**Лекция Last**

**Методы штрафных функций**

Для минимизации функций многих переменных были разработаны различные методы, среди которых следует отметить метод проекции градиента, метод условного градиента, метод линеаризации и методы штрафных функций.

Методы проекции градиента, условного градиента и линеаризации оказываются эффективными, когда допустимое множество оптимизационной задачи имеет достаточно простую структуру [5, 6].

Методы штрафных функций оказались более приспособленными из существующих методов оптимизации для решения задач с ограничениями, имеющими не совсем простую структуру.

В связи с этим, в пособии из методов минимизации функций многих переменных с ограничениями будут рассматриваться только методы штрафных функций.

Идея метода штрафных функций состоит в преобразовании исходной задачи с ограничениями в одну или в последовательность задач без ограничений. Для этого строится вспомогательная функция, в которую наряду с целевой функцией исходной задачи включаются штрафные функции. Штрафные функции строятся таким образом, чтобы нарушение каких-либо ограничений исходной задачи приводило бы к увеличению величины штрафных функций, соответствующих нарушенным ограничениям. При увеличении величины нарушения ограничений величина штрафных функций, соответствующих этим увеличивающимся ограничениям также увеличиваются. При увеличении коэффициента штрафа значения штрафных функций в точках, находящихся в допустимых областях этих функций, уменьшаются, а в точках, нарушающих ограничения – значительно увеличиваются.

Определение. Функция называется [5,6] штрафной функцией множества , если и при всех выполняется условие



Для пояснения такого преобразования и принципов построения штрафных функций рассмотрим несколько задач.

В одной задаче требуется найти минимум функции  при наличии ограничения , .

Преобразуем эту задачу в следующую задачу безусловной оптимизации:

минимизировать , ,

где  – некоторое большое положительное число, которое называется коэффициентом штрафа. Из вида функции, которая минимизируется в задаче безусловной оптимизации, видно, что в точке , близкой к её минимуму, величина  должна быть близкой к нулю. В противном случае при достаточно большом  можно найти точку , в которой приращение  функции, увеличивающее её значение , будет меньше, чем уменьшение величины .

Рассмотрим другую задачу, в которой имеется ограничение-неравенство минимизировать  при наличии ограничения , .

В этом случае нельзя использовать такой же штраф, поскольку в этом случае при  штраф будет взиматься независимо от знака . Штрафную функцию при таком ограничении можно задать в следующем виде: . При использовании такой штрафной функции величина штрафа будет равна  при  и нулю, когда .

Тогда эквивалентной задачей безусловной оптимизации для данной задачи будет следующая:

минимизировать , .

Штрафную функцию необходимо выбирать так, чтобы при минимизации целевой функции штраф был положительным в недопустимых точках и равным нулю в допустимых точках.

Если в задаче ограничения заданы следующим образом

, и ,

то в качестве штрафной функции может быть выбрана следующая:

.

Функцию  будем называть вспомогательной. Задачу минимизации вспомогательной функции будем называть вспомогательной задачей.

Различают три основных вида методов штрафных функций:

- методы внутренней точки;

- методы внешней точки;

- комбинированные методы.

При использовании методов внутренней точки минимум вспомогательной функции всегда находится в допустимой области. Начальная точка для определения минимума вспомогательной функции с помощью методов безусловной минимизации также всегда должна находиться в допустимой области.

В методах внешней точки последовательность минимумов вспомогательной функции обычно находится вне допустимой области. Это позволяет выбирать произвольную начальную точку для методов, с помощью которых будет определяться безусловной минимум вспомогательной функции. Такая точка необязательно должна находиться в допустимой области задачи, как это требуется в методах внутренней точки.

В комбинированных методах в процессе вычислений одни из ограничений выполняются, а другие не выполняются, особенно те, которые имеют вид равенств. Однако при достижении решений в методах штрафных функций все ограничения выполняются.

В разных видах методов штрафных функций используются определённые типы штрафных функций.

В методах внутренней точки для ограничений вида могут использоваться следующие штрафные функции:

- 

- .

В методах внешней точки для ограничений вида  могут использоваться следующие штрафные функции:

- , 

- , .

- , .

Для ограничений-равенств вида  в методах внешней точки может использоваться следующая штрафная функция:

- .

