МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ (МИИТ)

Кафедра высшей математики

Р.Е. САРКИСЯН

СИСТЕМНЫЙ АНАЛИЗ И ПРИНЯТИЕ РЕШЕНИЙ

Часть 2

Детерминированные модели динамического программирования Численные методы оптимизации

Учебное пособие

Москва – 2008

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ (МИИТ)

Кафедра высшей математики

Р.Е. САРКИСЯН

СИСТЕМНЫЙ АНАЛИЗ И ПРИНЯТИЕ РЕШЕНИЙ

Часть 2

Детерминированные модели динамического программирования Численные методы оптимизации

Рекомендовано редакционно-издательским советом университета в качестве учебного пособия для специальности 22.06.02 «Управление инновациями»

Москва – 2008

УДК 005
С 20

Саркисян Р.Е. Системный анализ и принятие решений. Часть 2. Детерминированные модели динамического программирования. Численные методы оптимизации. Учебное пособие для специальности 22.06.02 - «Управление инновациями». –М.: МИИТ, 2008. -200 с.

Изложены основы многошаговых процессов и динамического программирования применительно к задачам организационного управления. Приведены также основные методы и алгоритмы численных методов оптимизации для решения задач проектирования, создания, эксплуатации и совершенствования систем и их компонентов.

Для студентов, обучающихся по специальности *«Управление инновациями».* Пособие будет полезно также преподавателям, аспирантам и научным работникам, интересующимся вопросами системотехники, прикладного системного анализа, теории исследования операций и теории управления.

Рецензенты:

Заведующий кафедрой АСУ ТП московского энергетического института (технического университета) доктор технических наук, профессор *А.В. Андрюшин;*

Президент НПО «ЭнергоНаука» доктор технических наук, профессор *Э.К. Аракелян*.

Глава 4. Детерминированные модели динамического программирования

Введение

Динамическое программирование (ДП) как вычислительный метод решения задач оптимизации, принятия решений и управления было разработано еще в 50-х гг. прошлого столетия американским ученым Ричардом Беллманом и его школой применительно к исследованию так называемых *многошаговых (многоэтапных) процессов.*

Главной вычислительной особенностью ДП является то, что с его помощью исходная оптимизационная задача (*статическая или динамическая*) подвергается *динамизации* и *«погружается»* в семейство взаимосвязанных задач меньшей размерности, решение которых и порождает оптимальное решение исходной задачи. *Динамизация* означает, что искомое решение первоначальной *n* - мерной задачи мы находим путем ее декомпозиции (разбиения) на *n* подзадач (называемых также этапами, шагами и т.д.), каждая из которых является

дномерной задачей, т.е. зависит лишь от одной переменной.

Процесс *динамизации* осуществляется в соответствии с весьма эффективным и действенным *принципом оптимальности,* служащим логической основой наилучшего продолжения оптимизируемого процесса на последующих этапах его *«естественного развития»* (или *эволюции*). Порождаемый *принципом оптимальности* вычислительный процесс имеет *рекуррентную природу*, что способствует построению *«внутренне изящных»* и *«внешне оправданных»* вычислительных алгоритмов для организации процесса поиска оптимальных решений.

В настоящее время динамическое программирование считается одним из двух современных направлений в теории оптимального управления, наряду с *принципом максимума Понтрягина,* нашедших широкое применение в задачах выработки решений и управления в технических и организационных системах. Бурное развитие ДП получило в связи с появлением быстродействующих вычислительных машин и программных средств. Несмотря на наличие у ДП мощной *операциональной поддержки*, тем не менее, ему также присуще пресловутое

«проклятие размерности» («curso of dimensionality»), суть которого заключается в том, что при большом количестве ограничений задачи возникают большие, а парой и непреодолимые трудности вычислительного характера.

Многошаговые процессы встречаются во многих важных приложениях современной науки и техники, в частности, в экономике, бизнесе, управлении организациями, военном планировании, оптимальном управлении техническими системами, в социальной сфере. Этим и объясняется популярность методов ДП и профессиональный интерес к базовым его принципам и концепции оптимизации.

4.1. Многошаговые процессы и функциональное уравнение динамического программирования

Многошаговыми называются процессы достижения целей, представляющие собой последовательность переходов из одного состояния в другое, причем эти переходы осуществляются путем принятия и реализации решений (или управлений), оптимальных по отношению к очередному шагу процесса (состоянию, этапу и т.д.).

Логическая структура многошагового процесса изображена на рис. 4.1. Процесс состоит из N этапов (шагов, стадий развития и т.д.) принятия решения и управления, обеспечивающих переход процесса (объекта, системы) из состояния S_1, в котором находится процесс в начале (начальный этап), в состояние S_N как целевое состояние. При этом предполагается, что каждый шаг (или этап) процесса характеризуется своим входом $x_к$ и выходом $x_{к+1}$, управлением (или решением) $u_к$, $к = 1,\ldots, N$, а также ведущим преобразованием

$$x_{к+1} = \varphi_к(x_к, u_к), к = 1,\ldots, N, \qquad (1.1)$$

где $\varphi_к(.,.)$ - заданные преобразования. Переменные состояния и управления $x_к$ и $u_к$, $к = 1,\ldots, N$, могут быть как скалярными, так и векторными величинами, а преобразования $\varphi_к(.,.)$ считаются непрерывно дифференцируемыми функциями аргументов $x_к$ и $u_к$, $к = 1,\ldots, N$ (текущих состояний и текущих значений управляющих параметров).

Изображенный на рисунке процесс можно интерпретировать как моделирование некоторого непрерывного во времени процесса в дискретные моменты времени t_k, $k = 1, \ldots, N$. Состояния процесса (или системы)

S_k в дискретные моменты времени $t = t_k$ задаются скалярной или векторной величиной $x_к$, а управления (или решения) в эти моменты времени задаются скалярной или векторной величиной $u_к$ для всех $к = 1, ..., N$. Соответственно, последовательность $\{x_к\}$, $к = 1, ..., N$, интерпретируются как характеристики процесса или его состояния в дискретные моменты времени $t_1, t_2, ..., t_N$. Последовательность $\{x_к, u_к\}$, $к = 1, 2, ...,$ называется *многошаговым процессом принятия решения и управления.*

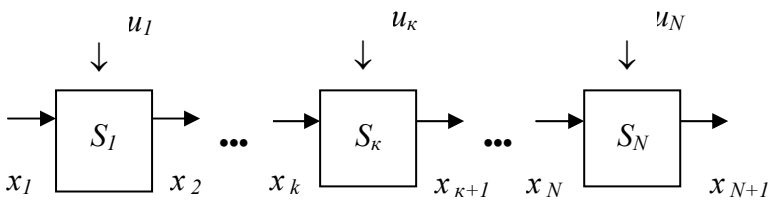

Рис 4.1. Многошаговый процесс и его состояния (этапы, стадии) $\{S_i\}$, $i = 1, ... , N$.

Перед тем, как иллюстрировать применение аппарата динамического программирования к задачам математического программирования, целесообразно

рассмотреть его связь с общей задачей оптимального управления.

Хорошо известно, что любая задача (или модель) управления состоит из следующих трех важных компонентов: *начального состояния* S_0; *целевого состояния* S_G; *средства* (или *стратегии, управления)* C для преобразования (или перевода) состояния S_0, в котором находится процесс (или система) в начальный момент времени, в целевое состояние S_G в будущем. Такую формулировку задачи управления можно представить в виде тройки ЗУ = $(S_0, S_G C)$.

В многошаговых процессах преобразование осуществляется за N шагов или этапов. Обычно средство (или стратегия) выбирается в соответствии с одним или несколькими критериями качества или эффективности, поэтому для постановки задачи оптимального управления многошаговым процессом мы должны ввести критерий (или критерии) оптимальности. Для этой цели обозначим через $R\ (u)$ критерий оптимальности процесса и представим задачу управления в виде задачи математического программирования

$$R(u) \rightarrow \max(\min), \qquad (1.2.)$$

$$\{u_к\} \in U$$

$$x_{к+1} = \varphi_к(x_к, u_к)$$

$$\{x_k\} \in X$$

где X и U - заданные пространства соответствующих размерностей. Согласно этой модели, необходимо найти такую последовательность управлений (управляющих переменных) $\{u_1, u_2, \ldots, u_N\}$, которая максимизирует или минимизирует целевую функцию $R(x, u)$, при этом переходы задаются соотношениями $x_{к+1} = \varphi_к(x_к, u_к)$, $к = 1, \ldots, N$.

Достаточные условия существования максимума или минимума целевой функции вытекают из теоремы Вейерштрасса: если функция $R(u)$ непрерывна, а множество допустимых управлений компактно (т.е. замкнуто и ограничено), тогда оптимальная последовательность управлений $\{u^*_к\}$, $к = 1, \ldots, N$, существует.

Приведенная модель многошагового процесса непосредственно вытекает из общей задачи оптимального управления, о которой вместе с общей задачей математического программирования было упомянуто еще

в первой части учебного пособия. Напомним, что под общей задачей оптимального управления подразумевается задача

$$J(u(t)) = \int_{t_0}^{t_1} \phi(x,u,t)dt + F(x^1,t_1) \rightarrow \max \quad (1.3)$$

$$\{u(t)\} \in U$$

$$dx(t)/dt = \varphi(x,u,t)$$

где t_0 и t_1 - начальный и конечный моменты времени, , $x^0 = x(t_0)$ и $x^1 = x(t_1)$ – значения вектора состояния $x(t) = (x_1(t), \ldots, x_n(t_1))^T$ в моменты времени t_0 и t_1 соответственно, $u(t) = (u_1(t), \ldots, u_m(t))^T$ – вектор управления, значения которого принадлежат заданной области U, φ, ϕ и F - заданные функции. Наличие функции F, представляющей *функцию конечного состояния,* показывает, что *целевой функционал* $J(u(t))$ зависит от конечного состояния и конечного момента времени. Когда x_o и φ заданы, управление $\{u(t)\} \in U$ однозначно определяет фазовую траекторию $\{x(t)\}$, поэтому целевой функционал $J(u(t))$ является функцией управлений $\{u(t)\} \in U$.

Решение задачи (1.3) с помощью динамического программирования будет обсуждаться в разделе 4.5. Здесь же мы покажем, что если подвергнуть ее дискретизации и ограничиться изучением поведения объекта лишь в

дискретные моменты времени $t_o, t_o + \Delta, t_o + 2\Delta, ..., t_o + N\Delta = t_1$, тогда, вместо (1.3), получим обычную задачу математического программирования. Для этой цели рассмотрим дискретные моменты времени $k = 0,1,..., N$, соответствующие моментам $t_o, t_o + 1\Delta, t_o + 2\Delta, ..., t_o + N\Delta = t_1$. Состояние объекта в каждый дискретный момент времени $\kappa, \kappa = 0,1,..., N$, описывается вектором x^κ, а управление – вектором u^κ. Согласно (1.1), состояние объекта в следующий момент времени будет описываться соотношением

$$x^{\kappa+1} = \varphi_\kappa(x^\kappa, u^\kappa), \kappa = 0,1,..., N, \qquad (1.4)$$

а целевой функционал в (1.3) примет вид

$$J(\{u^k\}) = \sum_{\kappa=0}^{N} \phi_\kappa(x^\kappa, u^\kappa) + F(x^N, N). \qquad (1..5)$$

Выражение (1.4) непосредственно можно получить из дифференциального уравнения *dx(t)/dt = φ(x,u,t)* как его конечно-разностный аналог. Задача (1.3) в дискретной постановке уже приобретает форму обычной задачи математического программирования

$$J(\{u^\kappa\}) = \sum_{k=0}^{N} \phi_\kappa(x^\kappa, u^k) + F(x_1, t_1) \to \max_{\{u^k\}}, \qquad (1.6)$$

$$\{u^k\} \in U$$

$$x^{k+1} = \varphi_\kappa(x^k, u^k)$$

$$k = 0, 1, ..., N - 1$$

Форма задачи (1.6) полностью совпадает с задачей (1.2), в которой максимизация или минимизация также производится по вектору управления $\{u^k\} \in U$, который при заданной функции $\varphi_\kappa(x_\kappa, u_\kappa)$ однозначно определяет переменные $\{x_k\} \in X$. Обратный переход к задаче (1.3), очевидно, можно осуществить путем стремления N к бесконечности, что эквивалентно условию $\Delta = (t_1 - t_0)/N \to 0$. Благодаря этому переходу непрерывную задачу управления называют *задачей оптимизации в бесконечномерном пространстве*.

В постановке задачи (1.6) цель управления заключается в определении последовательности управляющих векторов (или решений) $u^o, u^1, ..., u^N$, которые принадлежат заданной области управления U, т.е. $u^\kappa \in U$, $\kappa = 0,1,..., N$, и максимизируют целевую функцию $J(\{u^\kappa\})$. Очевидно, что сформулированная таким образом задача аналогична задаче оптимального

управления (1.3). Она, разумеется, соответствует управлению многошаговым процессом, изображенным на рис. 4.1 с точностью до нумерации индексов (или шагов). Величина x^o и x^N считаются фиксированными, кроме того, x^N также подчиняется уравнению (1.4).

Максимизация функции $J(\{u^\kappa\})$ при ограничениях $x^{k+1} = \varphi_\kappa(x^k, u^k)$ и $u^\kappa \in U$, $\kappa = 0,1,...,N$, - это обычная задача математического программирования, записанная в несколько специфической форме. Она обладает свойством сепарабельности, что позволяет вместе с принципом оптимальности строить основное *рекуррентное соотношение – функциональное уравнение динамического программирования (ФУДП),* описывающее процесс оптимального перехода из фиксированного начального состояния (x^o, t_o) в фиксированное конечное (или целевое) состояние (x^N, t_1). Достаточные условия существования оптимального управления вновь непосредственно вытекают из основной задачи математического программирования – теоремы Вейерштрасса.

Подход динамической оптимизации заключается в том, что для задачи (1.3) вводится *последовательность*

функций оптимального поведения {f(x, k)} и каждая функция $f(x,k)$ определяется как оптимальное значение функционала *J(u(t)),* соответствующее начальному моменту времени k и начальному состоянию x. С помощью этих функций процесс перехода описывается *рекуррентным соотношением*

$$f(x,\kappa) = \max_{u^\kappa \in U}\left\{\phi_\kappa(x^\kappa, u^\kappa) + f(x^{\kappa+1}, \kappa+1)\right\}, \qquad (1.7)$$

представляющее собой *функциональное уравнение динамического программирования (ФУДП).* Уравнение (1.7) означает, что оптимальное значение целевой функции *J(u(t))* в задаче с начальным состоянием x и начальным временем t равно оптимальному значению суммы двух слагаемых: функции $\phi_\kappa(x^\kappa, u^\kappa)$ в момент $t = \kappa$ и оптимального значения $f(x^{\kappa+1}, \kappa+1)$ в момент времени $t = \kappa + 1$. Уравнение отвечает граничному условию $f(x^N, t_1) = F(x^{t_1}, t_1)$, согласно которому оптимальное значение целевой функции *J(u(t))* в задаче с начальными параметрами (x^{t_1}, t_1) совпадает со значением функции конечных параметров $F(x^{t_1}, t_1)$ (мы намеренно приняли обозначение x^{t_1}, чтобы не путать с вектором x^1 ряда

$(x^o, x^1, ...)$). С учетом (1.4) это уравнение можно представить в окончательной форме

$$f(x,\kappa) = \max_{u^\kappa \in U} \left\{ \phi_\kappa(x^\kappa, u^\kappa) + f(\varphi_\kappa(x^\kappa, u^\kappa), \kappa+1) \right\}. \qquad (1.8)$$

В этой задаче форма обозначения $f(x,\kappa) = f_\kappa(x)$ не является принципиальной.

В процессе динамизации возникает класс задач, определяемых рядом параметров, и вслед за этим на основе принципа оптимальности определяется основное *рекуррентное соотношение – функциональное уравнение динамического программирования*. В качестве параметров многошагового процесса можно взять начальный момент и начальное значение вектора состояния x^0. Альтернативным является выбор в качестве параметров начальное состояние и промежуток времени $\tau = t_1 - t$ (или дискретное время $\tau = N - k$).

Отметим, что введение последовательности функций оптимального поведения *{f(x, t)}* с начальным моментом времени t и начальным состоянием x является ключевым моментом в *динамизации* задачи. Каждая из них характеризует оптимальное значение целевого функционала многошаговой задачи с начальными

значениями t и x, т.е. $f(x, t) = J^*(x, t)$. Оптимальное значение целевого функционала и функции оптимального поведения, очевидно, будут равны $f(x_0, t_0) = J^*(x_0, t_0)$. При $t = t_1 = t_0 + N\Delta$ выполняется граничное условие $f(x_1, t_1) = J^*(x_1, t_1) = F(x_1, t_1)$, как это следует из (1.3) или (1.5).

Эквивалентный механизм *динамизации* задачи (1.6) и ее «*погружения*» в семейство взаимосвязанных, но более простых задач, также основанный на семействе функций оптимального поведения *{f(x, k)}*, связан с определением каждой функции $f(x, k)$ из задачи

$$f(x, k) = \max_{(u^0, u^1, ..., u^k)} \{ \sum_{i=0}^{k} \phi_i(x^i, u^i) \}, \qquad (1.9)$$

$$x^{i+1} = \varphi_i(x^i, u^i)$$

$$i = 0, 1, ..., k-1$$

$$\{u^i\} \in U$$

для всех $\kappa = 0,1,..., N$. При $\kappa = N$, т.е. $t = t_1$, имет место $x = x^1$.

Благодаря свойству сепарабельности функций $\phi_i(x^i, u^i)$ и $\varphi_i(x^i, u^i)$, эту задачу можно представить в виде

$$f(x, k) = \max_{u^k} \{ \phi_\kappa(x^\kappa, u^\kappa) + \max_{(u^0, u^1, ..., u^{k-1})} \sum_{i=0}^{k-1} \phi_i(x^i, u^i) \}$$

$$x^{i+1} = \varphi_i (x^i, u^i)$$
$$i = 0, 1, \ldots, k-1$$
$$\{u^i\} \in U \qquad (1.10)$$

По определению, вторая слагаемая правой части этого выражения есть функция $f(x, k-1)$, подставляя которую в (1.6), получим

$$f(x,\kappa) = \max_{u^\kappa \in U} \left\{ \phi_\kappa (x^\kappa, u^\kappa) + f(x^\kappa, \kappa - 1) \right\}. \quad (1.11)$$
$$x^\kappa = \varphi_{\kappa-1} (x^{\kappa-1}, u^{\kappa-1})$$

Это уравнение представляет для задачи (1.6) *функциональное уравнение динамического программирования,* описывающее многошаговый процесс принятия решения и управления для моментов времени $k = 1, \ldots, N$.

Соотношения (1.8) и (1.11) показывают, что применение аппарата динамического программирования к чисто *«статическим»* задачам типа (1.2) или (1.6) позволяет *динамизировать* их, превратив решение задачи в динамический процесс перехода из начального состояния в конечное (целевое) состояние. Другими словами, динамическое программирование создает возможность поэтапного решения исходной задачи, рассматривая каждый этап общего процесса с точки

зрения определения наилучшей альтернативы его развития (или продолжения).

Механизм динамизации, основанный на свойстве сепарабельности целевой функции, применим и к задачам, в которых целевая функция имеет мультипликативную форму, т. е. к задачам типа

$$R(x,u) = \prod_{k=1}^{N} r_k (x_k, u_k) \to \max. \qquad (1.12)$$
$$(u_1, u_2, \ldots, u_N)$$
$$x_{\kappa+1} = \varphi_\kappa(x_\kappa, u_\kappa), \ \forall \kappa$$
$$u_\kappa \in U, \ \forall \kappa$$

Динамизация этой задачи с помощью функций $f(x, \kappa)$ происходит аналогичным образом, и в результате мы получает последовательность задач

$$f(x, \kappa) = \max_{u_k} \{ r_k(x_k, u_k) \cdot \max_{(u_1,\ldots,u_{k-1})} \prod_{i=1}^{k-1} r_i (x_i, u_i) \} =$$

$$= \max_{u_k} \{ r_k(x_k, u_k) f(x, \kappa - 1) \}, \qquad (1.13)$$

для всех $\kappa = 1, \ldots, N$ и начального условия $f(x, 0) = 1$. В (1.13), по определению функций $f(x, \kappa)$, принято обозначение

$$f(x, к - 1) = \max_{(u_1,\ldots,u_{k-1})} \prod_{i=1}^{k-1} r_i(x_i, u_i). \qquad (1.14)$$

Рекуррентное соотношение (1.13) возникает, например, при решении задачи повышения надежности системы путем резервирования при заданных ограничениях на стоимость и вес. Она будет рассмотрена в разделе 4.3.

Процесс динамической оптимизации и перехода из начального состояния в целевое состояние подчиняется следующим ведущим принципам:

а) *принцип погружения* - исходная задача «*погружается»* в семейство оптимизационных задач, и для каждого этапа (шага, состояния) решается своя оптимальная подзадача. Процесс *динамизации* и погружения эквивалентен введению *пространства состояний* (или множества состояний);

б) множество решений оптимизационных подзадач описывается *функциональным уравнением динамического программирования (ФУДП);*

в) решение множества оптизационных подзадач можно найти с помощью *алгоритма обратной прогонки (хода),*

который равнозначен упорядоченной процедуре решения последовательности функциональных уравнений;

г) *принцип оптимальности* - оптимальная стратегия обладает тем свойством, что *каковы бы ни были начальное состояние и решения, последующие решения образуют оптимальную стратегию для того состояния, которое возникает в результате предыдущих переходов*. Из этого принципа следует, что всегда существует стратегия (или решение), которая оптимальна для данного состояния (этапа, стадии), и что каждый этап *оптимальной траектории* (или *пути)* сам является оптимальным.

Как правило, целевая функция задачи (1.2) имеет специфическую структуру, а именно, она либо аддитивна, либо мультипликативна, причем как составляющие функции цели, так и функции, характеризующие ограничения задачи зависят лишь от *«своих»* переменных (переменных лишь данного шага, этапа и т.д.). Это свойство известно как *свойство сепарабельности.* Оно способствует упрощению и эффективной организации процесса решения оптимизируемой задачи. В соответствии с этим свойством и различают *сепарабельные задачи, сепарабельное программирование.*

Применение свойства *сепарабельности* целевой функции и идеи *динамизации (динамической оптимизации)* на основе *принципа оптимальности* будет в этой главе иллюстрировано на примерах решения классических задач организационного управления - *задачи оптимального использования ограниченных ресурсов, инвестирования, резервирования, транспортировки грузов*, и других. В конце главы будут отмечены важные аналитические связи динамического программирования с классическим вариационным исчислением и оптимальным управлением.

4.2. Задача инвестирования

Одна из распространенных постановок этой задачи такова: коммерческой фирме необходимо сформулировать свою стратегию размещения наличных денежных средств заданной величины C в N проекты (производственные или инновационные программы, ценные бумаги, и т.д.), которая принесет фирме наибольший суммарный доход от реализации всех проектов (так называемая задача финансирования инвестиционного процесса).

Переменными задачи служат объемы инвестиций x_j в j - ю программу, ожидаемый доход от которой описывается заданной функцией $r_j(x_j)$, $j = 1, \ldots, N$; ограничениями являются $x_1, \ldots, x_N \geq 0$, $x_1 + \ldots + x_N \leq C$, а в качестве целевой функции можно взять суммарный доход $r_1(x_1) + \ldots + r_N(x_N)$. Предполагается, что величина дохода каждой программы $r_j(x_j)$ зависит только от величины x_j, причем имеет место $r_j(0) = 0$, $j = 1, \ldots, N$, и при увеличении объема инвестиции ожидаемый доход постепенно *«насыщается»*, т.е. наступает эффект убывания отдачи вложенных средств. Его можно описать с помощью условий $\partial r / \partial x \succ 0$, $\partial^2 r / \partial x^2 \prec 0$.

