробл. случайного поиска в вычислительных системах и сетях (Рига), 1982, вып. 10 (в печати).

180. *Растригин Л. А., Рипа К. К.* Синтез факторных планов экспериментов методом адаптивного случайного поиска. — Пробл. случайного поиска (Рига), 1980, вып. 8, с. 237—253.
181. *Растригин Л. А., Рипа К. К.* Автоматная теория случайного поиска. Рига; Зинатне, 1973. 340 с.
182. *Растригин Л. А., Рипа К. К., Тарасенко Г. С.* Адаптация случайного поиска. Рига, Зинатне, 1978. 242 с.
183. *Растригин Л. А., Сытенко Л. В.* Многоканальные статистические оптими заторы. М., Энергия, 1973. 144 с.
184. *Растригин Л. А., Тарасенко Г. С.* Об одном адаптивном алгоритме слу чайного поиска. — Пробл. случайного поиска (Рига), 1974, вып. 3, с. 108—112.
185. *Растригин Л. А., Трахтенберг В. С.* Оценка числовых характеристик функций методом многомерной экстраполяции. — В кн.: Проблемы ста тистической оптимизации. Рига, Зинатне, 1968, с. 179—189.
186. *Растригин Л. А., Трахтенберг В. С.* Применение экстраполяции при оптимальном проектировании сложных систем. — В кн.: Методы стати стической оптимизации. Рига, Зинатне, 1968, с. 43—58.
187. *Растригин Л. А., Эренштейн М. Х.* Математические модели обучения (в задаче обучения запоминанию иностранных слов). — В кн.: Адаптация в системах обработки информации. Рига, Зинатне, 1977, с. 36—48.
188. *Растригин Л. А., Эренштейн М. Х.* О сходимости некоторых процессов обучения (на примере задачи стохастического запоминания). — Пробл. случайного поиска (Рига), 1978, вып. 6, с. 267—274.
189. *Растригин Л. А., Эренштейн Р. Х.* Принятие решения коллективом реша ющих правил в задачах распознавания образов. — Изв. АН СССР. Автоматика и телемеханика, 1975, № 9, с. 133—144.
190. *Растригин Л. А., Эренштейн Р. Х.* Коллектив алгоритмов. — В кн.: Тр. IV Междунар. объед. конф. по искусственному интеллекту. М., 1975, ч. 3, с. 138—144.
191. *Растригин Л. А., Эренштейн Р. Х.* Коллектив алгоритмов для обобщения алгоритмов решения задач. — Изв. АН СССР. Техн. кибернетика, 1978, №2, с. 116—126.
192. *Растригин Л. А., Эрмуйжа А. А.* Адаптивная процедура альтернатив ного управления в вычислительной системе. — В кн.: Структурная адап тация многомашинных систем обработки информации. Рига, Зинатне, 1978, с. 82—87.
193. *Растригин Л. А., Эрмуйжа А. А.* Об адаптивном управлении передачей данных в вычислительной сети. — В кн.: Адаптация многомашинных вычислительных систем. Рига, Зинатне, 1980, с. 36—49.
194. *Растригин Л. А., Эрмуйжа А. А.* Адаптивное управление длиной инфор мационного поля кадра в вычислительной сети. — Пробл. случайного поиска в вычислительных системах и сетях (Рига), 1982, вып. 10 (в печати).
195. *Растригин Л. А., Эрмуйжа А. А.* Адаптация процесса передачи данных в вычислительной сети. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 178—191.
196. *Растригин Л. А., Ямпольская Т. С., Растригин В. Л., Абрамович В. Л.* Адаптивная программа обучения запоминанию иностранных слов. — Адаптивные системы (Рига), 1972, вып. 2, с. 66—75.