В комбинированных методах для ограничений, которые должны выполняться могут использоваться штрафные функции методов внутренней точки, а для ограничений, которые не обязательно должны выполняться могут использоваться штрафные функции методов внешней точки.

Штрафные функции добавляются к целевой функции решаемой задачи таким образом, чтобы решалось семейство задач без функциональных ограничений с разными штрафными параметрами. Последовательность решений этих задач при неограниченном возрастании штрафного параметра должна сходиться к решению исходной задачи. Это является основной целью методов штрафных функций.

Сформулируем и докажем некоторые свойства штрафных и вспомогательных функций, позволяющие обосновано применять методы штрафных функций для решения задач оптимизации с ограничениями.

Свойства штрафных и вспомогательных функций удобно продемонстрировать на примере рассмотрения конкретной задачи. Для этого рассмотрим следующую задачу.

Требуется минимизировать непрерывную выпуклую функцию , ,

при наличии ограничений 

,

где  – непрерывные вогнутые функции (),  – непрерывные линейные функции (), допустимое множество , задаваемое ограничениями  () и  (), не пусто.

Множество  в этом случае имеет вид:



Данная задача является задачей выпуклого программирования

Для решения этой задачи с помощью метода штрафных функций построим штрафную функцию  следующим образом:



и сформируем вспомогательную функцию , которая имеет вид:

,

где  – коэффициент штрафа.

Функция  будет выпуклой и непрерывной, но не гладкой функцией.

Для методов штрафных функций сформулированы и доказываются теоремы о сходимости.

**Методы поиска решений в пространстве состояний.**

**Путь решения задачи**

Выполнение определённых действий и команд в системе или на объекте, находящихся в начальном или промежуточном состоянии меняет эти состояния, порождая новые, которых вообще-то может быть много и которые образуют то, что называется *пространством состояний.*

Среди таких состояний, если система или объект исправны, и работают без ошибок, как правило, имеется состояние, которое будет соответствовать желаемому или целевому, и процесс поиска решения при попадании в это состояние будет закончен.

Процесс поиска такого состояния может быть сложным, и поэтому возникает потребность в некотором едином методе представления множества состояний и поиска решений. Таким методом является метод, использующий представление состояний, в которых может находиться система или объект, в виде графа.

Если отождествить состояние , в котором находилась система или объект перед выполнением действий или команд, с корнем или начальной вершиной такого графа, то, применяя к состоянию какой-либо оператор , мы порождаем новое состояние , образуя тем самым следующую вершину графа (рис. 2.1).

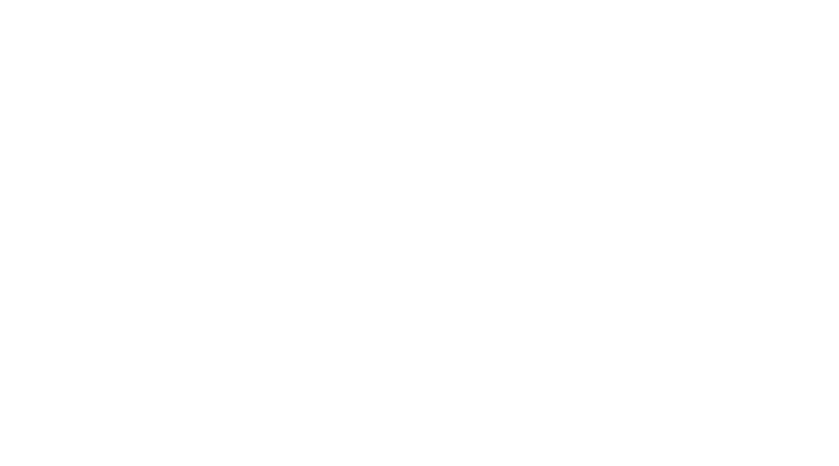
























































Рис. 6.1. Пример фрагмента графа, представляющего решение задачи.

Эта новая вершина может быть промежуточной или целевой. Если вершина промежуточная, то процесс порождения новых вершин (с помощью операторов ) будет продолжен, пока не найдётся целевая. Процесс применения оператора к некоторой вершине называется *раскрытием вершины*. От каждой порожденной вершины к породившей ее расставляются указатели. Такие указатели позволяют найти путь к начальной вершине, после того, как будет обнаружена целевая.