Обозначив через $R(x)$ функцию суммарного дохода фирмы, можно сформулировать задачу нахождения оптимальной стратегии инвестирования в виде

$$R(x) = \sum_{j=1}^{N} r_j(x_j) \to \max . \qquad (2.1)$$

$$(x_1, \ldots, x_N) \geq 0$$

$$\sum_{j=1}^{N} x_j \leq C$$

Для построения алгоритма решений этой (*чисто статической)* задачи воспользуемся сперва свойством сепарабельности ее целевой функции и ограничений, и путем введения множества состояний S_k, величины текущего ресурса $z \le C$ и семейства функций $\{f_к(z)\}$, $k = 1, \dots, N$, сформулируем задачи

$$f_к(z) = \max_{\substack{(x_1,..,x_к) \\ \sum_{j=1}^{к}(x_j) \le z}} \sum_{j=1}^{к} r_j(x_j) \qquad (2.2)$$

для всех $k = 1, \dots, N$, и $0 \le z \le C$. На основе свойства сепарабельности (2.2) можно представить в виде

$$f_к(z) = \max_{\substack{0 \le x_к \le z \\ (x_1,..,x_{к-1}) \\ \sum_{j=1}^{к-1} x_j \le z - x_к}} \{r_k(x_k) + max \sum_{j=1}^{к-1} r_j(x_j)\} =$$

$$= \max_{0 \le x_к \le z} \{r_к(x_к) + f_{к-1}(z - x_k)\}, \qquad (2.3)$$

где по определению семейства функций $\{f_к(z)\}$, $k = 1, \dots, N$, принято обозначено

$$f_{к-1}(z - x_к) = \max_{\substack{(x_1,..,x_{к-1}) \\ \sum_{j=1}^{к-1} x_j \le z - x_к}} \sum_{j=1}^{к-1} r_j(x_j). \qquad (2.4)$$

Соотношение (2.3) представляет собой функциональное уравнение динамического программирования для инвестиционной задачи, полученное на основе свойства сепарабельности. Оно описывает N – шаговый процесс оптимизации и принятия решений. Функция оптимального поведения $f_k(z)$, как видно из (2.3), зависит от текущей величины объема инвестиций в первые $к$ программ; когда $к$ достигает значения N и, следовательно, текущий ресурс z будет равен C, из (2.2) получим величину $f_N(C)$, равную оптимальному значению целевой функции $R(x)$, то есть $f_N(C) = R^* = R(x^*)$, где $x^* = (x^*_1, \dots, x^*_N)^T$ - оптимальная стратегия инвестирования.

Прежде чем построить на основе рекуррентного уравнения (2,2) алгоритм решения исходной задачи, рассмотрим альтернативный путь получения

функционального уравнения динамического программирования, опираясь лишь на упомянутый выше *принцип оптимальности.*

Рассмотрим произвольное состояние S_k, которое характеризует процесс инвестирования некоторого ресурса $z,\ 0 \le z \le C,$ в первые $к$ проекты, причем, $k = 1, 2\ldots., N,$ - произвольный шаг. Выделим из этого объема ресурсов некоторое количество $x_к$ (пока неизвестное) для инвестирования только в последнюю, k -ю программу с уровнем дохода $r_k(x_k)$ и потребуем, чтобы оставшиеся ресурсы объема $z - x_к$ были наилучшим образом (т.е. оптимально) инвестированы в первые $к - 1$ программы. Обозначив оптимальный доход от первых $к - 1$ программ через $f_{к-1}(z - x_к),$ для величины суммарного дохода данного состояния S_k получим выражение

$$r_к(x_к) + f_{к-1}(z - x_к). \qquad (2.5)$$

При фиксированном значении z эта величина зависит только от величины $x_к$, следовательно, определив величину $x_к$ из условия максимума функции состояния (2,4) и обозначив результат через $f_к\ (z)$, получим

рекуррентное уравнение динамического программирования

$$f_к(z) = \max\{r_к(x_к) + f_{к-1}(z - x_к)\}, \qquad (2.6)$$
$$0 \le x_к \le z$$

справедливое для всех $k = 1, 2\ldots., N$, и $z,\ 0 \le z \le C$. Оно, разумеется, полностью совпадает с функциональным уравнением (2.2). Таким образом, как принцип оптимальности, так и свойство сепарабельности порождают одно и тоже функциональное уравнения (2.6). Это рекуррентное уравнение служит правилом принятия оптимального решения и порождает соответствующий алгоритм нахождения оптимальной стратегии инвестирования $x^* = (x^*_1, \ldots, x^*_N)^T$ при заданных значениях C и N.

Функциональное уравнение (2.6) порождает простой и, вместе с тем, весьма эффективный алгоритм решения, состоящий из двух этапов - прямого и обратного хода вычислений.

Прямой ход алгоритма.

Шаг 1. Положить $к = 1, f_o(z) = 0$ для всех $z = 0, 1, \ldots, C,$ вычислить функцию

$$f_1(z) = \max\{r_1(x_1) + 0\},$$

$$0 \le x_1 \le z$$

для всех значений $z = 0, 1, \ldots, C$, и сохранить результаты в виде таблицы T_1, содержащей массив данных z, $f_1(z)$ и $x_1(z)$ для всех $z = 0,1,\ldots,C$. Общая форма этих таблиц T_1 приведена ниже в виде таблицы T_k.

Шаг 2. Положить $k = 2$, используя массив $f_1(z)$ предыдущего шага, вычислить функцию

$$f_2(z) = \max\{r_2(x_2) + f_1(z - x_2)\}$$

$$0 \le x_2 \le z$$

для всех значений $z = 0, 1, \ldots, C$, и сохранить результаты в виде таблицы T_2 со значениями z, $f_2(z)$ и $x_2(z)$, $z = 0, 1, \ldots, C$, аналогичной таблице T_k, и т.д.

Шаг к. Используя массив $f_{к-1}(z)$ предыдущего, $к - 1$ –ого шага, вычислить функцию

$$f_к(z) = \max\{r_к(x_к) + f_{к-1}(z - x_к)\}$$

$$0 \le x_к \le z$$

для всех $z = 0, 1, \ldots, C$, и сохранить результаты в виде таблицы T_k.

Таблица $Tк$

Z	$f_к(z)$	$x_к(z)$
0	$f_к(0)$	$x_к(0)$
1	$f_к(1)$	$x_к(1)$
.	.	.
.	.	.
.	.	.
C	$f_к(C)$	$x_к(C)$

и т.д., до последнего шага.

Шаг N. Положить $к = N$, $z = C$, вычислить функцию

$$f_N(C) = \max\{r_N(x_N) + f_{N-1}(C - x_n)\}$$

$$0 \le x_N \le C$$

и сохранить результаты в виде таблицы T_N:

Таблица T_N

z	$f_N(z)$	$x_N(z)$
C	$f_N(C)$	$x_N(C)$

Этим шагом завершается прямой ход (прогон) алгоритма, а найденные оптимальные результаты уже

содержатся в таблицах T_k, $k = 1, ..., N$. Чтобы их вывести, воспользуемся обратным ходом алгоритма. Необходимые для этого действия сводятся к следующему.

Согласно принципу оптимальности, результаты последнего, N *-ого* шага алгоритма составляют оптимальную стратегию инвестирования в последнюю программу, т.е. $x^*_N = x_N(C)$ и $R^* = R(x^*) = f_N(C)$. Чтобы вывести остальные оптимальные решения $x^*_к$, $к = N - 1, N - 2, ..., 1$, обрабатываем заполненные таблицы в обратной последовательности.

Полагая $к = N$, $z = C$, выводим содержимое таблицы T_N, т.е. оптимальные значения $f_N(C)$ и $x_N(C) = x^*_N$. После того как N –ому проекту уже выделен ресурс величины x^*_N, оставшийся ресурс величины $C - x^*_N$, согласно принципу оптимальности, должен быть оптимальным образом инвестирован в первые $N - 1$ программы. Следовательно, полагая $к = N - 1$, в таблице T_{N-1} находим оптимальное решение x^*_{N-1} в строке, уровень текущего ресурса которой равен величине $z = C - x^*_N$, так что $x^*_{N-1} = x_{N-1}(C - x^*_N)$. Значение функции $f_{N-1}(C - x^*_N)$ при этом показывает оптимальный уровень дохода от первых $N - 1$ программ. Далее, полагая $к = N - 2$, в таблице T_{N-2}

находим оптимальное решение x^*_{N-2} в строке, соответствующей уровню текущего ресурса $z = C - x^*_N - x^*_{N-1}$, то есть $x^*_{N-2} = x_{N-2}(z)$, а величина $f_{N-2}(z)$ будет характеризовать оптимальный уровень дохода от инвестиций в первые $N - 2$ программ, и т.д. На последнем шаге обратного хода алгоритма находим оптимальное решение $x^*_1 = x_1(z)$, соответствующее уровню ресурса $z = C - x^*_N - x^*_{N-1} - ... - x^*_2$, и величину $f_1(z) = r_1(x^*_1)$ как оптимальный уровень дохода от первой программы.

Таким образом, прямой ход алгоритма динамического программирования связан с вычислением и формированием таблиц T_k со значениями $\{z, f_к(z), x_к(z)\}$, $z = 0, 1, ..., C$, $к = 1, 2, ..., N$, а обратный ход алгоритма позволяет вывести оптимальные решения $\{x_к(z), f_к(z)\}$, для всех значений текущего ресурса $z = C$, $C - x^*_N$, $C - x^*_N - x^*_{N-1}, ..., C - x^*_N - x^*_{N-1} - ... - x^*_2$. Обратный ход алгоритма можно выполнить с помощью следующего подалгоритма

шаг 1. Положить $к = N$, $z = C$;

шаг 2. Вывести $f_к(z)$, $x_к(z)$;

шаг 3. Положить $z := z - x_к(z)$;

шаг 4. Положить $к := к - 1$;

шаг 5. Если к ≥ 1, перейти к шагу 2

шаг 6. Остановится.

Результаты решения, разумеется, должны удовлетворить условиям

$$\sum_{j=1}^{N} x_j^* \le C,\ f_N(C) = \sum_{j=1}^{N} r_j(x_j^*). \qquad (2.7)$$

Это и есть процесс динамизации (чисто *статической)* задачи математического программирования и ее *«погружение»* в семейство взаимосвязанных одномерных задач. На каждом шаге рассматривается совместное инвестирование средств в первые *к* программы, *к = 1, 2,..., N,* оптимальным образом, т.е. в соответствии с принципом оптимальности. На первом шаге участвует лишь первая программа с функцией дохода $r_1(x_1)$; на втором шаге участвуют первые две программы с функциями $r_2(x_2)$ и $f_1(z)$ и т.д.; на произвольном *к* –м шаге в процесс инвестирования вовлечены первые *к* программы с функциями $r_к(x_к)$ и $f_{к-1}(z)$, при этом функция $f_{к-1}(z)$ характеризует оптимальное поведение процесса до текущего шага, т.е. оптимальную предысторию. На последнем *N* –м шаге в процессе инвестирования

присутствуют все программы, а состояние характеризуется функциями $r_N(x_N)$ и $f_{N-1}(z)$, где $f_{N-1}(z)$ – оптимальный уровень дохода от первых $N-1$ программ.

В книге Б. Кузнецова *«Жизнь, смерть, бессмертие»,* посвященной Альберту Эйнштейну, называются два критерия – *внутренняя изящность и внешнее оправдание,* которые по мнению автора теории относительности следует предъявить к любой научной теории. Без преувеличения можно сказать, что алгоритмы динамического программирования, порождаемые рекуррентными соотношениями (ФУДП), также удовлетворяют этим двум критериям, а разработка и реализация соответствующих вычислительных программ на ЭВМ может служить признаком высокого мастерства и умения имитировать (или воспроизвести) сложные и взаимосвязанные процессы и явления с учетом присущих им внутренних и внешних отношений.

В рассмотренной задаче целевая функция *R(x)* имела аддитивную форму, т.е. $R(x) = r_1(x) + ... + r_N(x_N)$. В следующем разделе мы рассмотрим задачу с так называемой мультипликативной целевой функцией, т.е. $R(x) = r_1(x) \cdot ... \cdot r_N(x_N)$.

4.3. Задача оптимального резервирования

Содержательная постановка этой задачи и ее математическая модель были рассмотрены еще в разделе 3.1 нелинейного программирования первой части пособия. Приведем ее математическую постановку задачи:

$$R(m) = \prod_{j=1}^{N} r_j(m_j) \to \max. \qquad (3.1)$$

$$(m_1, \ldots, m_N)$$

$$\sum_{j=1}^{N} c_j m_j \le C$$

$$\sum_{j=1}^{N} w_j m_j \le W$$

$m_j, j = 1, \ldots, N,$ - целевые

Напомним, что в этой задаче рассматривается проблема повышения надежности некоторой системы (или устройства), состоящей из N последовательно соединенных звеньев, путем резервирования каждого ее звена, m_j, $= 0, 1, 2, \ldots,$ дополнительными (резервными) элементами при ограничении на общий вес и стоимость резервирования. Таким образом, переменные m_j, $j = 1, \ldots, N,$ этой задачи характеризуют количество

резервированных элементов звеньев, параметры c_j, w_j характеризуют стоимость и вес элементов соответственно; $r_j(m_j) = 1 - (1 - p_j)^{1+m_j}$, $j = 1, \ldots, N,$ – функция надежности j-ого звена; p_j - вероятность безотказной работы элементов (как основных, так и резервных), C и W – общая стоимость и общий вес резервирования. Предполагается, что при отказе основного элемента звена автоматически подключается один из резервных элементов. Целевая функция $R(m)$ представляет собой функцию надежности системы из N последовательных звеньев. Стратегия резервирования заключается в определении таких значений $m_1^*, \ldots, m_N^*$, которые удовлетворяют ограничениям по стоимости и весу и максимизируют целевую функцию $R(m)$.

Рассмотрим вначале решение этой задачи при выполнении ограничения по стоимости. Как и выше, введем в рассмотрение последовательность функций $\{f_к(z)\}$ и определим каждую из них как оптимальную величину функции надежности первых $к$ последовательно соединенных звеньев при фиксированном уровне ресурса по стоимости z. Пусть $S_к$ характеризует это состояние с

числом звеньев $к$ и текущим уровнем ресурса z ($0 \le z \le C$). По определению, функцию $f_к(z)$ находим из задачи

$$f_к(z) = \max_{(m_1,\ldots,m_к)} \prod_{j=1}^{к} r_j(m_j) \qquad (3,2)$$

$$\sum_{j=1}^{к} c_j m_j \le z$$

m_j, $j=1, \ldots, K$, - *целевые*

для всех $к = 1,\ldots, N,$ и $z = 0, 1,\ldots, C.$ Построим функциональное уравнение на основе принципа оптимальности. Для этой цели выделим $m_к$ резервных элементов последнему, к -му звену, потратив на эту операцию $c_к m_к$ единиц ресурса по стоимости. Надежность этого звена будет равна величине $r_к(m_к)$. Согласно принципу оптимальности, оставшиеся ресурсы в объеме $z - c_к m_к$ должны быть использованы для оптимального резервирования остальных $к - 1$ звеньев. Обозначив функцию надежности этих $к - 1$ звеньев для оптимальной стратегии резервирования через $f_{к-1}(z - c_к m_к)$, для функции надежности последовательного соединения первых $к$ звеньев получим выражение

$$r_к(m_к) f_{к-1}(z - c_k m_к). \qquad (3.3.)$$

Оно, как видно, зависит лишь от величины $m_к$, следовательно, максимизировав его по $m_к$ и обозначив результат максимизации через $f_к(z)$, получим рекуррентное соотношение

$$f_к(z) = \max \{ r_к(m_к) f_{к-1}(z - c_к m_к) \}, \qquad (3,4)$$

$$0 \le m_к \le \lfloor z / c_k \rfloor$$

где $\lfloor z / c_k \rfloor$ - целая часть отношения z/c_k. Выражение (3.4) представляет собой функциональное уравнение динамического программирования, описывающее процесс оптимального решения задачи.

Рекомендуем студентам самостоятельно вывести соотношение (3.4), исходя из свойства сепарабельности целевой функции $R(x)$ и функции ограничения $\sum_{j=1}^{к} c_j m_j \le z$.

Функциональное уравнение (3.4) порождает следующий вычислительный алгоритм.

Шаг 1. Положить $k =1$, $f_o(z) =1$ для всех $z = 0, 1,\ldots, C,$ вычислить функцию

$$f_1(z) = \max \{ r_1(m_1) \}$$

$$0 \le m_1 \le \lfloor z / c_1 \rfloor$$

для всех $z = 0, 1, \ldots, C$, и сохранить результаты в виде таблицы T_1, содержащей данные $(z, f_1(z), m,(z)), z = 0, 1, \ldots, C)$.

Шаг 2. Положить $к = 2$, используя массив $f_1(z)$ предыдущего шага, вычислить значение функции

$$f_2(z) = \max \left\{ r_2(m_2) f_1(z - c_2 m_2) \right\}$$

$$0 \le m_2 \le |z / c_2|$$

для всех $z = 0, 1, \ldots, C$, и сохранить результаты в виде таблицы T_2 с данными $(z, f_2(z), m,(z)), z = 0, 1, \ldots, C)$, *и* так далее.

Шаг к. Используя массив $f_{к-1}(z)$ предыдущего шага, вычислить значение функции

$$f_к(z) = \max \left\{ r_к(m_к) f_{к-1}(z - c_к m_к) \right\}$$

$$0 \le m_k \le |z / c_k|$$

для всех $z = 0, 1, \ldots, C$, и сохранить результаты в виде таблицы $T_к$.

Таблица *Тк*

Z	$f_к(z)$	$m_к(z)$
0	$f_к(0)$	$x_к(0)$
1	$f_к(1)$	$x_к(1)$

.	.	.
.	.	.
.	.	.
C	$f_к(C)$	$x_к(C)$

и т.д., до последнего шага.

Шаг N. Положить $к = N, z = C$ и, используя массив $f_{N-1}(z)$ предыдущего шага, вычислить значение функции

$$f_N(z) = \max \left\{ r_N(m_N) f_{N-1}(z - c_N m_N) \right\},$$

$$0 \le m_N \le |z / c_N|$$

сохранив результаты в виде таблицы T_N, имеющей вид

Таблица T_N

Z	$f_N(z)$	$m_N(z)$
C	$f_N(C)$	$m_N(C)$

Полученные на последнем шаге результаты $m_N(C)$ *и* $f_N(C)$ определяют оптимальную величину m^*_N и $R^* = R(m^*)$. Этим завершается прямой ход алгоритма. Его обратный ход (или обратный прогон) аналогичен предыдущей задаче. Полагая $к = N, z = C$, из таблицы T_N выводим оптимальные величины $f_N(C), m_N(C) = m^*_N$. После того как

N -ому звену выделен ресурс, равный $c_N\ m^*_N$, для оптимального резервирования остальных *N - 1* звеньев остается ресурс величины $C - c_N\ m^*_N$. Следовательно, в таблице T_{N-1} находим оптимальное решение $m^*_{N-1} = m_{N-1}(z)$ в строке, которая соответствует текущему ресурсу $z = C - c_N\ m^*_N$. В этой же строке находим величину $f_{N-1}(C - c_N\ m^*_N)$, равную оптимальной величине функции надежности первых *N - 1* звеньев. Продолжая обработку остальных таблиц в обратной последовательности, находим остальные оптимальные решения.

Так, например, в таблице $T_к$ находим оптимальное решение $m^*_к = m_к(z)$, соответствующее уровню ресурсов $z = C - c_N\ m^*_N - \ldots - c_{к+1}\ m^*_{к+1}$, и оптимальное значение функции надежности последовательного соединения первых *к* звеньев $f_к(z)$. Последняя таблица T_1 содержит оптимальное решение $m^*_1 = m_1(z)$ в стоке с уровнем текущего ресурса $z = C - c_N\ m^*_N - \ldots - c_2\ m^*_2$, а также величину $f_1(z) = r_1(m^*_1)$. Процесс обработки таблиц $T_к$, $к = N, N-1, \ldots, 1$, обратным ходом можно выполнить с помощью следующего подалгоритма, аналогичного приведенному выше:

шаг 1. Положить $к = N,\ z = C$;

шаг 2. Вывести $f_к(z), m_к(z)$;

шаг 3. Положить $z: = z - c_к\ m_к(z)$;

шаг 4. Положить $к: = к - 1$;

шаг 5. Если $к \geq 1$, перейти к шагу 2;

шаг 6. Остановиться.

Полученные результаты должны удовлетворить условиям

$$\sum_{j=1}^{N} c_j m_j^* \leq C,\ \ f_N(C) = \prod_{j=1}^{N} r_j(m_j^*). \qquad (3.5)$$

Решение задачи резервирования с учетом ограничений по стоимости и весу можно осуществить с помощью метода Лагранжа. Для этой цели выберем одно из ограничений, например, ограничение по весу $\sum_{j-1}^{N} \omega_j m_j \leq W$, для включения в состав целевой функции. Целевая функция в условиях задачи имеет мультипликативную форму, т.е. $R(m) = \prod_{j=1}^{N} r_j(m_j)$, поэтому предварительно прологарифмируем ее и составим функцию Лагранжа в форме

$$\ln L(m, \lambda) = \ln R(m) + \lambda(W - \sum_{j=1}^{N} w_j m_j) =$$

$$= \sum_{j=1}^{N} (\ln r_j(m_j) - \lambda w_j m_j) + \lambda W. \qquad (3.6)$$

В правой части этого выражения слагаемое λW можно исключить из процесса оптимизации, так как оно не влияет на искомое решение m^*_j, $j = 1,..., N$. Тогда из (3.6) получим после потенцирования выражение для функции Лагранжа в виде

$$L(m,\lambda) = \prod_{j=1}^{N} e^{-\lambda w_j m_j} r_j(m_j). \qquad (3.7)$$

Теперь новая задача оптимального резервирования примет форму

$$L(m,\lambda) \to \max_{(m_1,...,m_N)}. \qquad (3.8)$$

$$\sum_{j=1}^{N} c_j m_j \le C$$

$$m_j \ge 0, \forall j - \textit{целые}$$

Как и в случае задачи (3.1), эту задачу можно подвергнуть динамизации путем введения последовательности функций $\{f_k(z)\}$, текущих параметров k и z, и определения функций $f_k(z)$ из задачи

$$f_к(z) = \max_{0 \le m_к [z/C_к]} \left\{ e^{-\lambda w_к m_к} r_к(m_к) f_{к-1}(z - c_к m_к) \right\}, \qquad (3.9)$$

где по определению функций $\{f_k(z)\}$ обозначено

$$f_{к-1}(z - c_к m_к) = \max_{(m_1,...,m_{к-1})} \prod_{j=1}^{к-1} e^{-\lambda w_j m_j} r_j(m_j). \qquad (3.10)$$

$$\sum_{j=1}^{k-1} c_j m_j \le z - c_к m_к$$

$$m_j \ge 0 \forall j, - \textit{целые}$$

Рекуррентное уравнение (3.9) порождает такой же вычислительный процесс, как и в случае решения задачи при ограничении по стоимости. Различие в решениях заключается лишь в том, что теперь вычисленные значения m^*_j, $j = 1,..., N$, зависят от фиксированного значения неопределенного множителя $\lambda = \lambda_0$. Если при выбранном значении этого параметра окажется, что полученные решения удовлетворяют условиям

$$\text{а)} \sum_{j-1}^{N} \omega_j m_j^* \le W; \ \text{б)} \ \lambda(W - \sum_{j=1}^{N} \omega_j m_j^*) = 0, \quad (3.11)$$

то они и составляют оптимальную стратегию резервирования $m^* = (m^*_1,...., m^*_N)^T$, обеспечивающую максимум функции надежности при выполнении обоих ограничений. В противном случае необходимо изменить значение $\lambda = \lambda_0$ и повторить вычисления.

Необходимо при этом учитывать экономический смысл множителя λ как *теневой цены* ресурса W. Если

на очередном этапе вычислений окажется, что $\sum_{j-1}^{N} \omega_j m_j^* < W$, то прежнее значение λ следует уменьшить, полагая, например, $\lambda := \lambda - \Delta\lambda$, где $\Delta\lambda$ некоторое положительное приращение, а в случае выполнения условия $\sum_{j-1}^{N} \omega_j m_j^* > W$ - наоборот, увеличить λ, полагая λ: = $\lambda + \Delta\lambda$. Если же величина *W* слишком велика, сумма $\sum_{j-1}^{N} \omega_j m_j^*$ будет сходиться к величине $\bar{W} < W$, и тогда значения λ будут стремиться к нулю, что обеспечит выполнение условия (3.11) *б) (*условие дополняющей нежесткости Слейтера). При организации вычислительного процесса рекомендуется уменьшить величину шага $\Delta\lambda$, например, в два раза каждый раз, когда разность $\sum_{j-1}^{N} \omega_j m_j^*$ -W меняет свой знак. Для организации правила остановки также можно задаваться точностью приближения, полагая, например, $|W - \sum_{j=1}^{N} \omega_j m_j^*| \le \varepsilon$, где ε - точность аппроксимации.