197. *Репин В. Г., Философов Л. В.* Об оптимальном совместном использо вании алгоритмов распознавания. — Изв. АН СССР. Радиотехника и электроника, 1969, № 6, с. 963—973.
198. *Рипа К. К.* Алгоритм синтеза факторных планов эксперимента. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 250—262.
199. *Розенблатт Ф.* Принципы нейродинамики. Перцептроны и теория меха низмов мозга. Пер. с англ. М., Мир, 1965. 480 с.
200. *Рыбников К. А.* Введение в комбинаторный анализ. М., Изд-во МГУ, 1972. 252 с.
201. *Рыжков А. П.* Алгоритм разбиения графа на минимально связные под графы. — Изв. АН СССР. Техн. кибернетика, 1975, № 6, с. 122—128.
202. *Самойленко С. И.* Принципы адаптивной коммутации в вычислитель ных сетях. — В кн.: Структурная адаптация многомашинных систем об работки информации. Рига, Зинатне, 1978, с. 6—12.
203. Сети ЭВМ. Под ред. В. М. Глушкова. М., Связь, 1977. 280 с.
204. *Советов Б. Я.* Теория информации. Теоретические основы передачи ин формации в АСУ. Л., Изд-во ЛГУ, 1977. 184 с.
205. Современное состояние теории исследования операций. Под ред. Н. Н. Мои сеева. М., Наука, 1979. 464 с.
206. *Соснин Л. Б.* Автоматизация проектирования технологических марш рутов с применением метода обучения. — В кн.: Материалы II Всесоюз. науч.-техн. конф. «Автоматизация технической подготовки производства в машиностроении». Минск, 1972, ч. 2, с. 138—139.
207. *Срагович В. Г.* Моделирование некоторых случайных процессов. — Журн. вычисл. математики и мат. физики, 1963, т. 3, *№* 3, с. 589—593.
208. *Стоян Ю. Г., Соколовский В. З.* Метод сужающихся окрестностей для минимизации функционалов, заданных на перестановках. — Автоматика и вычисл. техника, 1979, № 2, с. 63—69.
209. *Стронгин Р. Г.* Численные методы в многоэкстремальных задачах. М., Наука, 1978. 240 с.
210. Структурная адаптация многомашинных систем обработки информации. Под общ. ред. Л. А. Растригина. Рига, Зинатне, 1978. 110 с.
211. *Сытенко Л. В.* О модели подсистемы хранения информации в терри ториальной вычислительной сети с адаптивным накоплением данных в информативно-вычислительных центрах. — Адаптивные системы (Рига), 1974, вып. 5, с. 39—54.
212. *Тарасенко Г. С.* Исследование адаптивного алгоритма случайного поиска. — Пробл. случайного поиска (Рига), 1976, вып. 5, с. 119—124.
213. *Тарасенко Г. С.* Построение релаксационных алгоритмов поиска с адап тацией плотности распределения случайных векторов. — Пробл. слу чайного поиска (Рига), 1978, вып. 7, с. 97—105.
214. *Тарасенко Г. С.* Исследование релаксационных алгоритмов случайного поиска общего вида. — Пробл. случайного поиска (Рига), 1978, вып. 7, с. 67—96.
215. *Тарасенко Г. С.* Структурная адаптация пакета программ оптимизации.— В кн.: Структурная адаптация многомашинных систем обработки ин формации. Рига, Зинатне, 1978, с. 73—77.
216. *Тарасенко Г. С., Эренштейн М. Х.* Исследование сходимости одного про цесса обучения. — Пробл. случайного поиска (Рига), 1980, вып. 8, с. 280—288.
217. *Татубалин В. Н.* Теория вероятностей в естествознании. М., Знание, 1972. 48 с.