Общая процедура построения дерева в пространстве состояний при этом выглядит следующим образом.

1). К корню дерева () применяются операторы из множества *G* (их может быть несколько). Полученные при этом вершины образуют первый уровень новых вершин.

2). Каждая из вновь полученных вершин проверяется, не является ли она целевой. Если нет, то процесс продолжается по отношению к каждой из них. Образуется второй уровень вершин. Если к какой-либо вершине никакой оператор из *G* не применим, то эта вершина становится терминальной (конечной). Как видим, на каждом шаге проводятся две операции: порождение новой вершины и проверка, не является ли новая вершина целевой, т.е. совпадающей с целевым состоянием.

3). Когда целевая вершина будет найдена, в обратном направлении (от цели к началу) просматриваются указатели дуг и выделяется путь решения. Практически этот путь удобнее отображать посредством операторов, связанных с этими дугами (см. рис. 2.1).

В общем случае число вершин может быть большим. Их последовательное раскрытие, анализ и пометка пути осложняют задачу. Возникает проблема перебора вершин, связанные с выбором порядка их порождения и анализа. Здесь возможны следующие варианты:

* если вершины раскрываются в том же порядке, в котором они порождаются, то такой процесс называется полным перебором в ширину *(breadth- first process*);
* если на каждом шаге первой раскрывается вершина, которая была построена последней, то такой процесс называется полным перебором в глубину *(depth-firstprocess).* В этих процессах расположение целевой вершины не влияет на порядок раскрытия. Поэтому эти процессы часто называют процессами слепого перебора;
* если есть некоторая дополнительная (эвристическая) информация о предметной области, которая позволяет делать суждения о характере графа пространства состояний и расположения цели, то такой метод построения графа называется эвристическим («эвристический» означает «служащий открытию»). Эвристическая информация, опирающаяся, как правило, на предыдущий опыт, позволяет выполнять поиск в наиболее перспективных направлениях.

Говоря о графе, будем рассматривать только один наиболее простой его тип — граф типа «дерево». Как известно, *деревом* называется граф, каждая вершина которого имеет только одну вершину, непосредственно предшествующую ей (родительскую), за исключением вершины-корня, которая предшествующих вершин не имеет.

**Метод полного перебора в ширин*у***

Как уже было сказано, в этом методе вершины раскрываются в том порядке, в котором они строятся. Основной алгоритм состоит в выполнении следующих действий.

1). Раскрывается начальная вершина . Она раскрывается до тех пор, пока ее можно раскрыть, применяя один и тот же оператор (или разные, смотря по условию). При этом образуются вершины первого уровня: , , ... . Их раскрывают в свою очередь, и образуются вершины второго уровня и т.д. (рисунок 2.1 может служить примером этого метода: и - вершины первого уровня, и - вершины второго уровня, и - третьего и т.д.).

2). Расставляются указатели, ведущие от новых вершин к корню. Это могут быть условные имена, буквы, цифры, имена операторов и т.п. Однако среди них могут быть и реальные величины, например, расстояния, стоимость, вес и т.д.

3). Проверяется, нет ли среди полученных вершин целевой . Если есть, то формируется решение на основе соответствующего оператора. Если целевых вершин нет, то рассматривается первая порожденная вершина и к ней применяется тот же алгоритм. После чего, переходят ко второй и т.д., пока среди получаемых вершин не окажется целевой.

Метод полного перебора в ширину гарантируют нахождение целевой вершины как раз потому, что перебор - полный. Путей достижения цели, вообще говоря, может быть много. В этом случае у нас имеется возможность выбрать наикратчайший (или самый дешевый, или самый легкий - критериев много) путь. Но может быть случай, когда граф поиска окажется бесконечным и тогда этот алгоритм никогда не кончит работу.

Таким образом, метод полного перебора гарантирует поиск оптимального решения, если дерево пространства состояний не бесконечно.

Синонимами названия метода являются: метод грубой силы, метод проб и ошибок. На рис. 6.2 показан фрагмент дерева полного перебора в ширину.

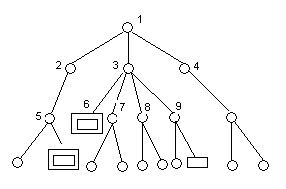


Рис. 6.2. Пример графа поиска, построенного при поиске в ширину.