4.4.Задача распределения двух видов ресурсов

Технику сокращения размерности задачи с помощью метода Лагранжа иллюстрируем на примере решения следующей производственной задачи. Необходимо найти наилучший план (или стратегию) распределения имеющихся двух видов ресурсов (*производственных факторов*) соответственно в объемах C_1 и C_2 между *N* производственными программами, рассматривая в качестве критерия оптимальности функцию суммарного дохода от реализации программ. Для построения математической модели задачи обозначим через $r_j(x_j, y_j)$, $j = 1, \ldots, N$ - функцию дохода j -ой программы, когда для ее реализации выделены ресурсы x_j и y_j , а через $R(x;y)$ обозначим функцию суммарного дохода. Тогда оптимальную программу можно получить, решая задачу

$$R(x, y) = \sum_{j=1}^{N} r_j(x_j, y_j) \to \max, \qquad (4.1)$$

$$\{(x_j, y_j)\}$$

$$\sum_{j=1}^{N} x_j \le C_1, \ \sum_{j=1}^{N} y_j \le C_2$$

$$x_j, y_j \ge 0, \ \forall j$$

По своей форме эта задача не отличается от задачи (2.1) и может быть решена аналогичным образом. Для этой цели введем в рассмотрение последовательность функций $\{f_к(z_1, z_2)\}$ и подвергнем задачу *динамизации,* «*погружая*» ее в N – шаговый процесс перехода в оптимальное состояние. Каждое возможное состояние S_k, будем характеризовать количеством звеньев $к$, текущими уровнями ресурсов z_1 и z_2, (заменяющих C_1 и C_2) и функцией $f_к(z_1, z_2)$, которую определим с помощью следующего функционального уравнения

$$f_к(z_1, z_2) = \max_{0 \le x_к \le z_1,\ 0 \le y_к \le z_2} \{r_к(x_к, y_к) + f_{к-1}(z_1 - x_к; z_2 - y_к)\}, \quad (4,2)$$

которое справедливо для всех $к = 1, ..., N$, и ресурсов $z_1 = 0, 1, ..., C_1$ и $z_2 = 0, 1, ..., C_2$.

В этом уравнении присутствуют двумерные функции $f_к(z_1, z_2)$ и двумерные решения $(x_j; y_j)$, $к = 1, ... ,N$. По мере увеличения количества видов распределяемых ресурсов размерность задачи возрастает, что, в свою очередь, приведет к увеличению объема необходимых вычислений. В этом и заключается упомянутое выше *«проклятие*

размерности», которое присуще и динамическому программированию. Применение метода множителей Лагранжа позволяет избавиться от части ограничений, путем их включения в состав целевой функции. Для этой цели выберем одно из ограничений, например, ограничение по ресурсу C_2 для включения в состав целевой функции. Соответствующая функция Лагранжа будет иметь вид

$$L(x, y, \lambda) = \sum_{j=1}^{N} r_j(x_j, y_j) + \lambda(C_2 - \sum_{j=1}^{N} y_j) =$$

$$\sum_{j=1}^{N} (r_j(x_j, y_j) - \lambda y_j) + \lambda C_2 . \quad (4,3)$$

В этом выражении слагаемое λC_2 не влияет на максимизацию функции $L(x, y, \lambda)$ по x_j и y_j, $j = 1, ..., N$, поэтому им можно пренебречь и оптимизационную задачу представить в виде

$$L(x, y, \lambda) = \sum_{j=1}^{N} (r_j(x_j, y_j) - \lambda y_j) \to \max \quad (4,4)$$

$$\{x_j; y_j\}$$

$$\sum_{j=1}^{N} x_j \le C_1$$

$$\lambda, x_j, y_j \ge 0, \ \forall j$$

«Погружая» теперь эту задачу в семейство динамических задач путем замены N на κ, C_1 на z, и рассматривая функции $f_\kappa(z)$, $\kappa = 1,\ldots, N$, для всех $z = 0, 1,\ldots, C_1$, как оптимальные уровни доходов, приходим к функциональному уравнению

$$f_\kappa(z) = \max_{\{x_j;\, y_j\}} \sum_{j=1}^{\kappa} (r_j(x_j, y_j) - \lambda y_j) =$$

$$= \max_{\substack{0 \le x_\kappa \le z \\ y_\kappa \ge 0 \\ \sum_{j=1}^{\kappa} x_j \le z \\ \lambda, x_j, y_j \ge 0, \forall j}} \left\{ (r_\kappa(x_\kappa, y_\kappa) - \lambda y_\kappa) + f_{\kappa-1}(z - x_\kappa) \right\} \qquad (4.5)$$

для всех значений $\kappa = 1,\ldots, N$, и $z = 0, 1,\ldots, C_1$.

Выигрыш от этих действий состоит в том, что, с одной стороны, функции $f_\kappa(z)$ теперь являются одномерными, кроме того, в формуле (4,5) переменные x_κ и y_κ становятся независимыми и, поэтому, оптимизацию по ним можно выполнить раздельно. Однако в новой задаче (4.5) теперь присутствует неопределенный множитель

$\lambda \ge 0$, значение которого необходимо определить таким образом, чтобы выполнялось ограничение по весу $\sum y_j(\lambda) \le C_2$. Это и служит платой за снижение размерности функцией $f_\kappa(z)$, $\kappa = 1,\ldots, N$.

Алгоритм решения задачи сводится к следующему.

Шаг 1. Положить $\kappa = 1$, присвоить параметру λ некоторое начальное значение, например, $\lambda = \lambda_o$, вычислить значения функции

$$а)\ r_1'(x_1, \lambda) = \max_{y_1 \ge 0} \{ r_1(x_1, y_1) - \lambda y_1 \}$$

для всех $x_1 = 0, 1,\ldots, C_1$, и сохранить результаты в виде таблицы H_1, содержащей массив $\{x_1, r_1'(x_1, \lambda), y_1(x_1)\}$; полагая далее $f_O(z - x_1) = 0$ для всех $z = 0, 1,\ldots, C_1$, используя значения функции $r_1'(x_1, \lambda)$ из таблицы H_1, вычислить значения функции

$$б)\ f_1(z) = \max_{0 \le x_2 \le z} \{ r_1'(x_1, \lambda) + f_o(z - x_1) \}$$

для всех $z = 0, 1,\ldots, C_1$, и сохранить результаты в виде таблицы T_1, содержащей массив данных $\{z, f_1(z), x_1(z)\}$, $z = 0, 1,\ldots, C_1$.

Шаг 2. Положить $\kappa = 2$, вычислить значения функции

$$а)\ r_2'(x_2, \lambda) = \max_{y_\kappa \ge 0} \{ r_2(x_2, y_2) - \lambda y_2 \}$$

для всех $x_1 = 0, 1, \ldots, C_1$, сохранив результаты в таблице H_2 в виде массива $\{x_2, r_2^{/}(x_2, \lambda), y_2(x_2)\}$, $x_2 = 0, 1, \ldots, C_1$, используя далее значения функции $r_2^{/}(x_2, \lambda)$, вычислить значения функции

$$б)\ f_2(z) = \max_{0 \le x_2 \le z} \{r_2^{/}(x_2, \lambda) + f_1(z - x_2)\}$$

для всех значений $z = 0, 1, \ldots, C_1$, и сохранить результаты в виде таблицы T_2, содержащей массив данных $\{z, f_2(z), x_2(z)\}$, $z = 0, 1, \ldots, C_1$, и так далее.

Шаг к. Вычислить значения функции

$$а)\ r_к^{/}(x_к, \lambda) = \max_{y_к \ge 0} \{r_к(x_к, y_к) - \lambda y_к\},$$

для всех значений для всех $x_1 = 0, 1, \ldots, C_1$, сохранив результаты в виде таблицы $H_к$,

Таблица *Нк*

z	$r_к^{/}(x_к, \lambda)$	$y_к(x_к)$
0	$r_к^{/}(0, \lambda)$	$y_к(0)$
1	$r_к^{/}(1, \lambda)$	$y_к(1)$
.	.	.
.	.	.
.	.	.
C	$r_к^{/}(C_1, \lambda)$	$y_к(C_1)$

далее используя массив $f_{к-1}(z)$ предыдущего шага, вычислить функцию

$$б)\ f_к(z) = \max_{0 \le x_к \le z} \{r_к^{/}(x_к, \lambda) + f_{к-1}(z - x_к)\}$$

для всех $z = 0, 1, \ldots, C_1$, и сохранить результаты в виде таблицы $T_к$, и т.д.

Таблица Тк

Z	$f_к(z)$	$x_к(z)$
0	$f_к(0)$	$x_к(z)$
1	$f_к(1)$	$x_к(z)$
.	.	.
.	.	.
.	.	.
C_1	$f_к(C_1)$	$x_к(C_1)$

Шаг N. Положим $к = N$, $z = C_1$, вычислить значения функции

$$а)\ r_N^{/}(x_N, \lambda) = \max_{y_N \ge 0} \{r_N(x_N, y_N) - \lambda y_N\}$$

для всех значений $x_N = 0, 1, \ldots, C_1$, и сохранить результаты в виде таблицы H_N, аналогичной таблице $H_к$, далее

используя значения функции $r'_N(x_N, \lambda)$ из таблицы H_N,, вычислить

$$б)\ f_N(z) = \max_{0 \le x_N \le C_1} \left\{ r'_N(x_N, \lambda) + f_{N-1}(z - x_N) \right\}$$

и сохранить результаты в виде таблицы T_N:

Таблица T_N

Z	$f_N(z)$	$x_N(z)$
C_1	$f_N(C_1)$	$x_N(C_1)$

Прямой ход алгоритма этим завершается. Искомые результаты можно найти обратным ходом алгоритма путем обработки таблиц $(T_N, H_N), (T_{N-1}, H_{N-1}) \ldots, (T_1, H_1)$.

Процедура обработки таблиц весьма проста: для $k = N$ из таблицы T_N находим величины $x^*_N = x_N(C_1)$ и $f_N(C_1)$, далее по значению x^*_N в таблице H_N находим решение $y_N(x_N)$. Для $k = N - 1$ строка таблицы T_{N-1}, соответствующая уровню текущего ресурса $z = C_1 - x^*_N$, содержит значение $x^*_{N-1} = x_{N-1}(z)$, а в таблице H_{N-1} этому значению x_{N-1} соответствует решение $y_{N-1}(x_{N-1})$ и т.д. В последних таблицах T_1 и H_1, находим значения $x^*_1 = x_1(z)$ при $z = C_1 - x^*_N - \ldots - x^*_2$ и $y_1(x_1)$. Заметим, что все эти

решения зависят от фиксированного значения $\lambda = \lambda_0$. Если для них выполняются условия

$$\sum_{j=1}^{N} y_j \le C_2 \quad \text{и} \quad \lambda (C_2 - \sum_{j=1}^{N} y_j) = 0,$$

то найденные решения x^*_j и y^*_j, $j = 1, \ldots, N$, составляют оптимальную стратегию распространения ресурсов C_1 и C_2. В противном случае необходимо измерить текущее значение $\lambda = \lambda_0$ и повторить вычисления. При изменении текущего значения λ на некоторую величину $\Delta\lambda > 0$, т.е. полагая $\lambda = \lambda \pm \Delta\lambda$, следует учитывать то обстоятельство, что множитель λ как коэффициент чувствительности оптимального значения целевой функции R^* по ресурсу C_2 выступает также в качестве *«теневой»* цены этого ресурса. Поэтому, если окажется, что $\sum_{j=1}^{N} y_j(\lambda) < C_2$, на следующей итерации значение λ необходимо уменьшить, положив $\lambda := \lambda - \Delta\lambda$. Это означает, что прежнее значение λ было слишком большое, что *«сдерживало»* потребление ресурса C_2. И наоборот, если выполняется условие $\sum_{j=1}^{N} y_j(\lambda) > C_2$, значения λ следует увеличить,

полагая $\lambda = \lambda + \Delta\lambda$, т.к. прежнее значение λ было низкое, что привело к использованию большего количества ресурса C_2. Может оказаться, что величина C_2 слишком большая для удовлетворения нужд производства, тогда вычислительный процесс не будет сходиться к этой величине, и для выполнения условия дополняющей нежесткости Слейтера $\lambda(\sum_{j=1}^{N} y_j - C_2) = 0$ необходимо, чтобы значения множителя λ сходились к нулю. Этого эффекта можно обнаружить, работая на ЭВМ в интерактивном режиме.

Рекомендуется при разработке машинной программы организовать процесс сходимости таким образом, чтобы всякий раз, когда разность $\sum_{j=1}^{N} y_j - C_2$ меняет свой знак, величину шага $\Delta\lambda$ уменьшить в два раза, полагая $\Delta\lambda := \Delta\lambda/2$. Это действие существенно ускорит процесс сходимости. Как принято в численных методах оптимизации, при необходимости сходящийся процесс можно остановить путем задания некоторого порогового значения для контролируемой величины $\sum_{j=1}^{N} y_j - C_2$.

Можно, например, релаксационный процесс остановить, когда выполняется условие $|\sum y_j - C_2| \le \varepsilon$, где ε - требуемая точность аппроксимации искомого решения.

4.5. Транспортная задача с нелинейной функцией затрат

Модель транспортной задачи с линейной целевой функцией рассматривалась нами во второй главе. Когда целевая функция задачи является нелинейной функцией переменных x_{ij}, $i = 1,..., m$, $j = 1,..., n$, решить задачу с помощью методов линейного программирования уже не представляется возможным. При умеренном числе отправителей (складов, пунктов отправления) решение транспортной задачи с нелинейной функцией затрат можно осуществить на основе принципа оптимальности динамического программирования. Пусть исходная задача представлена в виде

$$R(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} r_{ij}(x_{ij}) \to \min_{\{x_{ij}\}}, \qquad (5.1)$$

$$\sum_{j=1}^{n} x_{ij} = a_i, i = 1,..., m$$

$$\sum_{i=1}^{m} x_{ij} = b_j, j = 1,..., n$$

$$\sum a_i = \sum b_j$$

$$x_{ij} \geq 0 \forall i, j$$

где $r_{ij}(x_{ij})$ – нелинейные функции аргументов x_{ij} для всех i и j. Введем в рассмотрение функции $r_j(x_{1j},...,x_{mj}) = \sum_{i=1}^{m} r_{ij}(x_{ij}), j = 1,..., n,$ переменные z_i, характеризующие текущее состояние ресурсов a_i, $i = 1, ..., m$, последовательность функций $\{f_\kappa(z_1, z_2,..., z_m)\}, \kappa = 1,..., n,$ как функций оптимальных затрат и определим эти функции на основе задачи

$$f_\kappa(z_1,..., z_m) = \min_{\{x_{ij}\}} \left\{ \sum_{j=1}^{\kappa} r_j(x_{1j},..., x_{mj}) \right\}. \qquad (5.2)$$

$$\sum_{j=1}^{\kappa} x_{ij} = z_i, i = 1,..., m$$

$$\sum_{i=1}^{m} x_{ij} = b_j, j = 1,..., k$$

$$\sum_{i=1}^{m} z_i = \sum_{j=1}^{\kappa} b_j$$

Эта задача описывает процесс отыскания оптимальных перевозок в первые κ пунктов назначения, $\kappa = 1,...., n,$ и, по существу, подвергает исходную задачу (5.1) *динамизации, «погружая»* ее в совокупность n оптимизационных задач с целевыми функциями $f_\kappa(z_1,..., z_m), \kappa = 1,..., n,$ соответственно, текущим уровнем

ресурсов $z_i = 0, 1, \ldots, a_i, \quad i = 1, \ldots, m,$ и решениями $x_{1к}, x_{2к}, \ldots, x_{mк}$.

Для того чтобы вывести соответствующее функциональное уравнение динамического программирования, описывающее правило определения решений $x_{iк}$, $i = 1, \ldots, m$, на каждом этапе, можно воспользоваться либо свойством сепарабельности целевой функции, либо применить по отношению к решению задачи *принцип оптимальности.* Студентам рекомендуем самостоятельно выполнить первый путь построения правила оптимизации.

Для построения функционального уравнения на основе принципа оптимальности рассмотрим произвольное состояние $S_к$, которому соответствует задача (5.2): нахождение оптимального плана перевозок имеющихся в пунктах отправки объемов $z_1, \ldots, z_m$ в первые *к* пунктов назначения с уровнем спросов $b_1, b_2, \ldots, b_k$ соответственно. Выделим для этой цели последнему, *к* – му пункту назначения товары в объемах $x_{1к}, x_{2к}, \ldots, x_{mк}$, и потребуем, чтобы оставшиеся в пунктах отправления объемы $z_1 - x_{1к}, z_2 - x_{2к}, \ldots, z_{mк} - x_{mк}$ были наилучшим образом (т.е. с минимальными затратами) перевезены в

оставшиеся *к - 1* пунктов назначения. Обозначив величину оптимальных затрат, связанных с удовлетворением спроса в этих *к - 1* пунктах, через $f_{к-1}(z_1 - x_{1к}, z_2 - x_{2к}, \ldots, z_{mк} - x_{mк})$, функцию суммарных затрат для данного состояния (или шага) можно представить в виде

$$r_к(x_{1k}, \ldots, x_{mк}) + f_{к-1}(z_1 - x_{1к}, z_2 - x_{2к}, \ldots, z_{mк} - x_{mк}), \quad (5.3)$$

где обозначено $r_к(x_1, \ldots, x_{mк}) = \sum_{i=1}^{m} r_{iк}(x_{iк})$. Эта функция зависит только от переменных текущего шага $x_{1k}, \ldots, x_{mк}$, следовательно, минимизируя ее по этим переменным и обозначая полученный оптимальный результат через $f_к(z_1, \ldots, z_m)$, получим искомое функциональное уравнение

$$f_к(z_1, \ldots, z_m) = \min_{\{x_{ik}\}} \left\{ r_к(x_{1к}, \ldots, x_{mк}) + f_{к-1}(z_1 - x_{1к}, \ldots, z_m - x_{mк}) \right\}$$

$$0 \le x_{iк} \le z_i, i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{iк} = b_к \quad (5.4)$$

для всех $к = 1, \ldots, n,$ и $z_i = 0, 1, \ldots, a_i, \quad i = 1, \ldots, m.$ По определению, в(5.4) обозначено

$$f_{\kappa-1}(z_1 - x_{1\kappa},...,z_m - x_{m\kappa}) = \min_{(x_{ij})} \sum_{j=1}^{\kappa-1} r_j(x_{1j},...,x_{mj}). \quad (5.5)$$

$$\sum_{j=1}^{\kappa-1} x_{ij} = z_i - x_{i\kappa}$$

$$i = 1,...,m$$

$$\sum_{i=1}^{m} x_{ij} = b_j, j = 1,...,\kappa-1$$

$$\sum_{i=1}^{m} (z_i - x_{i\kappa}) = \sum_{j=1}^{\kappa-1} b_j$$

Функциональное уравнение (5.4) описывает n – шаговый процесс выработки оптимального решения относительно переменных $x_{1\kappa}, x_{2\kappa},..., x_{m\kappa}$, $\kappa = 1,...,n$, соответствующих текущему уровню ресурсов $z_1, z_2,..., z_m$, причем, $z_i = 0, 1,...,a_i$, $i = 1,...., m$. Как видно из (5.4), на каждом шаге оптимизации необходимо вычислить и протабулировать m – мерные функции $f_\kappa(z_1,...,z_m)$ с соответствующими решениями $x_{1\kappa}(z), x_{2\kappa}(z),..., x_{m\kappa}(z)$, которые зависят от текущего уровня вектора ресурсов $z = (z_1,...,z_m)^T$. Когда k достигает значения n, а ресурсы z_i будут равны a_i, $i = 1,...., m$, задача (5.4) определяет

оптимальный уровень затрат, соответствующий оптимальной стратегии перевозок.

Быстродействие и соответствующие программные средства современных ЭВМ в принципе позволяют при небольших значениях m выполнить все вычисления и обработку результатов за приемлемые вычислительные ресурсы по времени и памяти, однако при больших значениях m возникают значительные трудности вычислительного характера.

Иллюстрируем работу алгоритма динамического программирования на следующих частных случаях транспортной задачи.

а) Транспортная задача «2хn» (два пункта отправления и n пунктов назначения). В этом случае задача (5.4) превратится в задачу

$$f_\kappa(z_1, z_2) = \min\{r_{1\kappa}(x_{1\kappa}) + r_{2\kappa}(x_{2\kappa}) + f_{\kappa-1}(z_1 - x_{1\kappa}, z_2 - x_{2\kappa})\},$$

$$0 \le x_{i\kappa} \le z_i, i = 1,2$$

$$x_{1\kappa} + x_{2\kappa} = b_\kappa \quad (5.6)$$

для всех значений $\kappa = 1, 2,..., n$, и $z_i = 0, 1,..., a_i$, $i = 1, 2$.

Переменные этой задачи связаны друг с другом уравнением $x_{1\kappa} + x_{2\kappa} = b_\kappa$, следовательно, можно в

выражении для функций $f_к(z_1, z_2)$ исключить одну из переменных, например, переменную z_2, и превратить их в одномерные функции. Для этой цели представим x_{2k} в виде $x_{2к} = b_к - x_{1к}$, а из уравнения связи $z_1 + z_2 = \sum_{j=1}^{к} b_j$ вычислим $z_2 = \sum_{j=1}^{к} b_j - z_1$. Тогда ограничения задачи (5.6) преобразуются в форму $0 \le x_{1k} \le z_1$, $0 \le b_к - x_{1к} \le \sum_{j=1}^{к} b_j - z_1$. Объединяя эти два условия, для переменной $x_{1к}$ находим ее нижний и верхний пределы:

$$A \le x_{1к} \le B,$$
$$A = \max\left\{0, z_1 - \sum_{j=1}^{к-1} b_j\right\}, \qquad (5.7)$$
$$B = \min\{z_1, b_к\}.$$

С учетом этих преобразований представим функциональное уравнение (4.6) в виде

$$f_к(z_1) = \min_{A \le x_{1к} \le B} \{r_{1к}(x_{1к}) + r_{2к}(b_к - x_{1к}) + f_{к-1}(z_1 - x_{1к})\} \quad (5.8)$$

для всех $к = 1, 2, \ldots, n$ и $z_1 = 0, 1, \ldots, a_1$. Всякий раз, когда для фиксированного значения z_1 на основе задачи (5.8) будет определено решение $x_{1к}(z_1)$, значение переменной

$x_{2к}$ определится из выражения $x_{2к} = b_к - x_{1к}(z_1)$ для всех значений z_1 и $к$. Основные шаги соответствующего вычислительного алгоритма сводятся к следующему.

Шаг 1. Положить $к = 1$, $f_o(z_1) = 0$ для всех $z_1 = 0, 1, \ldots, a_1$, вычислить значение функции

$$f_1(z_1) = \min_{A \le x_{11} \le B} \{r_{11}(x_{11}) + r_{12}(b_1 - x_{11}) + f_0(z_0 - x_{11})\}$$

для всех значений $z_1 = 0, 1, \ldots, a_1$, $A = \max\{0, z_1\}$, $B = \min\{z_1, b_1\}$, и сохранить результаты в виде таблицы T_1, содержащей массивы данных $\{z_1, f_1(z), x_{11}(z_1)\}, z_1 = 0,1,\ldots, a_1$.

Шаг 2. Положить $к = 2$, используя массив $f_1(z_1)$ первого шага, вычислить значение функции

$$f_2(z_1) = \min_{A \le x_{12} \le B} \{r_{12}(x_{12}) + r_{22}(b_2 - x_{12}) + f_1(z_1 - x_{12})\}$$

для всех значений $z_1 = 0, 1, \ldots, a_1, A = \max\{0, z_1 - b_1\}$, $B = \min\{z_1, b_2\}$ и сохранить результаты в виде таблицы T_2 с данными $\{z_1, f_2(z_1), x_{12}(z_1)\}, z_1 = 0,1,\ldots, a_1$, и т.д.

Шаг к. Используя массив $f_{к-1}(z_1)$ предыдущего шага, вычислить значение функции

$$f_к(z_1) = \min_{A \le x_{1к} \le B} \{r_{1к}(x_{1к}) + r_{2к}(b_к - x_{1к}) + f_{к-1}(z_1 - x_{1к})\}$$

для всех значений $z_1 = 0, 1, \ldots, a_1$ и сохранить результаты в виде таблицы $T_к$.

Таблица Тк

Z_1	$f_к(z_1)$	$x_{1к}(z_к)$
0	$f_к(0)$	$x_{1к}(0)$
1	$f_к(1)$	$x_{1к}(1)$
.	.	.
.	.	.
.	.	.
α_1	$f_к(\alpha_1)$	$x_{1к}(\alpha_к)$

и так далее, до последнего шага $к = n$.

Шаг n. Положить $к = n$, $z_1 = a_1$, используя массив $f_{n-1}(z_1)$ предыдущего шага, вычислить значение функции

$$f_n(a_1) = \min_{A \le x_{2n} \le B} \{r_{1n}(x_{1n}) + r_{2n}(b_n - x_{1n}) + f_{n-1}(a_1 - x_{1n})\}$$

для значений $A = max\ \{0,\ a_1 - \sum_{j=1}^{n-1} b_j\}$, $B = \min\{a_1, b_n\}$ и сохранить результаты в виде таблицы T_n.

Таблица T_n

z_1	$f_n(z_1)$	$x_{1n}(z_1)$
a_1	$f_n(a_1)$	$x_{1n}(a_1)$

Полученные на этом шаге решения $x_{1n}^* = x_{1n}(a_1), x_{2n}^* = b_n - x_{1n}^*$ являются оптимальными. Величина $f_n(a_1)$ характеризует минимальные затраты транспортировки. Обработка таблиц $T_n, T_{n-1}, \ldots, T_1$ обратным ходом можно осуществить с помощью следующего подалгоритма:

шаг 1. Положить $к = n$, $z_1 = a_1$;

шаг 2. Вывести $f_к(z_1)$, $x_{1к}(z_1)$;

шаг 3. Вычислить $x_{2к} = b_к - x_{1к}(z_1)$;

шаг 4. Положить $z_1 := z_1 - x_{1к}(z_1)$;

шаг 5. Положить $к := к - 1$;

шаг 6. Если к ≥ 1, перейти к шагу 2;

шаг 7. Остановиться.