218. *Татубалин В. Н.* Теория вероятностей (краткий курс и научно-методи ческие замечания). М., Изд-во МГУ, 1972. 231 с.
219. *Татубалин В. Н.* Статистическая обработка рядов наблюдений. М, Зна ние, 1973. 62 с.
220. Теория и применение адаптивных систем. Алма-Ата, Казахский политех нический ин-т им. В. И. Ленина, 1971. 312 с.
221. Теория и применение случайного поиска. Рига, Зинатне, 1969. 306 с.
222. *Файнштейн И. А.* Математические модели и методы сегментации программ для вычислительных систем с иерархической памятью. — Моделиро вание сложных систем (Рига), 1973, вып. 2, с. 60—71.
223. *Федоров В. В.* Теория оптимального эксперимента. М., Наука, 1971. 312 *с.*
224. *Фещенко В. П., Матюшков Л. П.* Итерационный алгоритм разрезания графа на *k* подграфов. — Автоматизация проектирования сложных си стем. (Направление: Вычислительная техника в машиностроении). Науч.-техн. сб. (Минск), 1976, вып. 2, с. 74—77.
225. *Фиакко А., Мак-Кормик Г.* Нелинейное программирование. Методы после довательной безусловной оптимизации. М., Мир, 1972. 240 с.
226. *Флейшман Б. С., Букатова И. Л.* О некоторых аналитических оценках параметров эволюционного моделирования. — Автоматика и вычисл. техника, 1974, № 4, с. 34—39.
227. *Фогель Л., Оуэне А., Уолш М.* Искусственный интеллект и эволюцион ное моделирование. М., Мир, 1969. 230 с.
228. *Фурунжиев Р. И.* Проектирование оптимальных виброзащитных систем. Минск, Вышэйшая школа. 320 с.
229. *Фурунжиев Р. И.* Вычислительная техника и ее применение. Минск, Вышэйшая школа, 1975. 400 с.
230. *Хант* 3.,: *Мартин Дж., Стоун Ф.* Моделирование процесса формирования понятия на вычислительной машине. М., Мир, 1970. 300 с.
231. *Харди Г. Г., Литтлвуд Д. Е., Полна Г.* Неравенства. М., ИЛ, 1948. 456 с.
232. *Хасьминский Р. 3.* Применение случайного поиска в задачах оптимизации и опознания. — Пробл. передачи информации, 1965, № 3, с. 113—117.
233. *Химмельблау Д.* Прикладное нелинейное программирование. Пер. с англ. М., Мир, 1972. 534 с.
234. *Цетлин М. Л.* Конечные автоматы и моделирование простейших форм поведения. — В кн.: Исследования по теории и моделированию биоло гических систем. М., Наука, 1969, с. 11—107.
235. *Цыпкин Я. 3.* Теория импульсных систем. М., Физматгиз, 1958. 724 с.
236. *Цыпкин Я. 3.* Применение метода стохастической аппроксимации к оценке неизвестной плотности распределения по наблюдениям. — Автоматика и телемеханика, 1966, № 8, с. 94—96.
237. *Цыпкин Я. 3.* Адаптация и обучение в автоматических системах. М., Наука, 1968. 400 с.
238. *Цыпкин Я. 3.* Сглаженные рандомизированные функционалы в теории адаптации и обучения. — Автоматика и телемеханика, 1971,№8,с.29—50.
239. *Чавчанидзе В. В.* Аналитическое решение задач формирования понятия и распознавания образов. — Сообщ. АН ГССР, 1971, т. 61, № 1,с. 37—40.
240. *Чернышева Г. Д.* Об одном вероятностном способе адаптивной алгорит мизации задач с булевыми переменными. — В, кн.: Адаптация в системах обработки информации. Рига, Зинатне, 1977, с. 84—94.
241. *Шлезингер М. И.* Синтаксический анализ двумерных зрительных сиг налов в условиях помех. — Кибернетика, 1976, № 4, с. 113—129.