**Метод полного перебора в глубину**

В методе перебора в глубину раскрываются, прежде всего вершины, которые были построены последними.

Первой раскрываемой вершиной, а следовательно, и последней, является корневая, но процесс всегда будет идти по самой левой ветви вершин. Чтобы как-то ограничить перебор, вводится понятие глубины вершины в дереве перебора. Полагаем, что глубина корня дерева равна нулю, а глубина любой последующей вершины равна единице плюс глубина вершины, непосредственно ей предшествующей.

Отсюда следует, что наибольшую глубину всегда будет иметь та вершина, которая должна быть в этот момент раскрыта. Если образующийся путь оказывается бесполезным, то есть при заданной глубине раскрытия целевой вершины не получилось, необходимо вернуться в вершину, предшествующую раскрытой и попытаться еще раз применить к ней операцию раскрытия. И так до тех пор, пока не будет получена целевая вершина. На рис. 6.3 показан элемент дерева полного перебора в глубину.

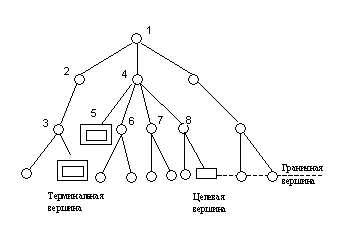


Рис. 6.3. Пример графа поиска, построенного при поиске в глубину.

Алгоритм перебора в глубину состоит в следующем.

1). Раскрывается начальная вершина соответствующая начальному состоянию *.*

2) Раскрывается первая вершина, получаемая в результате раскрытия . Ставится указатель.

3) Если она раскрывается, то следующей будет раскрываться вновь порождённая вершина. Если вершина не раскрывается, то процесс возвращается в предыдущую вершину .

4) По получении целевой вершины, процесс раскрытия заканчивается и по указателям строится путь, ведущий к корню.

5). Если для заданной глубины раскрытия целевая вершина не находится, то весь процесс повторяется снова, а в качестве новой вершины рассматривается самая левая из полученных вершин на предыдущем этапе.

Соответствующие дугам операторы образуют решение задачи.

Так же, как и метод поиска “в ширину”, этот метод относится к методам “грубой силы”. Он обеспечивает перебор всех состояний, если, конечно, прежде не определит целевую вершину .

**Эвристические методы поиска в пространстве состояний**

Методы полного перебора гарантируют решение задачи, если оно существует, а при наличии нескольких решений, гарантирует оптимальное. Однако на практике эти методы используются для решения лишь небольших по размерности графа состояний задач. Для реальных случаев чаще всего используется дополнительная информация, основанная на предыдущем опыте или полученная на основании теоретических выводов.

Такая информация называется *эвристической*, а организованная в правила - *эвристическими правилами* или *эвристиками*. Эвристическая информация носит сугубо специальный характер и может применяться только в рамках данной задачи, в лучшем случае, в рамках задач данного класса.

Чаще всего практические задачи решаются с помощью эвристического поиска. Эвристический поиск основан на функции оценки состояния. В отличие от алгоритмов полного перебора, эвристический поиск позволяет ранжировать пространственные состояния на основе их «перспективности». Эвристический поиск ищет в пространстве состояний более целенаправленно, чем алгоритмы полного перебора. Важно, что во многих задачах оценка состояния есть наилучшая оценка пути достижения данного состояния из начального. Если в нашей задаче возможна такая оценка, то алгоритмы эвристического поиска значительно упрощаются.

Основной класс алгоритмов эвристического поиска – это поиск от наилучшего состояния. Он включает три основных алгоритма: это жадный поиск, лучевой поиск и А\*. Общая их идея основана на поддержании в процессе поиска множества достигнутых состояний и выборе на каждом шаге одного или нескольких лучших состояний.