Удобно представить результаты решения задачи в виде таблицы

Таблица результатов

κ	$x_{1\kappa}$	$x_{2\kappa}$	$f_\kappa(z_1)$
n	x_{1n}	x_{2n}	$f_n(z_1)$
$n-1$	$x_{1,n-1}$	$x_{2,n-1}$	$f_{n-1}(z_1)$
.	.	.	.
.	.	.	.
.	.	.	.
1	x_{11}	x_{21}	$f_1(z_1)$

Величина $f_n(a_1)$, разумеется, будет удовлетворять условию

$$f_n(a_1) = R(x^*) = \sum_{i=1}^{2}\sum_{j=1}^{n} r_{ij}(x_{ij}^*). \qquad (5.9)$$

б) Транспортная задача «3xn» (три пункта отправления и n пунктов назначения).

Согласно условиям задачи (5.4), при $m = 3$ мы имеем дело с трехмерными функциями оптимальных затрат $f_\kappa(z_1, z_2, z_3), \kappa = 1,...,n$. Применяя метод множителей Лагранжа, можно снизить размерность этих функций до единицы, превратив их в одномерные функции вида $f_\kappa(z_1)$, $\kappa = 1,...,n$. Для иллюстрации этой

чисто математической возможности, представим задачу (5.1) для $m = 3$ в виде

$$R(x) = \sum_{i=1}^{3}\sum_{j=1}^{n} r_{ij}(x_{ij}) \to \min. \qquad (5.10)$$

$$\{x_{ij}\}$$

$$\sum_{j=1}^{n} x_{ij} = a_i, i = 1,2,3$$

$$\sum_{i=1}^{3} x_{ij} = b_j, j = 1,...,n$$

$$a_1 + a_2 + a_3 = \sum_{j=1}^{n} b_j$$

Заметим прежде всего, что при включении ограничений $\sum_{j=1}^{n} x_{2j} = a_2$ и $\sum_{j=1}^{n} x_{3j} = a_3$ в состав функции Лагранжа, можно избавиться от переменных z_2 и z_3, которые представляют текущее значение параметров a_2 и a_3 соответственно, причем можно использовать лишь один множитель Лангранжа, так как всякий раз, когда ограничение $\sum_{j=1}^{n} x_{2j} = a_2$ будет иметь место, благодаря связи $\sum_{j=1}^{n} x_{3j} = \sum_{j=1}^{n} b_j - a_1 - a_2$ автоматически будет

выполняться и второе ограничение $\sum_{j=1}^{n} x_{3j} = a_3$. Таким образом, функцию Лагранжа можно представить в виде

$$L(x,\lambda) = \sum_{i=1}^{3}\sum_{j=1}^{n} r_{ij}(x_{ij}) + \lambda(\sum_{j=1}^{n} x_{2j} - a_2) + (\sum_{j=1}^{n} x_{3j} - a_3) =$$

$$= \sum_{j=1}^{n}(\sum_{i=1}^{3} r_{ij}(x_{ij}) + \lambda x_{2j} + x_{3j}) - \lambda a_2 - a_3 . \qquad (5.11)$$

В этом выражении слагаемое $-\lambda a_2 - a_3$ не влияет на искомые решения x_{1j}, x_{2j} и x_{3j}, поэтому им можно пренебречь. Тогда решаемая задача примет вид

$$L(x,\lambda) = \sum_{j=1}^{n}\Big\{ \sum_{i=1}^{3} r_{ij}(x_{ij}) + \lambda x_{2j} + x_{3j} \Big\} \to \min . \qquad (5.12)$$

$$(x_{1j}, x_{2j}, x_{3j})$$

$$0 \le x_{1j} \le a_1$$

$$x_{1j} + x_{2j} + x_{3j} = b_j$$

$$j = 1,...,n$$

Значение множителя λ необходимо определить таким образом, чтобы выполнялись ограничения по a_2 и a_3.

Подвергая теперь решение задачи (5.12) динамизации с помощью последовательности функций $\{f_к(z_1)\}$, получим функциональное уравнение в виде

$$f_к(z_1) = \min\left\{(\sum_{i=1}^{3} r_{iк}(x_{iк}) + \lambda x_{2к} + x_{3к}) + f_{к-1}(z_1 - x_{1к})\right\}$$

$$(x_{1к}, x_{2к}, x_{3к})$$

$$0 \le x_{1к} \le z_1$$

$$x_{1к} + x_{2к} + x_{3к} = b_к$$

$$x_{iк} \ge 0, i = 1,2,3 \qquad (5.13)$$

для всех $к = 1,...,n,$ и $z_1 = 0, 1, ..., a_1$. При этом минимизацию по $x_{2к}$ *и* $x_{3к}$ в (5.13) можно выполнить раздельно, так как они уже являются независимыми переменными. Алгоритм решения задачи сводится к следующему.

Пусть $\lambda = \lambda_0$ - некоторое начальное значение множителя λ.

Шаг 1. Положить $к = 1$, $f_o(z_1) = 0$, $z_1 = 0, 1, ..., a_1$, вычислить

а) $h_1(x_{11}, \lambda) = \min\{r_{21}(x_{21}) + r_{31}(x_{31}) + \lambda x_{21} + x_{31}\}$

$$x_{21}, x_{31} \ge 0$$

$$x_{21} + x_{31} = b_1 - x_{11}$$

для всех значений $x_{11} = 0,1,...,\min\{a_1, b_1\}$ и сохранить результаты в виде таблицы H_1 с данными $\{x_{11}, h_1(x_{11}, \lambda), x_{21}(x_{11})\}, x_{11} = 0,1,...,\min\{a_1, b_1\}$; далее вычислить

б) $f_1(z_1) = \min\{r_{11}(x_{11}) + h_1(x_{11}, \lambda) + f_o(z_1 - x_{11})\}$

$$A \le x_{11} \le B$$

для всех значений $z_1 = 0,1,...,a_1, A = 0, B = \min\{z_1, b_1\}$, и сохранить результаты в виде таблицы T_1 с данными $\{z_1, f_1(z_1), x_{11}(z_1)\}, z_1 = 0,1,...,a_1$.

Шаг 2. Положить *к = 2*, вычислить

а) $h_2(x_{12}, \lambda) = \min\{r_{22}(x_{22}) + r_{32}(x_{32}) + \lambda x_{22} + x_{32}\}$,

$$x_{22}, x_{32}$$

$$x_{22} + x_{32} = b_2 - x_{12}$$

сохранив результаты в виде таблицы H_2 с массивом данных $\{x_{12}, h_2(x_{12}, \lambda), x_{22}(x_{12})\}, x_{12} = 0,1,...,\min\{a_1, b_2\}$, далее вычислить

б) $f_2(z_1) = \min\{r_{12}(x_{12}) + h_2(x_{12}, \lambda) + f_1(z_1 - x_{12})\}$

$$A \le x_{12} \le B$$

для всех значений $z_1 = 0,1,...,a_1$, и сохранить результаты в виде таблицы T_2 с данными $\{z_1, f_2(z_1), x_{12}(z_1)\}, z_1 = 0,1,...,a_1$, и так далее, до последнего шага.

Шаг n. Положить *к = n*, $z_1 = a_1$, вычислить

а) $h_n(x_{1n}, \lambda) = \min\{r_{2n}(x_{2n}) + r_{3n}(x_{3n}) + \lambda x_{2n} + x_{3n}\}$,

$$x_{2n}, x_{3n}$$

$$x_{2n} + x_{3n} = b_n - x_{1n}$$

сохранив результаты в виде таблицы H_n с массивом данных $\{x_{1n}, h_n(x_{1n}, \lambda), x_{2n}(x_{1n})\}, x_{1n} = 0,1,...,\min\{a_1, b_n\}$, далее вычислить

б) $f_n(a_1) = \min\{r_{1n}(x_{1n}) + h_n(x_{1n}, \lambda) + f_{n-1}(a_1 - x_{1n})\}$

$$A \le x_{1n} \le B$$

для всех значений $A = 0, B = \min\{a_1, b_n\}$ и сохранить результаты в виде таблицы T_n, содержащей массив данных $\{z_1, f_n(z_1), x_{1n}(z_1)\}$ для одного единственного значения $z_1 = a_1$. Этим завершается прямой ход алгоритма.

Для вывода результатов обрабатываем таблицы $\{H_к, T_к\}$, *к = n, n-1,...,1*, в обратной последовательности. Так, при *к = n* из таблицы T_n выводим величины $x^*_{1n} = x_{1n}(a_1)$ и $f_n(a_1)$, а из таблицы H_n – значение

$x_{2n}(x^*_{1n})$, при этом $x^*_{3n} = b_n - x^*_{1n} - x^*_{2n}$. Переходя к таблицам T_{n-1} и H_{n-1}, выводим $x^*_{1,n-1} = x_{1,n-1}(a_1 - x^*_{1n})$ и $x^*_{2,n-1} = x_{2,n-1}(x^*_{1,n-1})$ и так далее, до последних таблиц T_1 и H_1. В результате получим оптимальное решение в виде тройки $(x^*_{1\kappa}, x^*_{2\kappa}, x^*_{3\kappa}), \kappa = n, n-1, ..., 1$. Оно, разумеется, зависит от фиксированного значения $\lambda = \lambda_o$. Если при этом условия $\sum_{j=1}^{n} x^*_{2j} = a_2$ и $\sum_{j=1}^{n} x^*_{3j} = a_3$ имеют место, то найденное решение $(x^*_{1\kappa}, x^*_{2\kappa}, x^*_{3\kappa}), \kappa = 1, ..., n$, составит оптимальную стратегию перевозки. Оптимальное значение функции издержек будет равно величине

$$R^* = R(x^*) = f_n(a_1) - \lambda \sum_{j=1}^{n} x^*_{2j} - \sum_{j=1}^{n} x^*_{3j} \qquad (5.14)$$

Если же контролируемые ограничения не выполняются, необходимо изменить значение параметра λ, положив $\lambda := \lambda \pm \Delta\lambda$, и повторить вычисления. Следует при этом иметь в виду, что из-за наличия в задаче ограничений в виде равенств, знак множителя λ может быть как положительным, так и отрицательным. Однако, если на очередном шаге будет иметь место условие $\sum x^*_{2j} \prec a_i$,

текущее значение λ следует уменьшить, а при условии $\sum x^*_{2j} \succ \alpha_i$, наоборот, - увеличить. В любом случае при организации вычислительного процесса на ЭВМ следует в режиме диалога с ЭВМ прогнозировать направление изменения λ с тем, чтобы ускорить процесс сходимости.

4.6. Задача оптимального управления

До сих пор нами были рассмотрены задачи анализа и принятия решения, в которых время как переменная величина не присутствовало, а задача состояла в выборе из заданного множества решений (или альтернатив) наиболее предпочтительного решения, обеспечивающего достижение максимума или минимума заданной целевой функции (функции качества). Другими словами, мы имели дело со *статическими* задачами оптимизации или задачами *математического программирования.* Даже в рамках динамического программирования, когда исходная задача «*погружается»* в семейство взаимосвязанных оптимизационных задач, соответствующих определенным этапам или траекториям перехода из заданного состояния в целевое состояние, время в явном виде не присутствует.

Как отмечалось в первом разделе главы, в настоящее время динамическое программирование считается одним из двух эффективных методов исследования и решения задачи динамической оптимизации (задачи оптимального управления), наряду с принципом максимума. В общем случае задача оптимального управления представляется в форме (см. задачу (1.3))

$$J(u(t)) = \int_{t_o}^{t_1} \phi(x,u,t)dt + F(x_1,t_1) \to \max. \qquad (6.1)$$

$$\{u(t)\} \in U$$

$$dx/dt = \varphi(x,u,t)$$

$$x(t_o) = x_o; x(t_1) = x_1$$

Она определяет цель управления динамическим объектом (процессом, системой), который описывается системой дифференциальных уравнений

$$dx/dt = \varphi(x,u,t), \qquad (6.2)$$

где $x(t) = (x_1(t),\ldots, x_n(t))^T$ - вектор фазовых переменных (или координат), $u(t) = (u_1(t),.., u_m(t))^T$ – вектор управления, значения которого принадлежат заданному множеству управления U, т.е. $\{u(t)\} \in U$, $\varphi(x, u, t)$ – заданная вектор - функция, t – время.

Задача управления заключается в выборе такого допустимого вектора $u^*(t)$, который обеспечит переход объекта из заданного состояния $x_0 = x(t_0)$, в котором он находится в начальный момент времени t_0, в целевое состояние $x_1 = x(t_1)$ в момент времени $t_1 > t_0$, и при этом функционал $J(u(t))$ (критерий качества управления) достигнет своего максимального значения.

В теории оптимального уравнения уравнение (6.2) называется *уравнением движения* системы или ее *траекторией* для промежутка времени $t_0 \le t \le t_1$, а задача (6.1), как отмечалось выше, называется *задачей Больца*. Когда в составе целевой функции задачи слагаемое $F(x_1, t_1)$ отсутствует, она называется *задачей Лагранжа*, а когда целевой функционал содержит лишь функцию $F(x_1, t_1)$, задача носит название *задачи Майера*. Все эти модификации на самом деле эквивалентны друг другу, в чем легко убедиться, применив соответствующее преобразование переменных [6].

Суть метода динамического программирования применительно к задаче (6.1) заключается в том, что с его помощью задача подвергается *динамизации* и «*погружается*» в семейство взаимосвязанных оптимизационных задач, которые характеризуются рядом параметров, а центральная идея оптимизации – *принцип оптимальности* позволяет вывести (или строить) основное функциональное управление ДП, которое рекуррентно описывает правило выбора оптимальных решений.

Для того чтобы вывести функциональное уравнение динамического программирования применительно к

задаче (6.1), введем в рассмотрение функцию оптимального поведения $f(x, t)$, которая равна оптимальному значению целевого функционала $J^*(x, t)$ для начального состояния x и начального момента времени t. По существу, нам необходимо найти значение $f(x_0, t_0)$, соответствующее оптимальному значению целевого функционала $J^*(x_0, t_0)$.

Согласно принципу оптимальности, если $f(x, t)$ является функцией оптимального поведения для задачи с начальным состоянием x и начальным моментом времени t, то $f(x+\Delta x, t+\Delta t)$ характеризует оптимальное поведение системы для второй части оптимальной траектории с начальным состоянием $x+\Delta x$ и начальным моментом времени $t+\Delta t$. Нетрудно видеть, что прирост функции оптимального поведения будет равен величине $\phi(x, u, t)\Delta t$, так как этот прирост обусловлен только изменением подынтегральной функции задачи (6.1). Значения $f(x, t)$ на всем промежутке времени, начавшемся в момент t, таким образом, представляют собой оптимальную сумму вкладов двух частей этого интервала времени, поэтому

$$f(x,t) = \max\{\phi(x,u,t)\Delta t + f(x+\Delta x, t+\Delta t)\}. \qquad (6.3)$$

$\{u(t)\}$

Выражение (6.3) представляет собой функциональное уравнение ДП для задачи (6.1). Предполагая, что $f(x,t)$ является однозначной и непрерывно дифференцируемой функцией своих $n+1$ переменных $x_1,...,x_n$ и t как начальных параметров (предположение о гладкости $f(x,t)$), можно разложить функцию $f(x+\Delta x, t+\Delta t)$ в точке (x,t) в ряд Тейлора, т. е.

$$f(x+\Delta x, t+\Delta t) = f(x,t) + (\partial f / \partial x)^T \Delta x + (\partial f / \partial t)\Delta t + ..., \quad (6.4)$$

где принято векторное дифференцирование $\partial f / \partial x = (\partial f / \partial x_1,..., \partial f / \partial x_n)^T$. Подставляя (6.4) в (6.3) и выполняя переход к пределу, когда $\Delta t \to 0$, получим соотношение

$$\partial f(x,t) / \partial t = -\max_{\{u(t)\}} \left\{ \phi(x,u,t) + \sum_{j=1}^{n} \frac{\partial f}{\partial x_j} \varphi_j(x,u,t) \right\}, \quad (6.5)$$

где использовано значение для предела $\lim_{\Delta t \to 0} \Delta x / \Delta t = dx / dt = \varphi(x,u,t)$.

Соответствие (6.5), представляющее собой дифференциальное уравнение в частных производных, носит название *уравнения Беллмана* или *функциональное уравнение динамического программирования* для задачи оптимального управления (6.1). Оно является рекуррентным соотношением, отвечающим граничному условию

$$f(x(t_1), t_1) = F(x_1, t_1), \quad (6.6)$$

которое соответствует конечному состоянию (x_1, t_1).

В общем случае дифференциальное уравнение (6.5), которое, очевидно, является нелинейным, не имеет аналитического решения, однако его дискретный вариант уже представляет собой обычную задачу математического программирования, которая может быть решена традиционными методами как многошаговая задача управления. Обозначая через *{u*(t)}* оптимальное решение задачи (6.5) и через *H(x, u*, t)* оптимальное значение ее правой части, приходим к известному *уравнению Гамильтона – Якоби*

$$H(x, u^*, t) + \partial f(x, t)/\partial t = 0. \quad (6.7)$$

На рис. 4.2. изображена оптимальная траектория $x^*(t)$, состоящая из 4-х участков, также оптимальных, которые соответствуют правилу выбора (6.5) и принятому принципу оптимальности. Согласно этому принципу, оптимальная траектория (или поведение) обладает тем свойством, что, *каковы бы ни были начальное состояние и решение в этот момент, последующие решения должны составлять оптимальное поведение относительно состояния, получающегося в результате первого решения*.

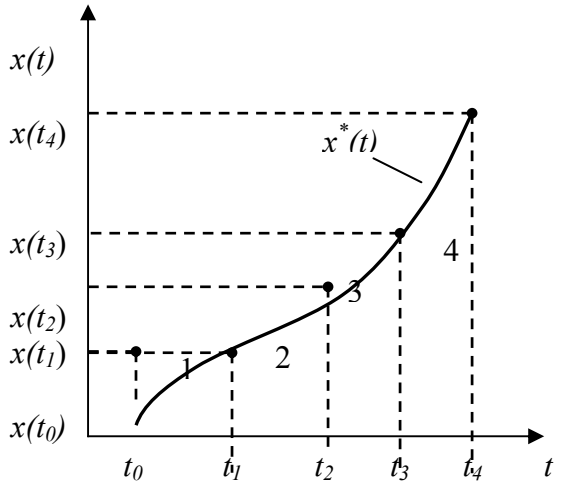

Рис. 4.2. Оптимальная траектория $x^*(t)$ и ее участки.

Предполагается, что начальное и конечное состояния траектории $x^*(t)$, соответствующей оптимальному уравнению $(u_1^*, u_2^*, u_3^*, u_4^*)$, фиксированы, т.е. $x(t_0) = x_0, x(t_4) = x_4$..

Согласно принципу оптимальности, отрезок 3 оптимальной траектории для моментов времени в пределах $t_2 \le t \le t_3$ должен представлять собой оптимальную траекторию по отношению к начальному состоянию $x(t_2)$ в начальном моменте t_2 вне зависимости от того, как система пришла к состоянию $x(t_2)$, являющемуся началом для этого участка траектории. Распространяя этот принцип на все отрезки траектории, получим оптимальную траекторию. Весьма интересная интерпретация принципа оптимальности дана Арисом [6]: «*Если вы не используете наилучшим образом то, чем вы располагаете, то вы никогда не распорядитесь наилучшим образом и тем, что вы могли бы иметь в дальнейшем».*

4.7. Связи с вариационной задачей и принципом максимума

Существует глубокая внутренняя связь между уравнением (6.5) и двумя другими классическими схемами оптимизации, а именно, *уравнением Эйлера* для *вариационной задачи* и *принципом максимума Понтрягина* для общей задачи оптимального управления (6.1). Эта связь является столь же фундаментальной, как и связь между условиями теоремы Куна – Таккера, условиями двойственности и условиями седловой точки функции Лагранжа для общей задачи математического программирования.

Ввиду большого научно – практического и познавательного значения этих трех вычислительный схем для современной исследовательской практики в области динамической оптимизации, ниже мы вкратце изложим основные аналитические соотношения для того чтобы и прояснить некоторые важные детали и связи, и возбудить у читателей интерес к дальнейшему развитию этого весьма важного аналитического аппарата для проблем системотехники.

Классическая задача вариационного исчисления представляется в виде [7]

$$J = \int_{t_o}^{t_1} \phi(x, \dot{x}, t)dt \to \max_{\{x(t)\}}, \qquad (7.1)$$

где *x(t)* – фазовая траектория системы, оба конца которой, $x(t_0) = x_0$ и $x(t_1) = x_1$, фиксированы, J - непрерывно дифференцируемая функция координат x, $\dot{x}$ и t, x_0, x_1, t_0 и t_1 фиксированные параметры.

Задача управления в постановке (7.1) сводится к тому, чтобы выбрать среди множества функций времени *x(t)* с фиксированными концами такую функцию, которая максимизирует интеграл от заданной функции $\phi(x, \dot{x}, t)$. Сравнивая задачи (6.1) и (7.1), приходим к выводу, что в вариационной задаче в качестве управления выступает скорость изменения фазовой координаты, т.е. $u(t) = \dot{x} \in E^n$, причем на управление *u(t)* никаких ограничений не накладывается. Допустим, что задача (7.1) имеет решение. Для того чтобы получить необходимые условия, которым искомое решение удовлетворяет, как и в нелинейном программировании, рассмотрим малые изменения этой оптимальной траектории в виде

$$z(t) = x(t) + \varepsilon\eta(t), \qquad (7.2)$$

где $\eta(t_0) = \eta(t_1) = 0$, ε - малый параметр, такой, что $lim\ z(t) = x(t)$ при $\varepsilon \to 0$. Учитывая вариацию (7.2), значение функционала в (7.1) на траектории $z(t)$ можно рассматривать как функцию, которая зависит от величины ε, т.е.

$$J(\varepsilon) = \int_{t_o}^{t_1} \phi(x + \varepsilon\eta, \dot{x} + \varepsilon\dot{\eta}, t)dt\,, \qquad (7.3)$$

и достигает своего максимума при $\varepsilon = 0$. Следовательно, учитывая условия $\eta(t_0) = \eta(t_1) = 0$, получим

$$\frac{\partial J(o)}{\partial \varepsilon} = \int_{t_0}^{t_1} (\frac{\partial \phi}{\partial x}\eta + \frac{\partial \phi}{\partial \dot{x}}\dot{\eta})dt = \int_{t_0}^{t_1} [\frac{\partial \phi}{\partial x} - \frac{d}{dt}(\frac{\partial \phi}{\partial \dot{x}})]\eta dt = 0\,. \quad (7.4)$$

При выводе этого соотношения использованы интегрирование по частям выражения $\frac{\partial \phi}{\partial \dot{x}}\dot{\eta}$ и равенство нулю значения $\frac{\partial \phi}{\partial \dot{x}}\eta$ из-за условий $\eta(t_0) = \eta(t_1) = 0$. Так как условие (7.4) должно выполняться для любого $z(t)$ с нулевыми граничными условиями, необходимо, чтобы выражение в квадратных скобках под знаком интеграла равнялось нулю для любого момента времени в интервале (t_0, t_1), т. е.

$$\frac{\partial \phi}{\partial x} - \frac{d}{dt}(\frac{\partial \phi}{\partial \dot{x}}) = 0\,. \qquad (7.5)$$

Полученное уравнение носит название *уравнения Эйлера* для вариационной задачи. Его структура подсказывает, что оно является обыкновенным дифференциальным уравнением второго порядка с граничными условиями $x(t_0) = x_0$ и $x(t_1) = x_1$. Оптимальная траектория $x(t)$, если она существует, называется *экстремалью*. Существуют и другие необходимые условия, которым должна удовлетворять *экстремаль*. В частности, из необходимого условия второго порядка $\partial^2 J / \partial \varepsilon^2 = 0$ непосредственно следует необходимое условие второго порядка $\partial^2 \phi / \partial \dot{x}^2 \le 0$ при всех $t \in (t_0, t_1)$, которое носит название *условия Лежандра*.