242. *Шоломов Л. А.* О реализации булевой функции многопороговым элемен том. — Автоматика и телемеханика, 1966, № 3, с. 97—104.
243. *Шолохов В. Г.* Система автоматической настройки генератора флуктуа ционной ЭДС. — Исследования по физике и радиотехнике. Тр. Моск. физ.-техн. ин-та, 1962, т. 8, с. 33—37.
244. Элементы теории передачи дискретной информации. Под ред. А. П. Пур това. М., Связь, 1972. 232 с.
245. *Эшби У. Р.* Схема усилителя мыслительных способностей. — В кн.: Автоматы, М., ИЛ, 1956.
246. *Эшби У. Р.* Конструкция мозга. М., ИЛ., 1962. 398 с.
247. *Юдин Д. Б.* Методы качественного анализа сложных систем. II. — Изв. АН СССР. Техн. кибернетика, 1966, № 1, с. 3—16.
248. *Юдин Д. Б.* Математические методы управления в условиях неполной информации. М., Сов. радио, 1974. 400 с.
249. *Юдин Д. Б.* Задачи и методы стохастического программирования. М., Сов. радио, 1979. 392 с.
250. *Яблонский А. И.* Рандомизированная стратегия в адаптивных процессах двуальтернативного выбора. — Автоматика и вычисл. техника, 1977, № 1, с. 32—36.
251. *Яблонский А. И.* Рандомизированная стратегия в процессах структур ной адаптации. — В кн.: Структурная адаптация сложных систем уп равления. Воронеж, ВПИ, 1977, с. 18—30.
252. *Яблонский А. И.* Рандомизация стратегии в задаче о «двуруком бан дите». — В кн.: Моделирование и оптимизация в условиях системы авто матизированного проектирования. Материалы респ. семинара. Таллин, НИИ Таллин, электротехн. з-да, 1977, с., 113—118.
253. *Яблонский А. И.* Рандомизированные адаптивные процедуры многоаль тернативного выбора. — В кн.: Адаптация многомашинных вычислитель ных систем. Рига, Зинатне, 1980, с. 94—99.
254. *Яблонский С. В,* Введение в дискретную математику. М., Наука, 1979. 272 с.
255. *Якубович В. А.* Метод рекуррентных целевых неравенств в теории адап тивных систем. — В кн.: Адаптивные системы. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1976, с. 32—64. (Вопр. кибернетики).
256. *Cover T., Helman M.* The two-armed-bandit problem with time-invariant finite memory. — IEEE Trans., 1970, vol. IT-16, N 2, p. 185—195.
257. *Ercoli P., Mercurio L.* Threshold logic with one or more than one thre shold. — In: Inform. Progress 1962. Amsterdam, N. Holland Publ. Co., 1963, p. 741—746.
258. *Gutierrez I., Moraga C.* Multithreshold periodic ternary threshold logic. — In: Proc. Intern. Symp. Multiple-Valued Logic. New York, 1974, p. 413—422.
259. *larvi T.* A random search optimizer with an application to a max-min problem. A special trajectory estimation problem. Turku, Univ. of Turku, 1973, p. 70.
260. *Kiefer I.* Optimum experimental designs with applications to systematic and rotable designs. — In: Proc. 4th Berkeley Symp. Mathematical Statis tics and Probability, 1961, p. 381—398.
261. *Mitchel T. I.* Computer construction of «D-optimal» first-order desighs. — Technometrics, 1974, vol. 16, N 2, p. 211—220.
262. *Narenda K. C.* Application of learning automata to telephone traffic routing and control. — IEEE Trans., 1977, vol. SMC-7, N 11, p. 785—792.

263. *Robbins M.* Some aspects of the sequential design of experiments. — Bull, Amer. Math. Soc., 1952, vol. 58, p. 529—532.
264. *Satterthwaite F. E.* Random balance experimentation. — Technometrics, 1959, vol. 1, N 2, p. 111.