Простейший из них — **жадный поиск**. Если его применить в задаче поиска наилучшего пути в графе, это даст известный [алгоритм Дейкстры](http://ru.wikipedia.org/wiki/%C0%EB%E3%EE%F0%E8%F2%EC_%C4%E5%E9%EA%F1%F2%F0%FB). Жадный поиск, выбирая состояние, из которого будет продолжаться поиск, ищет состояние с наилучшей оценкой пути от начального в данное. Поэтому он и называется «жадным» — поскольку «хватает» самое лучшее на данный момент состояние, не думая о последствиях. Естественно, как и многие жадные алгоритмы, такая стратегия не приводит к оптимальному решению. Конечные состояния зачастую достигаются слишком длинным, неоптимальным путем. Кроме того, жадный поиск постоянно должен поддерживать множества всех достигнутых состояний, которых может быть слишком много, отчего чрезмерно расходуется память.

Примером, где могут успешно использоваться идеи жадных алгоритмов, является задача составления числа из разрядов. Простейшая постановка такой задачи имеет следующий вид.

Пусть заданы цифры 1, 3, 9, 3, 4, 2, из которых надо составить наибольшее число. Каждая цифра может быть использована только один раз.

Используя идеи и принципы жадных алгоритмов, построим для данной задачи жадный алгоритм следующего вида по шагам.

Шаг 1. Выбираем наибольшее из имеющихся цифр и ставим, его справа к ранее установленным цифрам или первым, если таковых ещё нет. Следует переход к Шагу 2.

Шаг 2. Производится проверка. Если в заданном множестве после вычёркивания использованной цифры остались другие, то следует переход к Шагу 1. В противном случае к Шагу 3.

Шаг 3. Формирование числа закончено.

Решение. 9 → 4 → 3 → 3 → 2 → 1 = 943321.

К сожалению, жадный алгоритм является эвристическим и не всегда позволяет получать оптимальное решение задач.

**Алгоритм лучевого поиска и алгоритм А\***— это попытки улучшить поведение жадного поиска и исправить эти две присущие ему проблемы. Лучевой поиск работает следующим образом: на каждом шаге мы поддерживаем некоторое множество из N лучших состояний. Далее из каждого из этих состояний делаем все возможные шаги и получаем множество состояний следующего поколения. В этом множестве мы удаляем дубликаты, то есть одинаковые состояния. Оцениваем оставшиеся и сортируем в порядке ухудшения оценки. Далее выбираем N лучших, и так до тех пор, пока мы не найдем интересующее нас конечное состояние.

У лучевого поиска есть свои достоинства и недостатки. Основной его минус в том, что, в отличие от жадного, лучевой поиск не гарантирует нахождения конечного состояния с наилучшим качеством, потому что в процессе движения фронта лучшее состояние может из него выпасть. На практике с этим можно бороться, настраивая ширину фронта. Чем фронт уже, тем быстрее работает алгоритм, но тем чаще он ошибается. Чем шире фронт, тем алгоритм работает лучше, но дольше. Это одно из его важных преимуществ – возможность легко балансировать между скоростью и качеством. Лучевой поиск очень популярен в академической среде, особенно среди китайских ученых. Он прост в реализации и работает достаточно неплохо.

Идея алгоритма А\* заключается в том, что мы выбираем на каждом шаге не лучшее, а наиболее перспективное на данный момент состояние. То состояние, через которое с наибольшей вероятностью проходит путь до лучшего конечного состояния.

К оценке текущего состояния в алгоритме А\* добавляется эвристическая оценка снизу остатка пути до конечного состояния. Если эвристическая оценка снизу достаточно хороша, то А\* работает эффективно и быстро находит наилучшее состояние. В этом случае А\* будет полиноминальным по числу шагов, а не экспоненциальным. А\* — это действительно хороший алгоритм, в своей работе я использую его гораздо чаще, чем лучевой поиск.

**Методы многокритериальной оптимизации**

При проектировании сложных технических систем очень часто возникают проблемы, связанные с выбором лучшего проектного решения или для определения оптимальных параметров и характеристик проектируемой системы или создаваемого оборудования.

В тех случаях, когда при выборе лучшего варианта из полученных работоспособных вариантов выбирается тот, у которого критерий качества или эффективности на допустимом множестве параметров и характеристик оказывается больше или меньше.

В достаточно общем случае такая задача может быть сформулирована в виде задачи математического программирования следующего вида.

Требуется минимизировать (максимизировать) целевую функцию



при наличии ограничений

, ,

, .

Для решения такой задачи в настоящее время разработано достаточно большое количество методов.

Более сложные задачи возникают в тех случаях, когда критериев много и задача выбора наиболее подходящего решения принимает следующий вид:

Задача 1.