Сравним теперь друг с другом соотношения (7.5) и (6.5). Заметим, что в (6.5) функция оптимального поведения $f(x, t)$ представляет собой оптимальное значение функционала $J^*(x, t)$, кроме того, в (7.1) имеет место $u(t) = \dot{x}$. Необходимое условие существования максимума функции в правой части (6.5) есть

$$\frac{\partial}{\partial \dot{x}}\left\{\phi(x,\dot{x},t)+(\frac{\partial J^*}{\partial x})^T\dot{x}\right\}=0, \qquad (7.6)$$

где использовано уравнение $\dot{x} = \varphi(x,\dot{x},t)$, а сумма заменена на скалярное произведение векторов $\partial J^*/\partial x = (\partial J^*/\partial x_1, \ldots, \partial J^*/\partial x_n)^T$ и $\dot{x}$. Учитывая, что $\partial J^*/\partial x$ не зависит от $\dot{x}$, из (7.6) получим соотношение

$$\frac{\partial \phi}{\partial \dot{x}} = -\frac{\partial J^*}{\partial x}. \qquad (7.7)$$

Если теперь продифференцировать обе части уравнения (7.7) по t:

$$\frac{d}{dt}(\frac{\partial \phi}{\partial \dot{x}}) = -\frac{d}{dt}(\frac{\partial J^*}{\partial x}) = -\frac{\partial^2 J^*}{\partial t \partial x} - (\dot{x})'\frac{\partial^2 J^*}{\partial x^2}, \qquad (7.8)$$

а уравнение Беллмана (6.5) продифференцировать по вектору x:

$$-\frac{\partial}{\partial x}(\frac{\partial J^*}{\partial t}) = \frac{\partial \phi}{\partial x} + (\dot{x})'\frac{\partial^2 J^*}{\partial x^2}. \qquad (7.9)$$

и сравнить полученные результаты, то с учетом выражения для вектора $\partial^2 J^*/\partial t \partial x = (\partial^2 J^*/\partial t \partial x_1, \ldots, \partial^2 J^*/\partial t \partial x_n)^T$ и (nxn) - матрицы $\partial^2 J^*/\partial x^2$ приходим к выводу, что уравнение Беллмана порождает уравнение Эйлера для вариационной задачи. При этом условие Лежандра второго порядка сводится к условию отрицательной

полуопределенности или отрицательной определенности матрицы $\partial^2 \phi/\partial \dot{x}^2$.

Для выявления связи между уравнением Беллмана и принципом максимума Понтрягина необходимо вернуться к задаче (6.1) и применить по отношению к ней известные преобразования, аналогичные преобразованиям нелинейного программирования. Заметим, что в этой задаче основным ограничением является уравнение движения $\dot{x} = \varphi(x,u,t)$ или же уравнение $\varphi(x,u,t) - \dot{x} = 0$, записанное в векторной форме. Поэтому введем в рассмотрение двойственные переменные $y_1(t), \ldots, y_n(t)$, называемые неопределенными множителями Лагранжа, и составим для задачи (6.1) функцию Лагранжа (целевой функционал двойственной задачи) в виде

$$L(u(t), y(t)) = J(u(t)) + \int_{t_o}^{t_1} y^T(t)(\varphi(x,u,t) - \dot{x})dt =$$

$$= \int_{t_o}^{t_1} \{\phi(x,u,t)dt + y^T(t)(\varphi(x,u,t) - \dot{x})dt + F(x_1,t_1), \qquad (7.10)$$

где использовано обозначение вектора $y(t) = (y_1(t), \ldots, y_n(t))^T$. Двойка $(\{u^*(t)\}, \{y^*(t)\})$, принадлежащая пространству функций, называется седловой точкой функции Лагранжа, если имеет место двустороннее неравенство

$$L(u(t), y^*(t)) \le L(u^*(t), y^*(t)) \le L(u^*(t), y(t)). \quad (7.11)$$

Если точка $(u^*(t), y^*(t))$существует, то траектория $\{u^*(t)\}$ является оптимальным решением задачи (6.5). Переписав правое неравенство в (7.11) в виде

$$\int_{t_o}^{t_1} \{(y^*(t) - y(t))^T (\varphi(x^*, u^*, t) - \dot{x}^*)\} dt \le 0,$$

(7.12)заключаем, что для его выполнения для произвольных функций $\{y(t)\}$ должно выполняться неравенство $\dot{x}^* = \varphi(x^*, u^*, t)$. С другой стороны, согласно левому неравенству в (7.11), имеем

$$J\{u^*(t)\} \ge J\{u(t)\} + \int_{t_o}^{t_1} \{y^{*T}(t)(\varphi(x,u,t) - \dot{x})\} dt, \quad (7.13)$$

откуда следует, что для всех $\{u(t)\}$, удовлетворяющих ограничениям, имеет место $J\{u^*(t)\} \ge J\{u(t)\}$, т.е. траектория $\{u^*(t)\}$ является оптимальной. При этом имеет место соотношение $J\{u^*(t)\} = L(u^*(t), y^*(t))$, аналогичное равенству значений оптимизируемой функции и функции Лагранжа в нелинейном программировании (см. гл.3 части 1).

Необходимое условие существования седловой точки функции Лагранжа можно получить из следующих соображений. Если в выражении (7.10), вместо $y(t)$, подставить $y(t) + \Delta y(t)$, где $\Delta y(t)$ – произвольное непрерывное приращение функции $y(t)$, то для функции Лагранжа получим приращение

$$\Delta L(u(t), y(t)) = \int_{t_o}^{t_1} \Delta y^T(t)(\varphi(x,u,t) - \dot{x}) dt. \quad (7.14)$$

Для существования минимума функции Лагранжа относительно функции $\{y(t)\}$ необходимо, чтобы $\Delta L(u(t), y(t)) = 0$, т.е. необходимо выполнение уравнения движения $\dot{x} = \varphi(x,u,t)$. Для получения других необходимых условий проинтегрируем выражение $-y^T(t)\,\dot{x}(t)$ в (7.10) по частям. В результате получим

$$L(u(t), y(t)) = \int_{t_o}^{t_1} \{H(x,u,y,t) + \dot{y}x\} dt +$$

$$+ F(x_1, t_1) - [y(t_1)x(t_1) - y(t_0)x(t_0)], \quad (7.15)$$

где $H(x, u, y, t) = \phi(x, u, t) + y^T\varphi(x, u, t)$ называется *функцией Гамильтона.* Если теперь, вместо траектории

{u(t)}, подставить в (7.15) «возмущенную» траекторию *{u(t) + Δ u(t)},* то функция Лагранжа получит приращение

$$\Delta L(u(t), y(t)) = \int_{t_o}^{t_1} \{\frac{\partial H}{\partial u}\Delta u + (\frac{\partial H}{\partial x} + \dot{y})^T \Delta x\} dt + [\frac{\partial F}{\partial x_1} - y(t_1)]\Delta x_1, \qquad (7.16)$$

где введены обозначения векторов $\partial H/\partial u = (\partial H/\partial u_1, \ldots, \partial H/\partial u_m)^T$, $\partial H/\partial x = (\partial H/\partial x_1, \ldots, \partial H/\partial x_n)^T$. Для существования максимума функции Лагранжа необходимо, чтобы это приращение равнялось нулю, кроме того, поскольку условие (7.16) должно выполняться при любых приращениях $\Delta\{u(t)\}$, из условия $\Delta L(u(t), y(t)) = 0$ получим

$$\text{а)} \quad \partial H/\partial u = 0,\ t \in (t_0, t_1);$$

$$\text{б)} \quad \dot{y} = -\partial H/\partial x,\ t \in (t_0, t_1);$$

$$\text{в)} \quad y(t_1) = \partial F/\partial x_1 \qquad (7.17)$$

Условия группы *а)* показывают, что функция Гамильтона принимает свое максимальное значение в каждой точке оптимальной траектории $\{u^*(t)\}$, причем это – внутренний оптимум, так как на управления *{u(t)}* не наложено

никаких ограничений. Если ограничения присутствуют, то вместо *а),* получим условие

$$\max_{\{u\} \in U} H(x, u, y, t) \qquad (7.18)$$

для всех $t \in (t_0, t_1)$. Условие (7.18) имеет простую интерпретацию: в каждый момент времени функция Гамильтона достигает в точках оптимальной траектории $\{u^*(t)\}$ максимума относительно управляющих параметров $u_1^*(t), \ldots, u_m^*(t)$. Согласно общей теории, в каждый момент времени из интервала $t \in (t_0, t_1)$ *(mxm)* – матрица $\partial^2 H/\partial u^2$ отрицательно определена или отрицательно полуопределена. В этом и заключается *принцип максимума* для функции Гамильтона.

Из обозначения функции Гамильтона $H(x, u, y, t) = \phi(x, u, t) + y^T \varphi(x, u, t)$ следует, что $\dot{x} = \varphi(x, u, t) = \partial H/\partial y$. Объединяя это соотношение с условиями б) и в) системы (7.17), получим известную систему *канонических уравнений*

$$\text{г)} \quad \dot{x} = \varphi(x, u, t) = \partial H/\partial y,\ x(t_0) = x_0;$$

$$\text{д)} \quad \dot{y} = -\partial H/\partial x,\ y(t_1) = \partial F/\partial x_1. \qquad (7.19)$$

Таким образом, чтобы пользоваться принципом максимума, необходимо ввести n сопряженных переменных (или координат) $y_1(t), \ldots, y_n(t)$, составить функцию Гамильтона

$H(x, u, y, t) = \phi(x, u, t) + y^T \varphi(x, u, t)$ и найти функции $\{u(t)\}$, $\{y(t)\}$ и $\{x(t)\}$, удовлетворяющие условиям

$$\max_{\{u\} \in U} H(x, u, y, t) \text{ при всех } t \in (t_0, t_1);$$

$$\dot{x} = \partial H / \partial y, \quad x(t_0) = x_0;$$

$$\dot{y} = -\partial H / \partial x, \quad y(t_1) = \partial F / \partial x_1. \qquad (7.20)$$

В этой системе имеется $2n$ дифференциальных уравнений; для первых n уравнений заданы начальные граничные условия $x(t_0) = x_0$, а для последних n уравнений – конечные граничные условия $y(t_1) = \partial F / \partial x_1$. Как и в нелинейном программировании, при решении системы (7.20) сперва определяется оптимальный вектор $u^* = u^*(y^*)$, используя далее условия связи, определяется вектор $y^* = y^*(t)$ как функция времени, после чего находится оптимальная траектория $u^*(t)$ как функция времени. После нахождения решения особый интерес представляет вопрос чувствительности оптимального значения целевого

функционала $J^* = J\{u^*(t)\}$. Можно показать (см., например, [.]), что имеют место соотношения

$$\partial J^* / \partial t_0 = -[\phi(x^*, u^*, t)]_{t_0},$$

$$\partial J^* / \partial x(t_0) = y^*(t_0). \qquad 7.21)$$

Последнее из этих условий означает, что оптимальные значения сопряженных переменных $y_1^*(t_0), \ldots, y_n^*(t_0)$ представляют собой коэффициенты чувствительности оптимального значения функционала относительно соответствующих фазовых переменных в начальный момент времени t_0.

Из принципа максимума непосредственно вытекает уравнение Эйлера для вариационной задачи, если учитывать выражение для управления $u = \dot{x}$. Займемся теперь связью между динамическим программированием и принципом максимума. Уравнение Беллмана, записанное в терминах оптимального значения целевого функционала J^*, имеет вид (см. выражение (6.5))

$$\partial J^* / \partial t = -\max_{\{u(t)\}} \left\{ \phi(x, u, t) + (\partial J^* / \partial x)^T \varphi(x, u, t) \right\}. \qquad (7.22)$$

Воспользовавшись вторым условием (7.21), приходим к выводу, что в (7.22) оптимизируемая функция представляет собой функцию Гамильтона, т. е.

$$H(x, u, y, t) = \left\{\phi(x,u,t) + (\partial J^* / \partial x)^T \varphi(x,u,t)\right\} = \\ = \left\{\phi(x,u,t) + y^T \varphi(x,u,t)\right\}, \qquad (7.23)$$

следовательно, уравнение Беллмана можно записать в терминах функции Гамильтона:

$$\partial J^* / \partial t = -\max_{\{u(t)\}}\left\{H(x,u,y,t)\right\}. \qquad (7.24)$$

Согласно этому уравнению, необходимо найти максимум функции Гамильтона относительно управляющих параметров *{u(t)},* то есть требование, содержащее в принципе максимума (см. первое условие (7.20)). Если теперь предположить, что управление *u* максимизирует функцию *H(x, u, y, t),* то (7.24) принимает форму *уравнения Гамильтона – Якоби*

$$\partial J^* / \partial t = -H(x,u,\partial J^* / \partial x, t). \qquad (7.25)$$

Производная по *x* от обеих частей этого уравнения равна выражению

$$\frac{\partial^2 J^*}{\partial x \partial t} = \frac{\partial H}{\partial x} + (\frac{\partial H}{\partial y})' \frac{\partial^2 J^*}{\partial x^2}, \qquad (7.26)$$

с другой стороны, дифференцирование по *x* равенства $\partial J^*/\partial x = y$ приводит к соотношению

$$\dot{y} = (\dot{x})' \frac{\partial^2 J^*}{\partial x^2} + \frac{\partial^2 J^*}{\partial t \partial x}. \qquad (7.27)$$

Из-за непрерывной дифференцируемости функции $J^*(x, t)$, ее смешанные частные производные по *x* и *t* не зависят от порядка дифференцирования, поэтому сравнивая последние два выражения и учитывая граничные условия $J^*(x_1, t_1) = F(x_1, t_1)$ и $y(t_1) = \partial J^*(x_1, t_1) / \partial x = \partial F/\partial x_1$, получим *канонические уравнения принципа максимума*

$$\dot{x} = \partial H/\partial y, \quad x(t_0) = x_0; \\ \dot{y} = -\partial H/\partial x, \quad y(t_1) = \partial F/\partial x_1. \qquad (7.28)$$

Этот вывод подсказывает, что из уравнения Беллмана вытекает выполнение условие *принципа максимума.* Следует, однако, отметить, что из выполнения условия принципа максимума не вытекает выполнение уравнения Беллмана, так как в данном случае не требуется вводить предположение о непрерывной дифференцируемости функции $J^*(x, t)$ (или функции

оптимального поведения $f(x_0, t_0)$, которая равна оптимальному значению функционала $J^*(x, t)$ при $x = x_0$ и $t = t_0)$.

Итак, очевидна внутренняя логическая связь и взаимная обусловленность между этими тремя классическими схемами оптимизации динамических объектов: *уравнением Эйлера для вариационной задачи, функциональным уравнением динамического программирования Беллмана и принципом максимума Понтрягина для общей задачи управления.* Однако исследовательский опыт показывает, *принцип максимума* более плодотворный, т. к. *декомпозирует* задачу нахождения оптимального управления (*задачу синтеза)* на два этапа: определение оптимального управления как функция сопряженных переменных $u^* = u^*(y^*)$ и определение сопряженных переменных как функция времени $y^* = y^*(t)$. При этом уравнение Беллмана всегда приводит к нелинейным дифференциальным уравнениям в частных производных и при аналитическом решении задачи уступает по своей полезности принципу максимума. С точки зрения проблем программирования на ЭВМ и организации вычислительного процесса оба эти

подхода практически одинаковы, т. е. им свойственна одна и та же *алгоритмическая сложность.*

В заключение этого раздела рассмотрим ряд практических динамических задач, для решения которых могут быть полезны идеи *принципа оптимальности Беллмана, принципа максимума Понтрягина и уравнение Эйлера.*

а) Задача оптимального быстродействия. Необходимо перевести объект, описываемый линейным и автономным уравнениям движения, от заданного начального состояния фазовых координат к заданному конечному состоянию за минимальное время. Математическая модель этой задачи имеет вид

$$J = -\int_{t_0}^{t_1} dt \to \max_{\{u(t)\}}, \qquad (7.29)$$

$$\dot{x} = Ax + bu$$

где $x(t) = (x_1(t), \dots, x_n(t))^T$ – вектор фазовых координат, $u(t)$ – кусочно-непрерывная функция управления, $-1 \le u(t) \le -1$, A – (nxn) – матрица, b – $(nx1)$ – вектор, t_0 и t_1 – начальный и конечный моменты времени, $x(t_0)$ и $x(t_1)$ фиксированы. Уравнение движения $\dot{x} = Ax + bu$ говорит о том, что управляемый объект линейный.

Для этой задачи очевидны следующие соотношения: $H = -1 + y^T(Ax + bu)$, $\dot{y} = -\partial H/\partial x = -A^T y$, $\dot{x} = \partial H/\partial y = Ax + bu$. Согласно принципу максимума, оптимальное управление получается в виде $u^* = +1$, если $y^T b > 0$ и $u^* = -1$, если $y^T b < 0$. Такое управление называется *релейным,* а функция $y^T b$ называется функцией переключения. Вектор сопряженных переменных находится из линейного уравнения $\dot{y} + A^T y = 0$. По форме интеграла (7.29) видно, что $\phi(x, u, t) = 1$, поэтому уравнение Беллмана для оптимального значения функционала $J^*(x, t)$ (функция оптимального поведения $f(x, t)$) принимает вид

$$\partial J^*/\partial t = -\max_{\{u(t)\}}\{1 + \partial J^*/\partial x)^T \varphi(x,u,t)\} =$$

$$= -\max_{\{u(t)\}}\{1 + y^T(Ax + bu)\}, \qquad (7.30)$$

где подставлено значение $\partial J^*/\partial x = y$. Легко заметить, что решением этого уравнения является найденное выше *релейное управление* $u^* = +1$, если $y^T b > 0$ и $u^* = -1$, если $y^T b < 0$, где вектор y удовлетворяет уравнению $\dot{y} + A^T y = 0$.

б) Задача управления по минимуму энергии. Необходимо найти оптимальное управление как решение задачи динамической оптимизации

$$J = -\frac{1}{2}\int_{t_0}^{t_1}(x^T Dx + u^T Eu)dt + \frac{1}{2}x^{1T} Fx^1 \to \max_{\{u(t)\}}, \qquad (7.31)$$

$$\dot{x} = Ax + Bu$$

где $x(t) = (x_1(t), \ldots, x_n(t))^T$ – вектор фазовых переменных, $u(t) = (u_1(t), \ldots, u_m(t))^T$ – вектор управления, A – (nxn) – матрица, B – (nxm) – матрица, D и F отрицательно определенные матрицы резмерности (nxn), E – отрицательно определенная матрица размерности (mxm), $x^1 = x(t_1)$ и $x_0 = x(t_0)$ фиксированы. Записывая функцию Гамильтона в виде $H = (x^T Dx + u^T Eu)/2 + y^T(Ax + Bu)$, согласно принципу максимума получим $\partial H/\partial u = Eu + B^T y = 0$, откуда следует решение $u^* = -E^{-1}B^T y$. На основе функции Гамильтона составим канонические уравнения

$$\text{а) } \dot{x} = \partial H/\partial y = Ax + Bu = Ax - BE^{-1}B^T y, \quad x(t_0) = x_0;$$

$$\text{б) } \dot{y} = -\partial H/\partial x = -Dx - A^T y, \quad y(t_1) = \partial F/\partial x_1 = Fx_1. \qquad (7.32)$$

Для сопряженных переменных решение ищется в линейной форме $y = Q(t)x$, где $Q(t)$ - (nxn) – матрица,

элементы которой зависят от времени. Она обычно находится из *уравнения Рикатти* $dQ(t)/dt = QBE^{-1}B^TQ - QA - A^TQ - D$ с граничными условиями $Q(t_1) = F$. После чего оптимальное управление принимает вид $u^*(t) = -E^{-1}B^TQ^T(t)x(t)$. В данной задаче функция под знаком интеграла является квадратичной функцией, а ограничения линейны, поэтому ее решение эквивалентно решению соответствующей задачи квадратичного программирования (см. главу 3 части 1).

Связь с уравнением Беллмана следует из (7.24) и имеет вид

$$\partial J^*/\partial t = \max_{\{u(t)\}} H(x, u, y, t), \qquad (7.33)$$

в) Задача оптимального экономического роста. Известная из экономической теории *неоклассическая модель оптимального экономического роста в замкнутой форме* (т. е. без импорта и экспорта) представляет собой задачу оптимального управления [7]

$$W(c(t)) = \int_{t_0}^{t_1} e^{-\delta(t-t_0)}U(c(t))dt \to \max_{\{c(t)\}}. \qquad (7.34)$$

$$\dot{k} = f(k) - \lambda k - c$$

$$0 \le c(t) \le f(k(t))$$

$$k(t_0) = k_0,$$

В этой задаче в качестве «управления» выступает функция потребления на одного рабочего $c(t) = C(t)/L(t)$, где $C(t)$ – фонд потребления, $L(t)$ – общее число занятых – фактор труда, $k(t) = K(t)/L(t)$ – величина капитала на одного рабочего, $K(t)$ – объем капитала (или фактор капитала), $f(k) = F(k, 1) = y = Y/L$ - валовый выпуск на одного рабочего, $Y = F(K, L)$ – производственная функция, являющаяся по предположению однородной функцией первой степени, т. е. $F(mK, mL) = mF(K, L) = mY$; полагая $m = 1/L$, вместо $F(K, L)$, получим функцию $f(k) = F(k, 1)$; $\lambda = \mu + n$, где μ - норма амортизации капитала (доля выбитого капитала), $n = \dot{L}/L$ - темп роста численности рабочей силы. Дифференциальное уравнение $\dot{k} = f(k) - \lambda k - c$ представляет собой модель Солоу в относительных единицах, т. к. величина $I = Y - C$ представляет собой валовые инвестиции, а уравнение $\dot{K} = I - \lambda K$ характеризует темп роста капитала в экономике (фирменной, национальной и т. д.).

В выражении для подынтегральной функции *U(c(t))* представляет собой функцию полезности величины с*(t)* и удовлетворяет условиям а) $\partial U/\partial c > 0$ (условие положительности *предельной полезности*) и б) $\partial^2 U/\partial c^2 < 0$ *(закон убывания предельной полезности* или *закон Госсена).* Умножение функции *U(c(t))* на $e^{-\delta(t-t_0)}$ под знаком интеграла эквивалентно действию *дисконтирования* (или *приведения*) к моменту времени t_0. Наконец, оптимизируемый функционал $W(c(t))$ представляет уровень благосостояния, соответствующий траектории потребления на одного рабочего *{c(t)}*. Величина $W(c(t))$ определяется путем интегрирования (или суммирования) всех мгновенных полезностей по всему интервалу.

Таким образом, в задаче (7.34) имеется всего одна фазовая координата – уровень потребления и одно уравнение движения – дифференциальное уравнение экономического роста. Задача управления заключается в определении оптимальной траектории потребления, которая максимизирует функционал благосостояния.

При решении задачи (7.34) горизонт планирования обычно распространяется на все будущее, т. е.

предполагается, что $t_1 \to \infty$. При этом, чтобы гарантировать сходимость интеграла, принимается, что фактор дисконтирования δ является положительной величиной, а начальная капиталовооруженность рабочего меньше максимально достижимого уровня $\tilde{k}$. Тогда имеют место неравенства $c(t) \le f(\tilde{k})$ и

$$\int_{t_0}^{\infty} e^{-\delta(t-t_0)} U(c(t))dt \le$$

$$\int_{t_0}^{\infty} e^{-\delta(t-t_0)} U(f(\tilde{k}))dt = U(f(\tilde{k})/\delta. \quad (7.35)$$

Функция Гамильтона для задачи оптимального экономического роста на бесконечном горизонте планирования составляется в виде

$$H(k, c, y, t) = e^{-\delta(t-t_0)} U(c(t)) + y[f(k) - \lambda k - c] =$$

$$= e^{-\delta(t-t_0)} \{U(c(t)) + q[f(k) - \lambda k - c]\}, \quad (7.36)$$

где в качестве сопряженной переменной *y(t)* рассматривается величина $q e^{-\delta(t-t_0)} = y$ (неопределенный множитель Лагранжа). В соответствии с принципом максимума оптимальный уровень потребления на одного рабочего максимизирует функцию *H(k, c, y, t)* в каждый момент времени. Необходимое условие первого порядка

$\partial H/\partial c = 0$ порождает условие для сопряженной переменной

$$q = dU/dc = U'(c(t)). \qquad (7.37)$$

Согласно этому выражению, новая сопряженная переменная равна предельной полезности и является положительной величиной. Так как функция полезности считается строго вогнутой функцией, то условия второго порядка для внутреннего решения удовлетворяются.

Если теперь воспользоваться каноническими уравнениями (6.32), то с учетом соотношения $q\,e^{-\delta(t-t_0)} = y$ получим дифференциальное уравнение $\dot{q} = -(f'(k) - (\lambda + \delta))q$, или что эквивалентно, $\dot{q}/q = -(f'(k) - (\lambda + \delta))$. Учитывая связь (7.37) для оптимальной траектории, это отношение можно переписать в виде

$$\dot{q}/q = -(f'(k) - (\lambda + \delta)) = U''(c)c'/U'. \qquad (7.38)$$

Величина $\sigma = \sigma(c) = cU'(c)/U(c)$, представляющая собой отношение предельной полезности $U'(c)$ к средней полезности $U(c)/c$, называется *эластичностью (или относительной чувствительностью*) функции полезности. Она характеризует величину относительного

изменения функции полезности при единичном относительном изменении величины c, другими словами, характеризует *чувствительность функции U(c).*

Переписав выражение для эластичности σ в виде

$$cU'(c) = \sigma U(c), \qquad (7.39)$$

приходим к известному уравнению Эйлера относительно однородности функции полезности степени σ, определяемой в виде $U(tc) = t^{\sigma}U(c)$, $t > 0$.