Дополнительный список литературы*

265. Адаптивные телеизмерительные системы. Л. Энергоиздат, 1981. 246 с.
266. Адаптивная АСУ производством. М., Статистика, 1981. 205 с.
267. *Байзель М. М., Тарасенко Г. С.* Исследование адаптивного алгоритма оптимизации в обстановке помех. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 106—124.
268. Бектаев С. К., Коробков Б. П. Адаптивная оптимизация устройств сопряжения по критерию надежности. — Системы автоматизации научных исследований (Рига), 1980, вып. 4, с. 3—6.
269. *Гринченко С. Н., Загускин С. Л.* К вопросу о моделировании биологи ческих процессов биосферы на основе иерархии алгоритмов случайного поиска. — В кн.: Проблемы биосферы. Информационные материалы. Вып. 2. М., Научный совет АН СССР по проблемам биосферы, 1981, с. 136—142.
270. *Гринченко С. Н., Растригин Л. А.* О бионических алгоритмах случайного поиска. — В кн.: Вопросы кибернетики. Задачи и методы адаптивного управления. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1981, с. 137—147.
271. *Жданок А. И., Коробков Б. П.* Локальные характеристики и сходимость рандомизированного алгоритма разрезания графов. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 40—70.
272. Задачи и методы адаптивного управления. Вопросы кибернетики. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1981. 190 с.
273. *Зидере Я. Э., Растригин Л. А., Эренштейн М. Х.* Адаптивная система обучения с моделью. — Управл. системы и машины, 1980, № 6, с. 118—121.
274. *Зиновьев Э. В., Стрекалев А. А.* Оптимизация распределения файлов а сетях ЭВМ. — В кн.: Адаптация в многомашинных вычислительных системах. Рига, Зинатне, 1980, с. 79—94.
275. *Карманов В. Г.* Математическое программирование. М., Наука, 1980. 256 с.
276. *Коробков Б. П.* Рандомизированные методы решения некоторых задач адаптации вычислительных систем. — Пробл. случайного поиска в вы числ. системах и сетях (Рига), 1982, вып. 10 (в печати).
277. *Коробков Б. П., Растригин Л. А.* Рандомизированные методы разреза ния графов. — Изв. АН СССР. Техн. кибернетика, 1982 (в печати).
278. *Лбов Г. С.* Методы обработки разнотипных экспериментальных данных. Новосибирск, Наука, 1981. 284 с.
279. *Ливчак И. В., Эрмуйжа А. А.* Некоторые алгоритмы двуальтернативной адаптации, их исследование и возможности их применения при передаче данных в вычислительных сетях. — Пробл. случайного поиска в вычисл. системах и сетях (Рига), 1982, вып. 10 (в печати).

* Эти работы были опубликованы после сдачи рукописи книги в издательство в апреле 1979 г.