, , • • • , • • • .

при наличии ограничений

, ,

, .

Такие задачи называются задачами многокритериальной оптимизации.

Возникающая в таких задачах проблема неопределённости типична для любого технического проекта, поскольку значения многих их характеристик и параметров требуется минимизировать или максимизировать.

Например, при проектировании нового обрабатывающего оборудования естественно желание разработчиков создать оборудование, которое было бы как можно более точным, более надёжным и не особенно дорогим. Однако добиться всего этого одновременно невозможно в принципе. Реальная конструкция оказывается некоторым компромиссом. Но каким? Этого разработчики заранее не знают. В поиске наиболее подходящего компромисса и заключается проблема многокритериальности.

Математика, к сожалению, не может дать однозначного ответа на вопросы, что понимать под решением такой задачи и как получать это решение.

Такие проблемы удобно рассмотреть на примере, в котором рассматривается следующая многокритериальная задача.

Пусть требуется минимизировать критерии

, ,

которые задаются выпуклыми, непрерывно дифференцируемыми функциями. Ограничения отсутствуют.

Как известно из математического анализа, минимум каждой выпуклой, непрерывно дифференцируемой функции  (, ) при отсутствии ограничений достигается в точках, в которых

, .

Для разных функций  точки , в которых эти функции достигают минимума, в общем случае будут разными. Поэтому точек, в которых все функций  () будут достигать минимума, как правило, не существует.

Однако задачи, в которых выбор проектов, альтернатив и процессов необходимо производить с учётом многих параметров, имеют весьма важное прикладное значение и количество таких задач постоянно увеличивается.

В математике были предложены гипотезы, в которых предлагается под решением приведённой выше многокритериальной задачи считать решение следующих задач.

Эти гипотезы можно разделить на две группы. В одной группе гипотез вместо многих критериев различными способами строятся задачи с одним, специально сформированным критерием. В другой группе находится подход, предложенный Парето.

Рассмотрим некоторые методы, позволяющие заменить задачи со многими критериями задачей с одним критерием.

**1. Метод линейной свёртки критериев.**

В данном методе в качестве решения рассматриваемой выше многокритериальной задачи 1 предлагается использовать задачу с одним критерием, которая имеет следующий вид:



при наличии ограничений

, ,

, ,

где  − некоторые постоянные величины больше нуля, которые называются весовыми коэффициентами.

Весовые коэффициенты позволяют: во-первых, свести величины возможно различной физической природы  к таким величинам одной физической природы, чтобы обеспечить их сложение, а во-вторых, выделить важность каждого критерия.

Часто производится нормировка весовых коэффициентов  таким образом, чтобы сумма весовых коэффициентов  после нормировки была бы равна 1, т.е.

.

Нормированные весовые коэффициенты  вычисляются с помощью следующего соотношения:

.

Весовые коэффициенты  определяются, как правило, в результате проведения экспертизы.

В качестве решения приведённой выше многокритериальной задачи берётся решение следующей задачи с одним критерием:



при наличии ограничений

, ,

, ,

, .

Таким образом, содержание компромисса состоит в ранжировании целевых функций, которое вместе с назначением весовых коэффициентов  является дополнительной гипотезой, позволяющей свести задачу со многими критериями к задаче с единственным критерием.

**2. Метод использования контрольных показателей.**

В данном методе используется система нормативов или контрольных показателей: . Выбираемые параметры создаваемой технической системы  () должны быть больше или меньше этих нормативов или контрольных показателей, что определяется условиями решаемой задачи или конкретными условиями и требованиями создания системы.

Например, при разработке автомобиля часто требуется, чтобы расход бензина не превышал 8 или 10 литров на 100 км. Для обеспечения таких требований в математическую модель вводятся ограничения:

, ,

где величины  как раз и обозначают значения таких нормативов.

Очень часто целевая функция в таком методе представляется в виде:



и ищется такое значение вектора , при котором функция  достигает максимального значения.

Смысл такого критерия достаточно прост. При данном значении вектора  величина  определяет значение наихудшего из показателей . Значит, условие  означает выбор такой системы конструктивных параметров, которая максимизирует отношение худшего значения -го критерия к его контрольному показателю.

Если значения  не заданы, то они могут быть определены в результате экспертного опроса.