Продифференцировав обе части этого выражения по параметру $t > 0$ и полагая в полученном выражении $t = 1$, непосредственно получим соотношение (7.39). Доказательство однородности непосредственно следует из исследования производной функции $\varphi(t) = U(tc)/t^{\sigma}$. Важность этого вывода заключается в том, что, полагая $t = 1/c$, приходим к соотношению $U(c) = U(1)\,c^{\sigma} = U_0\,c^{\sigma}$. В этом выражении σ является функцией c, а $U_0 = U(1)$ - константа. По аналогии с этим, величина $\sigma_u = \sigma_u(c) = -cU''(c)/U'(c)$ называется *эластичностью предельной полезности* и служит показателем кривизны функции полезности. Используя выражение для σ_u, из (7.38)

получим $\sigma_u \ \dot{c}/c = U''(c)c'/U' = -(f'(k)-(\lambda+\delta))$, откуда следует дифференциальное уравнение

$$\dot{c} = (f'(k)-(\lambda+\delta))\,c/\sigma_u, \qquad (7.40)$$

которое, вместе с уравнением $\dot{k} = f(k)-\lambda k - c$, является условием для оптимальной траектории $\{c^*(t),\ k^*(t)\}$. В установившемся состоянии, когда $\dot{c} = \dot{k} = 0$ и $k = k^*$, из этих соотношений получим $c^* = f(k^*) - k^*$, $f'(k) = (\lambda+\delta))$. Читателям, интересесующимся экономической сущностью и подробностями интерпретации этих соотношений, мы рекомендуем работу М. Интрилигатора [6] (глава 16).

Возвращаясь к уравнению Беллмана (6.24), с учетом выражения для функции Гамильтона (7.36) получим соотношение

$$\partial J^*/\partial t = -\max_{\{c(t)\}}\{H(x,c,y,t)\} =$$

$$-\max_{\{c(t)\}}\{e^{-\delta(t-t_0)}[U(c(t)) + q(f(k)-\lambda k - c)]\} \qquad (7.41)$$

Очевидно, что максимум функции Гамильтона порождает выражение $q = U'(c(t))$, следовательно, огласно уравнению Гамильтона – Якоби, уравнение (6.41) примет вид

$$\partial J^*/\partial t = -e^{-\delta(t-t_0)}[U(c(t)) +$$

$$U'(c(t))\ (f(k)-\lambda k - c)]. \qquad (7.42)$$

В этом выражении предельная полезность $U'(c(t))$ не что иное, как $\partial J^*/\partial c$, т. е. она равна производной оптимального значения функционала (функции оптимального поведения) по управляемой переменной во всех точках оптимальной траектории $\{c^*(t)\}$.

На основе функции Гамильтона получим также канонические уравнения в виде

$$\dot{y} = \frac{d}{dt}(e^{-\delta(t-t_0)}q(t)) = -\frac{\partial H}{\partial k} = -\,(f(k) - \lambda)\,e^{-\delta(t-t_0)};$$

$$\dot{k} = \frac{\partial H}{\partial y} = f(k) - \lambda k - c\,.$$

Согласно общей теории, должны иметь место граничные условия $\partial J^*/\partial k(t_0) = y^*(t_0)$, $J^*(k_1,\ t_1) = F((k_1,\ t_1) = 0$, $y(t_1) = \partial J^*(\ k_1,\ t_1)/\ \partial k = \partial F/\partial k_1 = 0$. Эти условия должны выполняться для произвольного значения верхнего предела интеграла в выражении целевого функционала t_1, в том числе и при $t_1 = \infty$.

4.8. Задачи и вопросы по ДП

Задача 1. Решение инвестиционной задачи.

Коммерческой фирме необходимо инвестировать 15 единиц денежных средств (например, 15 млн. руб.) в три производственные (или инновационные) программы, функции ожидаемых доходов от которых представлены в таблице. Необходимо найти оптимальную стратегию финансирования программ, которая максимизирует суммарный доход фирмы.

Исходные данные

x	0	1	2	3	4	5	6	7	8 .. 15
$R_1(x)$	0	2.1	3.5	5	7	9	9	9.	9 …9
$R_2(x)$	0	2.5	2.8	4.8	8.5	10.5	12	12	12 ... 12
$R_3(x)$	0	2.8	4	4.5	10	11	14	16	17... 17

а) Построить функциональное уравнение динамического программирования и с его помощью решить задачу;

б) Как изменится решение, если выделенный для финансирования программ объем денежных средств превысит 20 единиц?

в) Сравнить объемы вычислений, которые соответствуют методу полного перебора и решению на основе принципа оптимальности:

г) Построить графическое изображение шагам работы алгоритма, его прямому и обратному ходу;

д) Как изменится оптимальная стратегия, если первой считать вторую программу (третью программу)?

е) Оценить ожидаемый доход фирмы при учете вероятности риска: $p_1 = 0,15$, $p_2 = 0,2$, $p_3 = 0,25$.

Задача 2. Определение наилучшей стратегии замены оборудования.

Технологическому подразделению фирмы необходимо определить наилучшую стратегию эксплуатации и замены оборудования в течение заданного горизонта планирования, состоящего из N отрезков (например, N лет). В начале первого отрезка принимается решение эксплуатировать оборудование до начала j-ого отрезка, $j \le N$, потом заменить его. Если $j < N$, то старое оборудование заменяется в начале отрезка j новым оборудованием, которое эксплуатируется до начала отрезка $k \le N$ и заменяется, и т.д. В качестве критерия

выбора решения рассматривается функция полных эксплуатационных затрат.

Обозначая через c_{ij} полные эксплуатационные затраты, начиная с начала отрезка *i*, когда оборудование приобретается, и кончая отрезком *j*, на котором оно заменяется, $1 \le i < j \le N$, f_n - уровень наименьших эксплуатационных затрат на отрезках $n, n+1,..,N$, процесс поиска оптимального решения можно описать с помощью функционального уравнения

$$f_n = \min\{c_{n\kappa} + f_\kappa\}, n = N, N-1,..,1,$$

$$\kappa = n+1,..., N+1$$

с начальным значением $f_{N+1} = 0$.

Построить алгоритм решения задачи и найти оптимальное решение с помощью данных для c_{ij} (расходы на обслуживание, ремонт, приобретение нового оборудования с учетом остаточной стоимости старого). Построить граф состояний и показать на нем оптимальный путь, соответствующий стратегии эксплуатации.

Матрица С

1	- 120	110	90	130	250
2	-	70	90	110	230
3		-	80	100	210
4			-	50	160
5				-	100
6					

Задача 3. Стратегия замены с учетом возраста оборудования. В предыдущей задаче возраст оборудования как фактор решения не учитывался. Однако, с целью более адекватного моделирования реальной ситуации, приходится включить его в процесс анализа и оптимизации решений в явном виде.

Обозначим через P_{in} стоимость замены оборудования возраста *i* на отрезке *n* новым оборудованием с учетом продажи старого оборудования по остаточной стоимости; k_{ni} - стоимость эксплуатации оборудования на отрезке *n*, возраст которого в конце отрезка будет равен *i*. Состояние процесса в начале каждого отрезка обозначим двойкой *(i, n),* а через $f_n(i)$ обозначим минимальную величину эксплуатационных

затрат на отрезках *n, n+1,…., N,* при условии, что в начале отрезка *n* возраст оборудования равен *i.*

Для произвольного состояния *(i, n)* значение $f_n(i)$, очевидно, будет определяться согласно правилу

$$f_n(i) = \min_{(A,B)} \begin{cases} A = \kappa_{n,i+1} + f_{n+1}(i+1), \\ B = P_{in} + \kappa_{n1} + f_{n+1}(1), \end{cases}$$

Величина *A* соответствует затратам, связанным с решением «*эксплуатировать»,* а *B*- с решением «*заменить*». Необходимые данные для заданного горизонта планирования *N = 6* лет привезены в виде таблицы *(k, p).*

Построить граф решения задачи с изображением всех вершин и дуг.

	$\rightarrow_j$				Значения Р	
↓i	95	250	240	250	275	240
	90	100	270	260	280	250
Знач. К	85	90	95	280	280	300
	80	85	100	150	300	305
	70	80	90	100	150	310
	75	80	84	100	110	200

Построить алгоритм решения задачи с помощью приведенного выше функционального уравнения для $f_n(i)$, *n = N, N-1,.., 1.* Выделить оптимальную траекторию (стратегию) и установить затраты, связанные с эксплуатацией каждого оборудования.

Задача 4. Оптимальное резервирование с учетом стоимости и веса оборудования.

Проектируемое оборудование состоит из *N* последовательно соединенных звеньев, в каждом из которых работают по одному основному элементу. Заданы стоимость c_j, вес w_j и вероятность безотказной работы p_j каждого элемента *j* –ого типа, *j = 1, … , N*. Для повышения надежности работы оборудования предусматривается резервирование основных элементов всех звеньев. Имеются ограничения по весу (*W*) и стоимости (*C*) резервирования. В предположении, что функция надежности каждого звена определяется формулой $r_i(m_i) = 1 - (1 - p_j)^{1+m_j}$, $j = 1,\ldots,N$, а надежность оборудования равна произведению функций надежности звеньев, т.е. $R(m) = r_1(m_1)\ldots r_N(m_N)$, найти оптимальную

стратегию резервирования, отвечающую заданным ограничениям. Данные для пяти вариантов решения задачи приведены в таблице.

Исходные данные

j	1	2	3	4	5
p_j	07	0.9	0.8	0.6	0.5
c_j	5	7	10	4	6
w_j	3	2	4	3	5

Варианты	1	2	3	4	5
C	40	45	35	50	55
W	20	25	30	25	30

Построить математическую модель задачи и на основе принципа оптимальности вывести функциональное уравнение динамического программирования. Решить задачу резервирования при условии, что в задаче имеется лишь одно ограничение – по стоимости или по весу, затем сравнить результаты и объяснить расхождение.

Построить логические схемы для изображения проектируемого оборудования с резервными элементами, многошагового процесса, а также прямого и обратного хода алгоритма. Вариант резервирования при наличии обоих ограничений программировать и решить задачу на ЭВМ и сравнить результаты с аналогичными результатами предыдущих вариантов, начиная процесс поиска со значения неопределенного множителя Лагранжа $\lambda = 0{,}01$. При каких значениях параметров C и W можно с уверенностью утверждать, что оптимальное значение λ равно нулю?

*Задача 5***.** Задача м*инимизации суммы квадратов при фиксированном значении их суммы*. В ряде приложений по теории управления, обработке данных и моделированию возникает математическая задача, представленная в виде

$$R(u) = \sum_{\kappa=1}^{N} u_\kappa^2 \to \min,$$

$$\sum_{\kappa=1}^{N} u_\kappa = C$$

$$u_\kappa \geq 0, \ \forall \kappa$$

где C – фиксированная величина (например, общий ресурс).

Построить функциональное уравнение динамического программирования и с его помощью решить задачу. Решить задачу с помощью метода множителей Лагранжа и сравнить результаты с решением, найденным по методу динамического программирования. Оценить сложность вычислений по этим двум методам. Как изменится решение, если в целевой функции величины $u_к^2$ умножить на соответствующие весовые коэффициенты $\alpha_к$, $к = 1, ..., N$?

Задача 6. Оптимальное управление одномерным объектом описывается с помощью математической задачи

$$R(u) = \int_{t_o}^{t_1} ((x-u)^2 + u^2)dt \to \min,$$

$$\{u(t)\}$$

$$dx/dt = ax + bu$$

$$x(t_o) = x_o; x(t_1) = x_1$$

где a, b, x_0 и x_1 фиксированные параметры. Построить уравнение Беллмана в частных производных (см. формулу (7.5)).

Задача 7. В предположении, что функция полезности аппроксимирована в виде формулы $u(q) = q_1q_2...q_n$, $q_k \geq 0$, $k = 1,..., n$, решить задачу потребительского выбора, заданную в виде

$$u(q) = \prod_{k=1}^{n} q_k \to \max,$$

$$(q_{1,...,}q_n)$$

$$\sum_{k=1}^{n} p_k q_k = C$$

с помощью метода динамического программирования и сравнить полученный результат с аналогичным результатом, полученным с помощью метода Лагранжа.

Оценить влияние величины бюджета C и рыночных цен $p_1,..., p_n$ на искомое решение. Оценить вычислительную сложность, связанную с решением данной задачи с помощью динамического программирования и метода множителей Лагранжа.

Глава 5. Численные методы оптимизации

5.1. стратегия поиска и условия сходимости

При решении практических задач оптимизации с помощью методов математического программирования нам не всегда удается найти решение с помощью аналитических методов и алгоритмов, как это удается, например, с помощью *симплекс – метода* в линейном программировании или *метода Лагранжа* в нелинейном программировании. Сложность и нелинейность приводят к тому, что в общем случае нахождение искомого решения или некоторого приемлемого приближения (или аппроксимации) находят с помощью различных численных методов поиска и аппроксимации.

Современный исследовательский опыт показывает, что подавляющее большинство таких методов основано на весьма простом рекуррентном соотношении вида

$$x^{k+1} = x^k + \alpha_k d^k, \qquad (1.1)$$

где x^k - текущая аппроксимация (или приближение), d^k - *«удачно» выбранное направление движения,* α_k - *«удачно» выбранная величина шага вдоль направления* d^k,

$x^{k+1} = x^k + \alpha_k d^k$ - следующая «удачная» точка на пути к искомому (но обязательно существующему) решению x^*.

В настоящее время существует большой арсенал разнообразных поисковых методов и процедур, которые отличаются друг от друга лишь правилами выбора направления движения (или направления поиска) и величины шага. Особую группу среди них составляют так называемые *методы случайного поиска,* о которых пойдет речь в соответствующем разделе *стохастического программирования*. Здесь же мы остановимся на численных методах условной оптимизации, представляющих практический интерес.

При систематизированном изучении численных методов их обычно классифицируют с помощью различных признаков, таких как *наличие или отсутствие ограничений, использование информации о производных целевой функции, учет случайных механизмов или без них и т. д.* Независимо от выбранного признака, поисковые методы и процедуры порождают последовательность решений (или приближений) $\{x^k\}$, $k = 0, 1, \ldots$, которая должна обладать рядом полезных свойств, в том числе и сходимостью к искомому решению x^*. При этом каждое

последующее решение выбирается либо на основе конкретного точечно-множественного отображения $x^{k+1} \in W(x^k)$, либо на основе точечно-точечного отображения $x^{k+1} = W(x^k)$, где $W(x)$ – выбранное алгоритмическое отображение [B,Sh]. Пусть, для определенности, решается задача

$$(D, f) : f(x) \to \min_{x \in D}, \qquad (1.2)$$

где $D \subseteq E^n$ – область допустимых решений, $f(x)$ – целевая функция, x – оптимизируемый вектор, а $X^* \subset D$ – непустое подмножество оптимальных решений этой задачи. Условия, гарантирующие сходимость последовательности $\{x^{k+1}\}$ к некоторой точке (решению) $x^* \in X^*$, сводятся к условиям следующей теоремы [B,Sh].

Теорема. Пусть на множестве D задано точечно-множественное отображение $W(x^*)$, $X^* \neq \varnothing$. Если

а) все точки $x^{k+1} \in W(x^k)$ образуют компактное множество, включенное в D;

б) для любого x^{k+1} при $x^{k+1} \neq x^*$ имеет место $f(x^{k+1}) \leq f(x^k)$;

в) при $x^k = x^*$ алгоритм либо останавливается, либо для любого $x^{k+1} \in W$ имеет

место $f(x^{k+1}) = f(x^k) = f(x^*)$;

г) отображение $W(x^k)$ замкнуто в x^k при $x^k \neq x^*$,

тогда алгоритм либо останавливается в точке x^*, либо имеет место $\lim_{k \to \infty} x^k = x^* \in X^*$. Последовательность $\{x^{k+1}\}$ при этом называется *релаксационной последовательностью*.

Заметим, что замкнутость алгоритмического отображения в общем случае сводится к следующему. Пусть X и Y - замкнутые непустые подмножества пространств E^p и E^q соответственно, $\{x^k\}$ и $\{y^k\}$ - сходящиеся к точкам x^* и y^* соответственно, и выполняется условие $y^k \in W(x^k)$. Алгоритмическое отображение W называется замкнутым, если $x^* \in W(y^*)$. Доказательство сходимости алгоритмов, основанных на замкнутых алгоритмических отображениях, приводится в конце главы, в разделе 5.6.

5.2. Методы безусловной оптимизации

а) Методы возможных направлений. Следует отметить, что данная группа методов составляет основу для всех остальных поисковых методов и процедур.

Сущность этих методов сводится к следующему. Рассмотрим решение задачи минимизации в форме

$$f(x) \to min, \qquad (2.1)$$
$$x \in E^n$$

где $x = (x_1, \ldots, x_n)^T$ – произвольный вектор, принимающий значения из n – мерного евклидова пространства E^n, $f(x)$ – минимизируемая функция, $f: E^n \to E^1$, X^* - множество решений задачи, причем $X^* \neq \varnothing$. Мы будем рассматривать лишь задачу на минимум, учитывая возможность замены задачи на максимум на задачу на минимум: $arg\ max\ f(x) = arg\ min\ (-f(x))$. Пусть ряд решений (или приближений) $x^0, x^1, \ldots, x^k$ к некоторой точке $x^* \in X^*$ уже известен. Следующее приближение находим из *рекуррентного соотношения*

$$x^{k+1} = x^k + \alpha_k d, \qquad (2.2)$$

где d^k - возможное направление движения единичной длины, т. е. $\| d^k \| = 1$ (через $\|z\| = (z^T z)^{1/2}$ обозначается норма (или длина) вектора, а $z^T z$ – скалярное произведение вектора на себя), α_k – возможный шаг по этому направлению, $\alpha_k \geq 0$. В предположении, что длина направления d^k равна единице, получим

$$\| x^{k+1} - x^k \| = \| \alpha_k d^k \| = |\alpha_k| \, \| d^k \| = |\alpha_k|. \qquad (2.3)$$

Это означает, что длина вектора $x^{k+1} - x^k$ равна величине положительного шага по выбранному направлению движения. Когда ограничения на переменную величину отсутствуют, любое направление движения является допустимым, но для задачи минимизации целевой функции $f(x)$ желательными, или «удачными», являются лишь те направления, движение вдоль которых приводит к уменьшению значения $f(x)$, т. е. необходимо выполнение условия $f(x^0) < f(x^1) < \ldots f(x^k) < f(x^{k+1})$.

Вопрос выбора «удачного» направления можно решить с помощью понятия градиента функции $f(x)$, который определяется как вектор $\nabla f(x) = (\partial f / \partial x_1, \ldots, \partial f / \partial x_n)^T$ размерности $(nx1)$. Пусть теперь точка x принадлежит некоторой ε - окрестности текущей точки (или приближения) x^k, т. е. $x \in B(x^k, \varepsilon) = \{x / \|x - x^k\| < \varepsilon\}$. Разлагая функцию $f(x)$ в ряд в этой точке, получим приближенное представление

$$f(x) \cong f(x^k) + \nabla f(x^k)^T (x - x^k). \qquad (2.4)$$

Из этого представления следует важный вывод о том, что для выполнения условия $f(x) < f(x^k)$ необходимо, чтобы имело место неравенство

$$\nabla f(x^k)^T (x - x^k) < 0. \qquad (2.5)$$

Это и есть условие выбора «удачного» направления перемещения из точки x^k таким образом, чтобы значения функции $f(x)$ уменьшались, т. е. скалярное произведение направления движения на градиент функции $f(x)$ должно быть отрицательно. Полагая $x = x^k + \alpha_k d^k$, из (2.4) получим

$$f(x^k + \alpha_k d^k) \cong f(x^k) + \nabla f(x^k)^T d^k \alpha_k, \ \alpha_k \geq 0. \qquad (2.6)$$

Таким образом, условие (2.5) эквивалентно условию

$$\nabla f(x^k)^T d^k < 0. \qquad (2.7)$$

Направление, удовлетворяющее этому условию, называется *приемлемым* в состоянии x^k, следующую точку $x = x^{k+1}$ можно взять на этом направлении, т. е. $x^{k+1} = x^k + \alpha_k d^k$, что и объясняет суть рекуррентной формулы (2.2). Ниже мы рассмотрим ряд простых и, вместе с тем, практичных правил выбора и направления, и величины шага движения по нему.

б) Градиентный метод. Сущность градиентного метода заключается в том, что в формуле для построения следующей удачной точки $x^{k+1} = x^k + \alpha_k d^k$ в качестве приемлемого направления выбирается направление *антиградиента*, т. е. $d^k = -\nabla f(x^k)$, а в случае задачи на максимум выбирается направление градиента, т. е. $d^k = \nabla f(x^k)$. Тогда основное рекуррентное соотношение для задачи (1.1) принимает вид

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k). \qquad (2.8)$$

В качестве удачной (или оптимальной в текущем состоянии) величины шага α_k разумно выбрать величину на основе правила

$$\alpha_k = \arg\min_{\alpha > 0} f(x^k - \alpha \nabla f(x^k)). \qquad (2.9)$$

Такой метод выбора величины шага получил название *метода наискорейшего спуска* (в случае задачи на максимум – *метода наискорейшего подъема).* Термин «*наискорейший»* означает, что движение происходит по направлению наискорейшего убывания (или возрастания в случае максимума) целевой функции, которое является направлением антиградиента (или градиента для задачи на максимум).

В практических задачах часто используется модификация градиентного метода, связанная с выбором в качестве направления

$$d^k = - A_k \nabla f(x^k),\ k = 0, 1, 2, \ldots, \qquad (2.10)$$

где A_k – диагональная матрица с элементами $a_{ii} = \sqrt{\rho_i}$, $i = 1, \ldots, n$, $a_{ij} = 0$, $\forall\ i \neq j$, ρ_i - некоторый неотрицательный параметр, служащий для изменения масштаба. Для выпуклой и дважды непрерывно дифференцируемой функции $f(x)$ обычно принимается $\rho_i = \partial^2 f/\partial x i^2$.

Рассмотрим в качестве примера задачу минимизации квадратичной функции $f(x) = x^T Hx/2$, где H – симметричная положительно определенная матрица размерности (nxn). Безусловный минимум этой функции имеет место в точке $x^* = 0$, так что $f(x^*) = 0$. Градиент этой функции равен $\nabla f(x) = Hx$, следовательно, в произвольной точке x^k получим $\nabla f(x^k) = Hx^k$. Поэтому в качестве направления необходимо выбрать $d^k = - \nabla f(x^k), = - Hx^k,\ k = 0, 1, 2, \ldots$. Выбирая в качестве аргумента оптимизируемой функции задачи (2.9) вектор $x = x^k - \alpha Hx^k$ и учитывая очевидное преобразование

$$(x^k - \alpha Hx^k)^T H(x^k - \alpha Hx^k) = x^{kT}(H - 2\alpha H^2 + \alpha^2 H^3)x^k, \qquad (2.11)$$

получим следующую задачу выбора оптимальной величины шага:

$$\alpha_k = \arg\min_{\alpha > 0} f(x^k - \alpha \nabla f(x^k)) =$$

$$\arg\min_{\alpha > 0} \{x^{kT}(H - 2\alpha H^2 + \alpha^2 H^3)x^k/2\}. \qquad (2.12)$$

Так как при $x \neq 0$ имеет место $x^{kT} H^3 x^k \neq 0$, то минимум в правой части (2.12) достигнет при $\alpha_k = x^{kT} H^2 x^k / x^{kT} H^3 x^k = z^T z / z^T Hz$, где принято обозначение $z = - d^k = Hx^k$. Новое решение и новое значение целевой функции будут равны соответственно

$$x^{k+1} = x^k - \alpha_k Hx^k = x^k - (z^T z/z^T Hz) Hx^k,$$

$$f(x^{k+1}) = (x^{k+1})^T H x^{k+1}/2 = z^T H^{-1} z - (z^T z)^2 / z^T Hz \geq 0.$$

Как следует из выражения $d^k = - Hx^k$, если в качестве приемлемого направления движения выбрать $d^* = H^{-1} d^k = -H^{-1} Hx^k = - x^k$, тогда при $\alpha_k = 1$, независимо от текущей точки x^k, новая точка x^{k+1}, а за ней и значение функции $f(x)$ равнялись бы нулю, другими словами, задача была бы решена за одну итерацию. Это - одна из важных особенностей квадратичных функций. Другая важная

особенность градиентного метода заключается в том, что из-за выбора точки x^{k+1} на основе правила (2.12), векторы $p^k = x^{k+1} - x^k$ составляют зигзагообразный путь, причем все они попеременно параллельны друг другу, т. е. $p^0 \,//\, p^2 \,//\, p^4 \ldots$, $p^1 \,//\, p^3 \,//\, p^5 \ldots$(см. рис.5.1). Этот факт лежит в основе метода параллельных касательных и используется для «*ускорения*» градиентного метода.