280. *Лукашин Ю. П.* Адаптивные методы краткосрочного прогнозирования. М., Статистика, 1979. 254 с.
281. *Малков В. П., Угадчиков А. Г.* Оптимизация упругих систем. М., Наука, 1981. 286 с.
282. *Никифорова Н. Е.* Об одной модели вычислительной системы с адап тивной дисциплиной обслуживания. — В кн.: Адаптация в многомашин ных вычислительных системах. Рига, Зинатне, 1980, с. 56—65.
283. *Никифорова Н. Е.* Исследование модели вычислительной системы с адап тивной дисциплиной обслуживания методом планирования экспери мента. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 228—235.
284. *Никифорова Н. Е., Стрекалев А. А.* Об одном адаптивном алгоритме распределения файлов в вычислительной сети. — Пробл. случайного поиска в вычисл. системах и сетях (Рига), 1982, вып. 10 (в печати).
285. *Растригин Л. А.* Проблемы распознавания образов. — В кн.: Некото рые вопросы теории распознавания образов в управлении. Рига, 1979, с. 3—30.
286. *Растригин Л. А.* Случайный поиск. М., Знание, 1979. 64 с. (Серия «Ма тематика и кибернетика», вып. 1),
287. *Растригин, Л. А.* Случайный поиск как метод адаптации вычислитель ных систем. — В кн.: Вопросы кибернетики. Проблемы информацион ного обмена в вычислительных сетях. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1979, с. 113—129.
288. *Растригин Л. А.* Двадцатилетие проблемы. — Пробл. случайного поиска (Рига), 1980, выш 8, с. 5—14.
289. *Растригин Л. А.* Диалоговое планирование эксперимента в задачах оп тимального проектирования. — В кн.: Интерактивная технология в САПР. Таллин, НИИ Таллин, электротехн. з-да, 1981, с. 130—132.
290. *Растригин Л. А.* Искусственный интеллект, адаптация и микроэлектро ника. — Микроэлектроника, 1981, т. 10, вып. 1, с. 4—25.
291. *Растригин Л. А.* Случайный поиск в задачах сегментации графов. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 192—227.
292. *Растригин Л. А.* Проблемы поисковой адаптации в вычислительных се тях. М., Научный совет АН СССР по комплексной проблеме «Киберне тика», 1982 (в печати).
293. *Растригин Л. А., Рипа К. К.* Синтез оптимальных планов экспериментов со смешанными факторами методами случайного поиска. — Зав. лаб., 1982 (в печати).
294. *Растригин Л. А., Самсон Д. В.* Восстановление таблиц методом много мерной линейной экстраполяции. — В кн.: Методы принятия решений. Рига, РПИ, 1979, с. 76—84.
295. *Растригин Л. А., Самсон Д. В.* Поисковый метод восстановления таб лиц. — В кн.: Методы принятия решений в условиях неопределенности. Рига, РПИ, 1980, с. 69—75.
296. *Растригин Л. А., Ширин А. В.* Адаптивный выбор дисциплины обслужи вания в вычислительной системе. — В кн.: Адаптация в многомашинных вычислительных системах. Рига, Зинатне, 1980, с. 49—56.
297. *Растригин Л. А., Эренштейн М. Х.* Альтернативная адаптация модели обучения. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 236—249.
298. *Расщепляев Ю. С., Фандиенко В. Н.* Синтез моделей случайных процес сов для исследования автоматических систем управления. М., Энергия, 1981. 144 с.
299. *Самойленко С. И.* Метод адаптивной коммутации. — В кн.: Вопросы кибернетики. Проблемы информационного обмена в вычислительных се-

тях. М., Научный совет АН СССР по комплексной проблеме «Кибернетика», 1979, с. 130—159.
300. *Самойленко С. И.* Адаптивная коммутация в вычислительных сетях. — В кн.: Адаптация в многомашинных вычислительных системах. Рига, Зинатне, 1980, с. 7—36.
301. *Стоян Ю. Г., Соколовский В. З.* Решение некоторых многоэкстремаль ных задач методом сужающихся окрестностей. Киев, Наукова думка, 1980. 208 с.
302. *Тарасенко Г. С.* Об одном адаптивном алгоритме случайного поиска для задач с ограничениями типа неравенств. — Пробл. случайного поиска (Рига), 1981, вып. 9, с. 71—82.
303. *Якубайтис Э. А.* Архитектура вычислительных сетей. М., Статистика, 1980. 279 с.
304. *Rubinstein R. Y.* Simulation and the Monte Carlo method. New York, Wiley, 1981. 275 p.

Леонард Андреевич РАСТРИГИН

Адаптация сложных систем

Методы и приложения

Редактор Ф. ФЕРБЕР
Художник Г. КРУТОЙ
Художественный редактор Э. БУРОВА
Технический редактор Э. ПОЧА
Корректор А. МОРОЗОВА

ИБ № 731

Сдано в набор 28.11.80. Подписано в печать 04.11.81. ЯТ 07376. Формат 60X90/16. Бумага типогр. № 1. Гарнитура литературная. Высокая печать. 23,5. физ. печ. л.; 23,5 усл. печ. л.; 20,55 уч.-изд. л. Тираж 1500 экз. Заказ *Ns* 2535. Цена 1 р. 60 к.
Издательство «Зинатне», 226018 Рига, ул. Тургенева, 19. Отпечатано в типографии «Циня» Государственного комитета Латвийской ССР по делам издательств, полиграфии и книжной торговли, 226011 Рига, ул. Блауманя, 38/40.