Данная модель может быть записана в виде задачи математического программирования следующим образом:

,

при наличии ограничений

, ,

, ,

, .

В качестве решения приведённой выше многокритериальной задачи берётся решение этой задачи математического программирования с одним критерием.

Неравенство  получено из приведённого выше представления целевой функции :



Если  равно минимальному значению отношения значений функций  к своим контрольным показателям , то выполняется неравенство

,

из которого и следуют эти ограничения

.

**3. Метод выделения основного критерия**

В этом методе также используется система нормативов или контрольных показателей: , которые включаются в следующие ограничения:

, .

Кроме того, среди целевых функций  выбирается некоторая основная целевая функция, например, . Тогда в качестве решения приведённой выше многокритериальной задачи берётся решение следующей задачи с одним критерием:



при наличии ограничений

, ,

, ,

, .

**4. Метод введения метрики в пространстве целевых функций**

В данном методе используется подход, в котором решается  задач математического программирования следующего вида:



при наличии ограничений

, ,

, ,

, .

Для каждой задачи  определяется вектор решения  и величина целевой функции  при этом значении вектора , которую обозначим через .

Совокупность скалярных величин  определяет в пространстве критериев некоторую тоску , которую назовём точкой «абсолютного максимума».

Для пояснения рассмотрим следующий пример с двумя критериями.







•

G

h

Для многих допустимых множеств G может не существовать такой точки  в пространстве критериев, которая была бы достижима.

Введём расстояние от такой недостижимой точки «абсолютного максимума» до множества G следующим образом:

.

Тогда в качестве решения приведённой выше многокритериальной задачи берётся решение следующей задачи с одним критерием:



при наличии ограничений

, ,

, ,

, .

Таким образом, с помощью предложенных выше методов для решения приведённой задачи многокритериальной оптимизации строятся задачи с одним критерием, решения которых берутся в качестве решения этой задачи многокритериальной оптимизации. Построенные задачи оптимизации с одним критерием являются задачами математического программирования, для решения которых разработано много разных методов.

**Метод уступок**

Одна из важнейших проблем выбора наиболее подходящей версии создаваемого объекта или изделия из имеющихся состоит в том, что обычно значения одних характеристик у какой-либо версии оказываются лучше, чем у других, а другие − хуже. Как правило, крайне редко существует такая версия, у которой все характеристики оказываются лучше, чем у других.

Для решения проблем выбора в таких случаях могут быть использованы идеи и методы принятия решений и многокритериальной оптимизации [11].

В этом случае собранные характеристики работы разных схем модернизации предприятия, из которых будет производиться выбор наиболее подходящей схемы, удобно рассматривать как критерии.

Наиболее подходящим для выбора лучшей схемы модернизируемого предприятия, на наш взгляд, является метод уступок [12].

Рассмотрим кратко одну из версий этого метод для выбора наиболее подходящей схемы модернизируемого предприятия из  проверяемых схем.

Пусть для каждой схемы   модернизируемого предприятия известны величины  характеристик. Такие характеристики собираются в описанном выше процессе проверки работоспособности схем и рассматриваются как критерии.

Эти характеристики для каждой схемы удобно представить в виде компонентов вектора, которые определяют величины критериев в порядке убывания их важности. Первая компонента этого вектора определяет величину наиболее важной характеристики схем. В таком векторе  для -й схемы , который назовём вектором критериев, -я компонента  определяет -ю () по важности характеристику рассматриваемой -й схемы.

Для каждого критерия, т.е. компоненты этого вектора, задаются величины уступок  ().

Если требуется, чтобы значение первой характеристики было меньшим, то величина уступки  прибавляется к минимальной из величин , ,…,, т.е. к , и определяется интервал .

Если требуется, чтобы значение первой характеристики было больше, то величина уступки  вычитается из величины , где , и определяется интервал .

Тогда те схемы, у которых первые компоненты вектора критериев попадают в соответствующий интервал, считаются эквивалентными и только по этим схемам продолжается дальнейшее сравнение. Остальные схемы, у которых первые компоненты вектора критериев не попадают в соответствующий интервал, из дальнейшего рассмотрения исключаются. Такая процедура повторяется для следующих компонентов оставшихся векторов  , пока не останется одна схема.