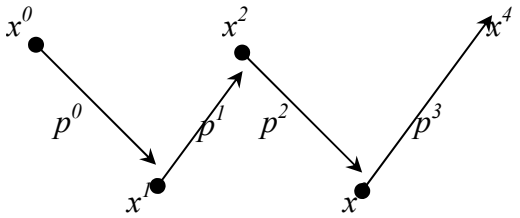

Рис. 5.1. Зигзагообразный путь с попеременно параллельными направлениями.

в) Метод сопряженных направлений. Оказывается, что градиентный метод поиска можно существенно улучшить, используя технику сопряжения направлений.

Определение. Два направления y и z из E^n называются C – сопряженными, если их скалярное произведение равно нулю, т. е. $y^T C z = 0$. Если это условие

выполняется для всех $z \in M \subset E^n$, тогда y называется *сопряженным* с подпространством $M \subset E^n$. Понятие сопряженности, таким образом, является обобщением понятия ортогональности (условие $y^T z = 0$ означает ортогональность этих векторов в E^n).

Рассмотрим решение задачи

$$f(x) = a^T x + (1/2) x^T C x \to \min, \qquad x \in E^n \tag{2.13}$$

где C – симметрическая положительно определенная матрица порядка (nxn) (следовательно, $f(x)$ является строго выпуклой функцией). Введем обозначение $p^k = x^{k+1} - x^k$, $k = 0, 1, 2, \ldots$С помощью алгоритма, который приводится ниже, приближения $x^0, x^1, x^2, \ldots$, выбираются так, чтобы выполнялось условие

$$p^{kT} C p^q = 0, \ \forall k \neq q. \tag{2.14}$$

Алгоритм сопряженных направлений.

Шаг 1. Выбрать начальную точку x^0, выполнить одномерный поиск минимума функции $f(x)$ вдоль направления p^0 и принять найденное решение за x^1, положить далее $k = 1$ и перейти к шагу 2.

Шаг 2. Из точки x^k по направлению p^k произвести минимизацию функции $f(x)$, где p^k сопряжено с

направлениями p^0, p^1, ... , p^{k-1}. Найденную точку считать x^{k+1}.

Шаг 3. Положить $k := k + 1$ и перейти к шагу 2.

Полученная в результате работы алгоритма точка x^n, где n – размерность пространства E^n, представляет собой решение задачи (2.13). Обычно последовательность $\{p^k\}$ заранее не известна, поэтому построение p^k и одномерная минимизация функции $f(x)$ вдоль этого направления можно осуществить совместно.

Основанием для эффективности данного подхода служит следующая лемма.

Лемма. Если функция $f(x)$ достигает на подпространстве $M \subset E^n$ своего минимума в точке x, а в подпространстве $N \supset M$ – в точке y, то вектор $y - x$ оказывается сопряженным с подпространством M, т. е. $(y - x)^T Cz = 0, \ \forall z \in M$. Действительно, для этих точек x и y имеют место условия $\nabla f(x)^T z = 0; \ \nabla f(y)^T z = 0, \ \forall z \in M$, следовательно, $(\nabla f(x) - \nabla f(y))^T z = 0, \ \forall z \in M$. Для функции (2.13) последнее условие эквивалентно условию $(Cx - Cy)^T z = 0, \ \forall z \in M$, и вектор $x - y$ оказывается сопряженным с подпространством M.

На рис. 5.1 приведена иллюстрация метода C – сопряжения для случая, когда N представляет собой координатную плоскость E^2, а M – суть линию L, принадлежащую E^2. Точка x_0 минимизирует функцию $f(x)$ на линии L, а точка $x^* = 0$ является точкой минимума этой же функции на E^2. Векторы x_0 и $(x^* - x_0)$ оказываются C – сопряженными. На рисунке изображены также линии уровня функции $f(x)$.

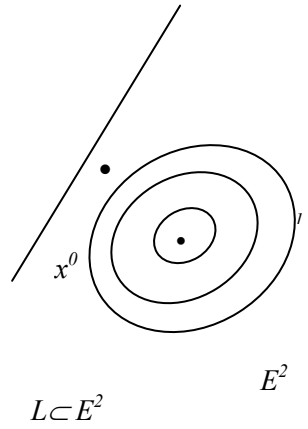

Рис. 5.2. Иллюстрация C - *сопряженности* векторов x_0 и $(x^* - x_0)$

Оказывается, что при заданном множестве из n C – сопряженных векторов, минимум квадратичной функции $f(x)$, для которой C представляет собой ее матрицу Гессе, состоящую из вторых частных производных, может быть достигнут не более чем за n шагов от любой точки, если выполнить ряд одномерных поисков вдоль каждого из сопряженных направлений, как в приведенном выше алгоритме.

Для функции, не являющейся квадратичной, рекомендуется пользоваться ее квадратичной аппроксимацией. В случае, если матрица C не удовлетворяет условию положительной определенности, данный алгоритм может и не сходиться. С практической точки зрения важным частным случаем для метода сопряженных направлений является метод сопряженных градиентов, который мы рассмотрим в следующем пункте.

г) Метод сопряженных градиентов (алгоритм Флетчера – Ривса, 1964 г.). Пусть минимизируется функция $f(x) = (1/2)x^TCx,\ x \in E^n,$ где C – симметричная положительно определенная матрица. Введем обозначения $S^k = -\nabla f(x^k),\ r^k = -S^k = \nabla f(x^k)$ и построим C -

сопряженные направления таким образом:

$$d^0 = S^0 = -\nabla f(x^0),$$
$$d^k = S^k + t_{k-1}d^{k-1},\ k = 1, 2, \ldots \quad (2.15)$$

где параметр t_{k-1} определяется из условия C – сопряженности направлений d^k и d^{k-1}, $k = 1, 2, \ldots$, так что

$$d^{kT}Cd^{k-1} = 0,\ k = 1, 2, \ldots \quad (2.16)$$

Подставляя в это условие выражение (2.15) для d^k, получим

$$t_{k-1} = r^{kT}Cd^{k-1}/d^{(k-1)T}Cd^{k-1},\ k = 1, 2, \ldots \quad (2.17)$$

Выполняя далее одномерную минимизацию функции $f(x)$ по направлениям d^k, согласно правилу

$$\alpha_k = \arg\min f(x^k + \alpha d^k), \quad (2.18)$$
$$\alpha \geq 0$$

находим величину оптимального шага, равную $\alpha_k = -r^{kT}d^k/d^{kT}Cd^k$, вместе с ней и следующую удачную точку $x^{k+1} = x^k - \alpha_k d^k,\ k = 0, 1, 2, \ldots$ Можно показать также справедливость соотношений $r^{k+1} = r^k + \alpha_k Cd^k,\ \alpha_k = -$

$r^{kT}r^k/r^{kT}Cd^k$, которые непосредственно следуют из полученных выше выражений.

Согласно этим соотношениям, после выбора начального приближения x^0, начального направления $d^0 = - r^0 = -\nabla f(x^0)$ и расчета величины α_0 по правилу $\alpha_k = arg\ min\ f(x^k + \alpha d^k)$, получим $x^1 = x^0 - \alpha_0 C\ d^0$. Новое направление d^1 строим, согласно формуле $d^1 = -r^1 + t_0 d^0$; оно обязательно удовлетворяет условию C – сопряженности $d^{1T}Cd^0 = 0$. Теперь $r^1 = r^0 + \alpha_0 Cd^0$ и вдоль луча $x^1 + \alpha\, d^1$ ищем величину шага по правилу $\alpha_1 = arg\ min\ f(x^1 + \alpha d^1)$, и т. д. Поскольку x^1 находится как точка минимума функции $f(x)$ на луче $x^0 + \alpha\, d^0$, векторы $r^1 = \nabla f(x^1)$ и d^0 оказываются ортогональными, т. е. $r^{1T}d^0 = 0$.

д) Метод Ньютона – Рафсона. Эффективность поисковых алгоритмов существенно зависит от объема используемой информации о минимизируемой функции $f(x)$ и ее производных. Алгоритм Ньютона – Рафсона строится на основе информации о производной второго порядка. Идея метода заключается в следующем. Пусть точки x^k, $k = 0, 1, 2, \ldots$, уже сгенерированы, и $B(x^k, \varepsilon)$ - ε -

окрестность точки x^k. Для произвольной точки $x \in B(x^k, \varepsilon)$ воспользуемся приближенным представлением

$$f(x) \cong f(x^k) + \nabla f(x^k)^T(x - x^k) + (1/2)(x - x^k)^T H(x - x^k), \quad (2.19)$$

где H – матрица размерности (nxn), элементы которой равны вторым частным производным функции $f(x)$. Если теперь минимизировать функцию (2.19), то, очевидно, получим уравнение

$$\nabla f(x^k) + H(x - x^k) = 0. \quad (2.20)$$

В предположении, что обратная функция H^{-1} существует, из (2.20) находим $x - x^k = -H^{-1}\nabla f(x^k)$, откуда, принимая x за следующую удачную точку x^{k+1}, получим формулу

$$x^{k+1} = x^k - H^{-1}\nabla f(x^k), \quad (2.21)$$

откуда следует, что в рекуррентной формуле $x^{k+1} = x^k + \alpha_k d^k$ в качестве составляющей $\alpha_k d^k$ можно взять $\alpha_k d^k = - H^{-1}\nabla f(x^k)$. В этом и заключается особенность данного метода и построенного на его основе алгоритма. Единственные ограничения метода – это существование непрерывных частных производных первого и второго

порядка функции $f(x)$, существование обратной матрицы H^{-1} и выполнение условия $\nabla f(x^k) \neq 0$. Элементы самой матрицы H равны соответственно

$$h_{ij} = \partial^2 f / \partial x_i \partial x_j, \quad i, j = 1, \ldots, n. \qquad (2.22)$$

В качестве правила остановки обычно используется выполнение неравенства

$$\| x^{k+1} - x^k \| \leq \varepsilon, \qquad (2.23)$$

где ε - заданная точность аппроксимации (или приближения) искомого решения x^*.

Известно, что в общем случае данный алгоритм не сходится к стационарной точке при произвольном начальном приближении x^0, однако если начальная точка «достаточно» близка к искомому оптимальному решению (к точке минимума), то можно построить подходящее правило наискорейшего спуска, при котором последовательность $\{x^k\}$ будет сходиться к стационарной точке.

5.3. Методы условной оптимизации

Наиболее распространенная форма *задачи условной оптимизации – задачи математического программирования* есть

$$(D, f): f(x) \to \min, \qquad (3.1)$$

$$q_i(x) \leq 0, \; i = 1, \ldots, m$$

$$h_i(x) = 0, \; i = 1, \ldots, l$$

$$x \in X \subseteq E^n$$

где $x = (x_1, \ldots, x_n)^T \in E^n$ - вектор, координаты которого представляют собой переменные задачи, $D \subset E^n$ – множество допустимых решений, заданное с помощью ограничений $q_i(x) \leq 0, \; i = 1, \ldots, m, \; h_i(x) = 0, \; i = 1, \ldots, l$, X - подмножество E^n, на котором определены все функции $f(x)$, $q_i(x)$ и $h_i(x)$.

При задании функций ограничений мы намеренно отказались от общей их формы в виде $q_i(x) \leq b_i$, $h_i(x) = d_i$, учитывая то обстоятельство, что численные методы условной оптимизации в основном исходят из формы задачи (3.1), к тому же общая форма ограничений всегда допускает их преобразование в виде ограничений задачи (3.1).

Без преувеличения можно сказать, что наиболее изящным среди всех численных методов решения классической задачи (3.1) является геометрический вариант симплекс метода линейного программирования, когда, вместо табличного варианта решения, последовательность сгенерированных вершин множества решений D соответствующей канонической задачи строится на основе рекуррентного соотношения

$$x^{k+1} = x^k + \alpha_k d^k. \qquad (3.2)$$

где x^k – текущая вершина, d^k – выбранное направление движения, которое совпадает с одним из ребер D, α_k – допустимая симплексными преобразованиями величина шага по направлению d^k, x^{k+1} - следующая *удачно сгенерированная* вершина D.

Ввиду большого познавательного значения этих построений, данный раздел мы сознательно начинаем с иллюстрации численного решения задачи линейного программирования, напоминая, что ее каноническая форма для задачи на максимум есть

$$(D, f): f(x) = c^T x \to \max, \qquad (3.3)$$
$$Ax = b$$
$$x \geq 0,$$

где x, c – $(nx1)$ - векторы, A – (mxn) – матрица, b – $(mx1)$ – вектор, $m < n$, причем матрица A допускает представление в виде $A = [a_1, a_2, \ldots, a_j, \ldots, a_n] = [B, N]$, где B - (mxm) – матрица полного ранга (т. е. состоящая из m линейно независимых строк или столбцов), N – $(mx(n-m))$ – матрица, I_B - подмножество индексов векторов - столбцов матрицы A - входящих в B, I_N - подмножество индексов входящих в N векторов матрицы A. Для определенности предположим, что линейно независимыми являются первые m векторы $a_1, a_2, \ldots, a_m$ матрицы A, следовательно, $B = [a_1, \ldots, a_m]$, $N = [a_{m+1}, \ldots, a_n]$, $I_B = \{1, \ldots, m\}$, $I_N = \{m+1, \ldots, n\}$ (в противном случае пришлось бы их просто перенумеровать, вместе с переменными $x_j, j = 1, \ldots, n$).

Как и в линейном программировании, матрица B называется *базисом*, а решение $x_B = B^{-1}b \geq 0$ с m координатами $x_{i\delta}$, $i = 1, \ldots, m$, - *допустимым базисным решением*. В этом решении матрица B^{-1} существует, т. к. $rank\ B = m$. Важно отметить, что входящие в N векторы единственным образом могут быть представлены в виде *линейной комбинации* базисных векторов $a_1, a_2, \ldots, a_m$, т.е.

$$a_j = \sum_{i \in I_B} y_{ij} a_i \ , \ \forall_j \in I_N \, , \qquad (3.4)$$

где $y_{ij} \geq 0$, $i \in I_B$ - коэффициенты разложения. Благодаря этому представлению (или разложению), матрица $B = [a_1,...,a_m]$ является (или служит) базисом для системы векторов a_j, $j = 1,..., n,$ каждый из которых имеет размерность $(m \times 1)$.

В разделе линейного программирования было доказано, что точка пространства *D,* определенная в виде

$$\bar{x} = \begin{bmatrix} x_B \\ 0 \end{bmatrix} = \begin{bmatrix} B^{-1} \cdot b \\ 0 \end{bmatrix} \begin{matrix} m \\ n-m \end{matrix}, \qquad (3.5)$$

является одной из вершин *D,* которая *«порождена»* базисом *B* и базисным решением $x_B = B^{-1} b \geq 0$. Справедливо также обратное утверждение: если какая-либо вершина *D* представлена в виде (3.5), то те из векторов $a_1,...,a_n$, которые соответствуют положительным координатам точки $\bar{x}$, являются линейно независимыми. В частности, пусть ровно *к* координат вектора $\bar{x}$, $k \leq m$,

положительны, т.е. $x_{i\delta} > 0$, *i = 1,..., k,* а векторы $a_1,...,a_k$ линейно независимы. Тогда, выбрав из совокупности $\{a_j\}$, *j = 1,..., n - k,* ровно *m - k* векторов, которые с векторами $a_1,...,a_k$ составляют *m* линейно независимых векторов, т.е. подмножество *B*, получим представление *A = [B, N],* для которого справедливо *rank B = m,* $B^{-1} \cdot b \geq 0$, т.к. $x_{i\delta} > 0$, *i = 1,..., k*, и, кроме того,

$$\bar{x} = (x_{i\delta},...,x_{m\delta},0,...,0)^T = \begin{bmatrix} B^{-1} \cdot b \\ 0 \end{bmatrix} \begin{matrix} m \\ n-m \end{matrix}, \qquad (3.6)$$

так что каждой вершине множества *D* соответствует некоторый базис *B*. Обозначим эту вершину через x^1 и положим *k = 1*.

Представление матрицы *A* в виде [*B, N]* порождает аналогичное разбиение для векторов *x* и *c,* а именно, $x = (x_B^T, x_N^T)^T$, $c = (c_B^T, c_N^T)^T$, что, в свою очередь, приводит к представлению уравнения *Ax = b* в виде $B \cdot x_B + N x_N = b$, общее решение которого равно

$$x_B = B^{-1} b - B^{-1} N x_N \, , \qquad (3.7)$$

поэтому значение целевой функции будет равно

$$f(x) = c^T x = c_B^T x_B + c_N^T x_N = c_B^T (B^{-1}b - B^{-1}Nx_N) + c_N^T x_N =$$
$$= c_B^T B^{-1} b + (c_N^T - c_B^T B^{-1} N) x_N . \qquad (3.8)$$

Заметим, что первое слагаемое в (3.8) представляет собой значение целевой функции в первой вершине множества *D: д*ействительно, учитывая представление (3.5), имеем

$$c_B^T B^{-1} b = \begin{bmatrix} c_B \\ c_N \end{bmatrix}^T \begin{bmatrix} B^{-1} \cdot b \\ 0 \end{bmatrix} = c^T \bar{x} = f(\bar{x}), \qquad (3.9)$$

а второе слагаемое в (3.8) показывает величину приращения целевой функции при переходе от текущей вершины множества *D* в какую - либо другую точку. Это приращение либо положительно, либо меньше или равно нулю. Величина

$$(CP)_j = c_j - c_B^T B^{-1} \alpha_j , j \in I_N \qquad (3.10)$$

называется *симплекс - разностью*, связанной с вектором a_j из *N*. Так как $x_N \geq 0$, при выполнении условия $(CP)_j \leq 0, \forall_j \in I_N$ слагаемое $(c_N^T - c_B^T B^{-1} \cdot N) x_N$, которое характеризует приращение значения целевой функции, будет меньше или равно нулю, т.е.

$\delta f(\bar{x}) = \sum_{j \in I_N} (CP)_j \cdot x_j \leq 0$, тогда из (3.8) следует условие оптимальности вершины $\bar{x}$, а именно, $f(x) \leq f(\bar{x})$, $\forall x \in D$.

Таким образом, условие $(CP)_j \leq 0, \forall_j \in I_N$ служит и правилом остановки, и условием оптимальности текущего базисного решения и соответствующей вершины $x^k = \bar{x}$. Если же среди величин $(CP)_j, \forall_j \in I_N$ есть положительные, то поиск оптимального решения продолжается. Выбирая при этом наибольшую положительную симплекс – разность, тем самым мы определим переменную, которая должна войти в новое базисное решение, и соответствующий этой переменной вектор, который должен быть включен в новый базис. С геометрической точки зрения этому выбору соответствует выбор *«удачного»* в текущей вершине x^k направления движения, которое совпадает с одним из боковых ребер, связывающих вершину x^k с другими вершинами множества *D*. Симплексные преобразования, как известно, обеспечивают переход по выбранному ребру в другую

вершину x^k, при наибольшем возрастании значения целевой функции.

Пусть наибольшая симплекс-разность есть

$$(CP)_{jo} = c_{jo} - c_B^T B^{-1} a_{jo} > 0, j_o \in I_N . \qquad (3.11)$$

Для нахождения следующей вершины множества *D,* в которой значение целевой функции больше, чем в точке x^k, воспользуемся рекуррентным соотношением (или формулой) $x^{k+1} = x^k + \alpha_k d^k$. Для этой цели введем в рассмотрение вектор e_{j_0}, состоящий из *(n – m)* координат, j_0-я координата которого равна единице, а остальные координаты равны нулю, и в текущей вершине x^k составим с его помощью *(n x1)* – мерный вектор

$$d^{j_0} = \begin{bmatrix} -B^{-1}a_{j_0} \\ e_{j_0} \end{bmatrix} = \begin{bmatrix} -y_{j_0} \\ e_{j_0} \end{bmatrix} \begin{matrix} m \\ n-m \end{matrix} \qquad (3.12)$$

где обозначено $y_{j_0} = B^{-1}a_{j_0} = (y_{1j_0},...,y_{mj_0})^T$. Первые m координат этого вектора совпадают с координатами вектора $-y_{j_0}$, а остальные $n - m$ координат – с координатами вектора e_{j_0}. По существу уравнение $y_{j_0} = B^{-1}a_{j_0} = (y_{1j_0},...,y_{mj_0})^T$ эквивалентно разложению вектора a_{j_0} через базисные векторы, т. е.

$$a_{j_0} = By_{j_0} = a_1 y_{1j_0} + a_2 y_{2j_0} + ... + a_m y_{mj_0} . \qquad (3.13)$$

Если $y_{ij_0} \le 0, i = 1,...,m$, тогда имеет место условие $d^{j_0} \ge 0$, т.е. все координаты вектора d^{j_0} неотрицательны, кроме того, этот вектор удовлетворяет условию

$$\begin{aligned} Ad^{j_0} &= [B,N]d^{j_0} = \\ BB^{-1}a_{j_0}y_{1j_0} - Ne_{j_0} &= a_{j_0} - a_{j_0} = 0 \end{aligned} . \qquad (3.14)$$

По существу d^{j_0} может служить направлением движения. Покажем, что любая точка, принадлежащая этому направлению d^{j_0}, т.е. точка $x(\alpha) = x^k + \alpha d^{j_0}$, соответствующая произвольной величине шага $\alpha \ge 0$ по этому направлению, принадлежит множеству решений канонической задачи D с описанием $Ax = b$, $x \ge 0$. Действительно, так как $x^k \in D$, то $x(\alpha) \ge 0$, кроме того, на основе (3.14) имеет место

$$Ax(\alpha) = Ax^k + \alpha Ad^{j_0} = Ax^k = b . \qquad (3.15)$$

С геометрической точки зрения условия (3.14) и (3.15) означают, что вектор d^{j_0} является одним из *крайних* (или *экстремальных*) направлений множества *D*, его бесконечным боковым ребром, на котором целевая функция неограниченно возрастает:

$$f(x(\alpha)) = c^T x(\alpha) = c^T (x^k + \alpha d^{j_0}) = c^T x^k + \alpha c^T d^{j_0}, \alpha \geq 0, \quad (3.15)$$

и при увеличении величины шага α значение $f(x(\alpha))$ неограниченно растет, что свидетельствует об отсутствии оптимального решения задачи.

Интересно заметить, что в выражении (3.15) имеет место соотношение

$$c^T d^{j_0} = -c_B^T B^{-1} a_{j_0} + c_N e_{j_0} = -c_B^T B^{-1} \cdot a_{j_0} + c_{j_0} = (CP)_{j_0} \quad (3.16)$$

следовательно, значение целевой функции в точках направления d^{j_0} будет равно

$$f(x(\alpha)) = c^T x^k + \alpha (CP)_{j_0}, \quad (3.17)$$

т.е. величина $\alpha(CP)_{j_0}$ служит приращением для значений целевой функции.

Если среди координат вектора y_{j_0} есть положительные, то направление d^{j_0} приводит к следующей вершине множества *D,* которая более предпочтительна, чем предыдущая вершина. Действительно, представив точку $x(\alpha)$ в развернутой форме

$$x(\alpha) = x^k + \alpha\ d^{j_0} = \begin{bmatrix} x_B \\ 0 \end{bmatrix} + \alpha \begin{bmatrix} -y_{j_0} \\ e_{j_0} \end{bmatrix} = \begin{bmatrix} x_B - \alpha y_{j_0} \\ \alpha e_{j_0} \end{bmatrix}, \quad (3.18)$$

приходим к выводу, что первые *m* координат точки $x(\alpha)$ имеют вид

$$x_i(\alpha) = x_{i\delta} - \alpha y_{ij_0}, i = 1,..., m, \quad (3.19)$$

а остальные *n - m* координат равны значениям

$$x_i(\alpha) = 0, i \neq j_0, x_{j_0}(\alpha) = \alpha \cdot 1, \quad (3.20)$$

так что, выбрав величину шага α на основе правила

$$\alpha = \alpha_0 = min\ \{x_{i\delta} / y_{ij_0}, y_{ij_0} \succ 0\} \quad (3.21)$$

$$i \in I_B$$

для некоторого индекса $i = i_0$, получим: $x_{i_0\delta} = 0$; $x_{j_0} = \alpha_0$, и точка

$$x(\alpha_0) = x^{k+1} = x^k + \alpha_0 d^{j_0} \quad (3.22)$$

оказывается следующей вершиной множества *D*, которая имеет свой (уже новый) базис и соответствующее базисное решение. Значение целевой функции в новой вершине, согласно выражению (3.17), будет равно величине

$$f(x(\alpha_0)) = f(x^{k+1}) = f(x^k) + \alpha_0 (CP)_{j_0}. \quad (3.23)$$

Эти построения позволяют строить следующий сходящийся алгоритм.

Предварительный этап: Представить задачу в канонической форме (3.3), выбрать базис *B,* вычислить x_B, найти вершину x^1 множества D Положить далее $k := k + 1$ и перейти к основному этапу алгоритма.

Основной этап алгоритма:

Шаг 1. Вычислить для всех векторов из *N* симплекс - разности, согласно формуле (3.10); если среди них нет положительных, прекратить поиск, текущее базисное решение x_B и текущая вершина x^k содержат оптимальное решение, а значение целевой функции наибольшее. Если среди симплекс-разностей есть положительные, перейти к следующему шагу.

Шаг 2. Среди положительных симплекс – разностей выбрать наибольшую величину; пусть j_o – ее индекс. Вычислить координаты вектора $y_{j_0} = B^{-1}a_{j_0}$ согласно уравнению (3.13); если все координату вектора y_{j_0} неположительные, прекратить поиск, задача не имеет решения из-за неограниченности целевой функции; если среди координат вектора y_{j_0} есть положительные, перейти к следующему шагу.

Шаг 3. Построить вектор d^{j_0} на основе формулы (3.13), далее с помощью правила (3.21) вычислить

величину шага α_k, построить следующую вершину множества *D* и вычислить значение целевой функции в новой вершине.

Шаг 4. Скорректировать текущий базис путем включения в него вектор a_{j_0} и исключения из него вектор a_{i_o}, который соответствует правилу выбора величины α_k, положить далее $k := k + 1$ и перейти к шагу 1.

Так как количество вершин множества *D* ограничено сверху числом $C_n^m = n!/m!(n - m)!$, за конечное число шагов (или итераций) алгоритм находит оптимальное решение, либо выдает информацию об отсутствии решения из-за неогрниченности значений целевой функции на множестве *D.*

Итак, геометрический вариант симплекс – алгоритма реализует метод возможных направлений $x^{k+1} = x^k + \alpha_k d^k$, не используя информации о значении целевой функции и тем более ее производных. Правило перехода из одной вершины множества *D* в другую его вершину и правило остановки работы алгоритма исходят из топологических свойств пространства решений и знака возможного приращения целевой функции.

5.4. Методы штрафных и барьерных функций

Общая стратегия численных методов решения оптимизационных задач сводится к тому, чтобы исходную задачу условной оптимизации заменить эквивалентной задачей безусловной оптимизации или задачей с более простыми ограничениями, для решения которых могут быть использованы известные методы и алгоритмы.

Для *методов штрафных функций* характерно то, что к значению целевой функции добавляется функция, интерпретируемая как штраф за нарушение каждого из ограничений. Метод генерирует ряд допустимых точек, который сходится к оптимальному решению исходной задачи.

Сущность *методов барьерных функций* заключается в том, что к целевой функции исходной задачи добавляется *барьерный член*, который не позволяет генерировать точки, которые выходят за пределы допустимой области. Таким путем строится последовательность допустимых точек, которые сходятся к искомому решению задачи.

а) Метод штрафных функций. Следует отметить, что выбор штрафных функций является вопросом искусства и зависит от формы ограничений исходной задачи. Рассмотрим следующие частные случаи.

Исходная задача содержит лишь *ограничение в виде равенства:*

$$(D, f): f(x) \to \min, \qquad (4.1)$$
$$h(x) = 0$$
$$x \in X \subseteq E^n$$

эквивалентная безусловная задача строится в виде

$$f(x) + \mu h^2(x) \to \min; \qquad (4.2)$$
$$x \in X \subseteq E^n$$

Исходная задача содержит лишь *ограничение в виде неравенства*

$$(D, f): f(x) \to \min. \qquad (4.3)$$
$$q(x) \le 0$$
$$x \in X \subseteq E^n$$

эквивалентная безусловная задача строится в виде

$$f(x) + \mu \max\{0, q(x)\} \to \min; \qquad (4.4)$$
$$x \in X \subseteq E^n$$

Исходная задача содержит *ограничения обоих видов*

$$(D, f): f(x) \rightarrow \min \qquad (4.5)$$
$$q_i(x) \le 0,\ i = 1, \ldots, m$$
$$h_i(x) = 0,\ i = 1, \ldots, l$$
$$x \in X \subseteq E^n$$

эквивалентная безусловная задача строится в виде

$$f(x) + \mu\, \varphi(x) \rightarrow \min, \qquad (4.4)$$
$$x \in X \subseteq E^n$$

где $\varphi(x)$ – штрафная функция, которая обычно имеет вид

$$\varphi(x) = \sum_{i=1}^{m} \phi(q_i(x)) + \sum_{i=1}^{l} \psi(h_i(x)), \qquad (4.5)$$

где $\phi(z)$ и $\psi(z)$ - непрерывные функции, удовлетворяющие условиям

а) $\phi(z) = 0$, если $z \le 0$, и $\phi(z) > 0$, если $z > 0$;

б) $\psi(z) = 0$, если $z = 0$, и $\psi(z) > 0$, если $z \ne 0$; (4.6)

Типичными считаются следующие формы задания этих вспомогательных функций:

$$\phi(z) = [\max\{0, z\}]^p,$$
$$\psi(z) = |z|^p, \qquad (4.7)$$

где p – целое число. Тогда функция $\varphi(x)$ принимает форму

$$\varphi(x) = \sum_{i=1}^{m} [\max\{o, q_i(x)\}]^p + \sum_{i=1}^{l} |\psi(h_i(x)|^p, \quad (4.8)$$

В предположении, что $\varphi(x)$ – непрерывная функция типа (4.8), составим *вспомогательную – двойственную по Лагранжу задачу*

$$L(\mu) \rightarrow \max, \qquad (4.9)$$
$$\mu \ge 0$$

где

$$L(\mu) = \inf\{f(x) + \mu\varphi(x)\}. \qquad (4.10)$$
$$x \in X$$

Как известно из нелинейного программирования (см. часть 1, главу 3) *основная теорема двойственности* утверждает, что

$$\sup_{\mu \ge 0} L(\mu) = \lim_{\mu \rightarrow \infty} L(\mu) = \min_{x \in D} f(x). \qquad (4.11)$$

Этот фундаментальный результат математического программирования означает, что решение исходной задачи условной оптимизации можно найти путем решения (двойственной по отношению к исходной задаче) задачи (4.9). Решение задачи сводится к реализации следующего алгоритма.

Алгоритм метода штрафных функций.

Шаг 1. Выбрать $\varepsilon > 0$ в качестве критерия остановки работы алгоритма, начальную точку x^1, начальное значение штрафного параметра $\mu_1 > 0$ и число $\beta > 1$. Положить $k = 1$ и перейти к следующему шагу.

Шаг 2. При начальной точке x^k решить задачу

$$f(x) + \mu_k \varphi(x) \to min$$
$$x \in X \subseteq E^n$$

и положить x^{k+1} равным решению этой задачи.

Шаг 3. Если имеет место условие

$$\mu_k \varphi(x^{k+1}) < \varepsilon,$$

то остановиться; в противном случае положить $\mu_{k+1} = \beta\mu_k$, заменить k на $k + 1$ и перейти к шагу 2.

Рекомендуется читателям с помощью этого алгоритма решить задачу

$$f(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 \to min,$$
$$x_1^2 - x_2 = 0$$
$$x \in X \subseteq E^2$$

для вспомогательной функции

$$f(x) + \mu_k \varphi(x) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 + \mu(x_1^2 - x_2)^2.$$

б) Метод барьерных функций. Как отмечалось выше, барьерные функции как бы *препятствуют* выходу текущих решений (приближений) за пределы допустимой области.

Рассмотрим в качестве исходной задачи следующую задачу

$$(D, f): f(x) \to min, \qquad (4.12)$$
$$q_i(x) \le 0,\ i = 1, \ldots, m$$
$$x \in X \subseteq E^n$$

где $q_i(x) \le 0,\ i = 1, \ldots, m,$ - непрерывные на $X \subseteq E^n$ функции. Если задача содержит и ограничения типа $h_i(x) =$

$0,\ i = 1, \ldots, l,$ требуется, чтобы внутренность множества допустимых решений была не пуста, т. е. $int\ D \neq \varnothing$.

Барьерная функция обычно составляется в виде

$$\varphi(x) = \sum_{i=1}^{m} \phi(q_i(x)), \qquad (4.13)$$

где $\phi(z)$ - функция скалярной переменной z, непрерывная на множестве $\{z / z < 0\}$ и удовлетворяющая условиям

$$\phi(z) \geq 0 \text{ при } z < 0,$$
$$\lim \phi(z) = \infty \text{ при } z \to 0. \qquad (4.14)$$

Более привычной формой задания этой функции является

$$\varphi(x) = \sum_{i=1}^{m} (-1/q_i(x)). \qquad (4.15)$$

Алгоритм метода барьерных функций.

Шаг 1. Выбрать $\varepsilon > 0$ в качестве критерия остановки работы алгоритма, начальную точку $x^1 \in X$, для которой выполняется условие $q(x^1) < 0$, начальное значение параметров $\mu_1 > 0$ и $\beta \in (0, 1)$, положить $k = 1$ и перейти к следующему шагу.

Шаг 2. При начальной точке x^k решить задачу

$$f(x) + \mu_k \varphi(x) \to min,$$
$$x \in X \subseteq E^n$$

при условии $q(x) < 0$ и положить x^{k+1} равным решению этой задачи.

Шаг 3. Если имеет место условие

$$\mu_k \varphi(x^{k+1}) < \varepsilon,$$

то остановиться; в противном случае положить $\mu_{k+1} = \beta\mu_k$, заменить k на $k + 1$ и перейти к шагу 2.

5.5. Методы возможных направлений.

В этом разделе мы рассмотрим ряд подходов к численному решению задачи условной оптимизации, которые основаны на классических *методах возможных направлений* и их модификациях (так называемые алгоритмы Зойтендейка). Эти методы легли в основу современных поисковых алгоритмов и соответствующих машинных программ, которые составляют основу математического обеспечения многих пакетов программ по оптимизации и успешно применяются в исследовательской практике.

Для общей задачи нелинейного программирования

$$(D, f): f(x) \to \min, \qquad (5.1)$$
$$q_i(x) \le 0,\ i = 1, \dots, m$$
$$h_i(x) = 0,\ i = 1, \dots, l$$
$$x \in X \subseteq E^n$$

возможные направления определяются следующим образом. Пусть x^k – некоторая точка из D, $k = 0, 1, 2, \dots$Направление d называется *возможным* в точке x^k, если существует число $\delta > 0$, такое, что для всех $\alpha \in (0, \delta)$ выполняется условие $x^k + \alpha d \in D$, или формально:

$$\exists\, \delta > 0:\ x^k + \alpha d \in D,\ \forall\, \alpha \in (0, \delta). \qquad (5.2)$$

Если в точках допустимого направления значение целевой функции меньше, чем в точке x^k, то можно осуществить *одномерную минимизацию* функции по этому направлению и найти точку минимума функции $f(x)$ на этом направлении. Это означает, что необходимо найти наибольшее значение величины $\alpha = \alpha_{max}$, для которого $f(x^k + \alpha d) < f(x^k)$ и при этом $x^k + \alpha_{max} d \in D$.

а) Алгоритм Зойтендейка для задачи с линейными ограничениями.

Особый интерес представляет случай, когда целевая функция задачи нелинейная, а ограничения линейны, т. е. задача

$$D, f): f(x) \to \min, \qquad (5.3)$$
$$Ax \le b,\ Ex = e$$
$$x \in X \subseteq E^n,\ x \ge 0$$

где A – (mxn) – матрица ограничений в виде неравенств, E – (lxn) - матрица ограничений в виде равенств, b – $(mx1)$ – вектор, e – $(lx1)$ – вектор.

В линейной алгебре доказывается следующее утверждение, называемое *леммой [3]*. Пусть $x \ge 0$ – некоторая допустимая точка задачи, и выполняются условия $A_1 x = b_1,\ A_2 x < b_2$, где $A^T = (A_1^T, A_2^T)$, $b^T = (b_1^T, b_2^T)$. Ненулевое направление d является возможным направлением в точке x тогда и только тогда, когда имеет место условие

$$A_1 d \le 0,\ E d = 0. \qquad (5.4)$$

Если, кроме этого, выполняется строгое неравенство

$$\nabla f(x)^T d < 0, \qquad (5.5)$$

то d является возможным направлением спуска. На этом основан следующий численный алгоритм *Зойтендейка.*

Шаг 1. Найти некоторую начальную точку x^k, для которой $Ax^1 \le b$, $Ex^1 = e$, положить $k = 1$ и перейти к следующему шагу.

Шаг 2. Предположим, что в точке x^k имеют место

$A^T = (A_1^T, A_2^T)$, $b^T = (b_1^T, b_2^T)$, $A_1 x^k = b_1$, $A_2 x^k < b_2$.

Шаг 3. Решить задачу

$$\nabla f(x^k)^T d \rightarrow \min. \qquad (5.6)$$
$$A_1 d < 0, \; E d = 0.$$
$$-1 \le d_j \le +1, \; j = 1, \ldots, n$$

и решение обозначить через d^k. Если имеет место условие $\nabla f(x^k)^T d^k = 0$, остановиться; точка x^k удовлетворяет условиям теоремы Куна – Таккера, т. е. является оптимальным решением задачи. В противном случае перейти к следующему шагу.

Шаг 4. Положить α_k равным оптимальному решению следующей одномерной задачи оптимизации

$$f(x^k + \alpha d^k) \rightarrow \min, \qquad (5.7)$$
$$0 \le \alpha \le \alpha_{max}$$

положить далее $x^{k+1} = x^k + \alpha_k d^k$, определить в x^{k+1} новое значение активных ограничений, переопределить A_1 и A_2, заменить k на $k+1$ и перейти к шагу 2.

Для линейных ограничений задачи величина α_{max} определяется из условия

$$\alpha_{max} = \min\{\tilde{b}_i / \tilde{d}_i, \; \tilde{d}_i > 0\}, \text{ если } \tilde{d} > 0,$$
$$\alpha_{max} = \infty, \text{ если } \tilde{d} \le 0, \qquad (5.8)$$

где $\tilde{b} = b_2 - A_2 x^k$, $\tilde{d} = A_2 d^k$.

б) Алгоритм Зойтендейка для задачи с нелинейными ограничениями – неравенствами. Займемся теперь решением задачи

$$(D, f): f(x) \rightarrow \min. \qquad (5.9)$$
$$q_i(x) \le 0, \; i = 1, \ldots, m$$
$$x \in X \subseteq E^n$$

Алгоритм ее решения содержит следующую последовательность действий.

Шаг 1. Выбрать начальную точку x^1, удовлетворяющую ограничениям $q_i(x^1) \le 0$, $i = 1, \ldots, m$, положить $k = 1$ и перейти к следующему шагу.

Шаг 2. Положить $I = \{i: q_i(x^1) = 0\}$ (так называемое *множество индексов активных ограничений)* и решить задачу

$$z \rightarrow \min. \qquad (5.10)$$
$$\nabla f(x^k)^T d - z \le 0$$
$$\nabla q_i(x^k)^T d - z \le 0,\ i \in I$$
$$-1 \le d_j \le +1,\ j = 1, \dots, n$$

Пусть двойка (z_k, d^r) – оптимальное решение этой задачи. Если $z_k = 0$, то остановиться; x^k является оптимальным решением задачи. Если $z_k < 0$, перейти к следующему шагу.

Шаг 3. Выбрать в качестве α_k оптимальное решение задачи

$$f(x^k + \alpha d^k) \rightarrow \min,$$
$$0 \le \alpha \le \alpha_{max}$$

где $\alpha_{max} = \sup\{\alpha : q_i(x^k + \alpha d^k) \le 0,\ i = 1, \dots, m\}$, положить $x^{k+1} = x^k + \alpha_k d^k$, заменить k на $k + 1$ и перейти к шагу 2.

в) Алгоритм Зойтендейка для общей задачи нелинейного программирования

$$(D, f): f(x) \rightarrow \min, \qquad (5.11)$$
$$q_i(x) \le 0,\ i = 1, \dots, m$$
$$h_i(x) = 0,\ i = 1, \dots, l$$
$$x \in X \subseteq E^n$$

Пусть x^k, $k = 0, 1, 2, \dots$, - уже найденные приближения к искомому решению x^*, а $I = \{i: q_i(x^1) = 0\}$ – множество индексов активных ограничений – неравенств в точке x^k. Выберем направление движения d^k как решение задачи

$$\nabla f(x^k)^T d \rightarrow \min. \qquad (5.12)$$
$$\nabla q_i(x^k)^T d \le 0,\ i \in I$$
$$\nabla h_i(x^k)^T d = 0,\ i = 1, \dots, l$$
$$-1 \le d_j \le +1,\ j = 1, \dots, n$$

Решение d^k является касательным к ограничениям – равенствам и к некоторым ограничениям – неравенствам. Линейный поиск вдоль этого направления

$$f(x^k + \alpha d^k) \rightarrow \min,$$
$$0 \le \alpha \le \alpha_{max}$$

позволяет вычислить величину α_k и следующую точку $x^{k+1} = x^k + \alpha_k d^k$, после чего эти действия повторяются.

Отметим, что в общем случае метод Зойтендейка не гарантирует сходимость этих алгоритмов для нелинейных ограничений. Проблема заключается в том, что *алгоритмическое отображение* $x^{k+1} = W(x^k)$ не замкнуто. Чтобы гарантировать сходимость, Топкис и Вейнот предложили следующую модификацию алгоритма при ограничениях – неравенствах [3]: выбрать направление как решение задачи

$$z \to \min. \qquad (5.13)$$
$$\nabla f(x^k)^T d - z \le 0$$
$$\nabla q_i(x^k)^T d - z \le - q_i(x^k),$$
$$i = 1, \dots, m$$
$$\le d_j \le + 1, j = 1, \dots, n$$

Работу алгоритма можно представить в виде следующей последовательности действий.

Шаг 1. Выбрать начальное приближение x^1, для которого $q_i(x^1) \le 0, i = 1, \dots, m,$ положить $k = 1$ и перейти к следующему шагу.

Шаг 2. Положить (z_k, d^k) решение линейной задачи (5.13). Если $z_k = 0$, то остановиться; x^k является оптимальным решением задачи. Если же $z_k < 0$, перейти к следующему шагу.

Шаг 3. Положить α_k решением задачи

$$f(x^k + \alpha d^k) \to \min,$$
$$0 \le \alpha \le \alpha_{max}$$

где $\alpha_{max} = \sup\{\alpha: q_i(x^k + \alpha d^k) \le 0, i = 1, \dots, m\}$, положить $x^{k+1} = x^k + \alpha_k d^k$, заменить k на $k + 1$ и перейти к *Шаг 2.* Специальные теоремы нелинейного программирования гарантируют сходимость этой важной модификации общей задачи [Б Ш].

г) Алгоритм Франка – Вольфа.

Важным частным случаем общей задачи нелинейного программирования является задача, в которой область допустимых решений D является выпуклым и компактным (замкнутым и ограниченным по расстоянию) множеством. Для такого случая весьма

успешным является *алгоритм Франка – Вольфа*, основанный на следующих действиях.

Шаг 1. Выбрать начальное приближение $x^1 \in D$, положить $k = 1$ и перейти к следующему шагу.

Шаг 2. Решить задачу

$$\nabla f(x^k)^T z \rightarrow \min \qquad (5.14)$$
$$z \in D$$

и обозначить решение через z^k.

Шаг 3. Вычислить направлений

$$d^k = z^k - x^k. \qquad (5.15)$$

Шаг 4. Определить наилучшую величину шага по правилу

$$f(x^k + \alpha d^k) \rightarrow \min$$
$$0 \le \alpha \le \alpha_{max}$$

положить $x^{k+1} = x^k + \alpha_k d^k$. Если имеет место условие

$$| f(x^{k+1}) - f(x^k) | \le \varepsilon, \qquad (5.16)$$

то остановиться, в противном случае заменить k на $k + 1$ и перейти к *Шаг 2.*

Покажем, что порождаемая в результате работы этого алгоритма последовательность $\{x^k\}$ целиком принадлежит D. Действительно, по определению направления $d^k = z^k - x^k$ имеем

$$x^{k+1} = x^k + \alpha_k d^k = x^k + \alpha_k (z^k - x^k) =$$
$$= \alpha_k z^k + (1 - \alpha_k) x^k \qquad (5.17)$$

и, поскольку точки z^k и x^k принадлежат D, то их линейная комбинация также принадлежит этому множеству из-за выпуклость, т. е. $\alpha_k z^k + (1 - \alpha_k) x^k \in D$.

Многие прикладные задачи проектирования, планирования и управления действительно удовлетворяют требованиям этой модификации, что и служит причиной большого распространения алгоритма Франка – Вольфа.

5.6. Метод проекции Розена

В приведенных выше численных методах неоднократно отмечалось, что направление антиградиента $\nabla f(x^k)$ одновременно служит для целевой функции

направлением наискорейшего спуска (направлением убывания ее значений). Однако при наличии ограничений движение вдоль этого направления может привести к недопустимым точкам. *«Вернуть»* эти точки в допустимое множество можно, *проектируя* направление антиградиента таким образом, чтобы сохранилась допустимость точек траектории, и в то же время значение целевой функции уменьшалось. Эта идея основана на применении так называемых *матриц проектирования*.

Определение. Матрица P размерности (nxn) называется *матрицей проектирования*, если выполняются условия

$$P = P^T, \quad PP = P. \qquad (6.1)$$

Матрица проектирования обладает следующим полезными свойствами:

а) матрица проектирования положительно полуопределенная, т. е. $z^T P z \geq 0$, для всех z, $z \neq 0$. Действительно, согласно (6.1) имеет место представление

$$x^T Px = x^T PPx = x^T P^T P x = (xP)^T Px = ||Px||^2 \geq ;$$

б) для того чтобы (nxn) - матрица P была матрицей проектирования, необходимо и достаточно, чтобы матрица $I - P$ также была матрицей проектирования, где I – единичная матрица размерности (nxn) (ее диагональные элементы равны единице, а все остальные элементы равны нулю). Действительно, по определению этих матриц имеем

$$(I - P)^T = I^T - P^T = I - P;$$

$$(I - P)(I - P) = II - IP - PI - PP = I - 2P + P = I - P;$$

в) если P – матрица проектирования и $Q = I - P$, то подпространства $L = \{Px,\ x \in E^n\}$ и $L^* = \{Qx,\ x \in E^n\}$ являются *ортогональными*, и имеет место представление

$$x = p + q,\ p \in L,\ q \in L^*\ \forall x \in E^n. \qquad (6.2)$$

Действительно, L и L^*, очевидно, являются *линейными подпространствами* и

$$P^T Q = P^T (I - P) = P(I - P) = P - PP = 0,$$

кроме того, если $x \in E^n$, то

$$x = Ix = (P + Q)x = Px + Qx = p + q,\ pL\ qL^*.$$

Покажем теперь, что это представление единственное. Пусть $x = p' + q', p' \in L, q' \in L^*$. Тогда x = p + $q = p' + q'$, откуда следует, что p - $p' = q + q'$, следовательно, p - $p' \in L$, $q - q' \in L^*$. Так как единственной точкой пересечения L и L^* является начало координат, получим $p - p' = q - q' = 0, p = p', q = q'$. Так что представление $x = p + q$, $p \in L$, $q \in L^* \ \forall x \in E^n$ действительно является единственным.

а) Алгоритм Розена для задачи с линейными ограничениями.

Исходная задача имеет вид (см. (5.3))

$$(D, f): f(x) \to min, \qquad (6.3)$$
$$Ax \le b,\ Ex = e$$
$$x \in X \subseteq E^n,\ x \ge 0$$

где, как и раньше, A – (mxn) – матрица ограничений в виде неравенств, E – (lxn) - матрица ограничений в виде

равенств, b – $(mx1)$ – вектор, e – $(lx1)$ – вектор, функция $f(x)$ дифференцируема во всех точках множества D.

В предположении, что приближения x^k, $k = 0, 1, 2, \ldots,$ уже сгенерированы, выберем в качестве приемлемого направления $d^k = - P\nabla f(x^k)$, где P – матрица проектирования, полагая при этом, что $P\nabla f(x^k) \neq 0$. Докажем, что если в точке x^k выполняются условия $A_1 x^k = b_1$, $A_2 x^k < b_2$, где $A^T = (A_1^T, A_2^T)$, $b^T = (b_1^T, b_2^T)$, то направление d^k является направлением спуска для функции $f(x)$ в точке x^k, кроме того, если матрица $M^T = (A_1^T, E^T)$ имеет полный ранг и $P = I - M^T(MM^T)^{-1}M$, то $d^k = - P\nabla f(x^k)$ – возможное направление спуска.

Действительно, т. к. P – матрица проектирования, т. е. $P = P^T$ и $PP = P$, получим

$$\nabla f(x^k)^T d^k = - \nabla f(x^k)^T P \nabla f(x^k) =$$

$$= - \nabla f(x^k)^T P^T P \nabla f(x^k) = -\|P\nabla f(x^k)\|^2 < 0, \qquad (6.4)$$

так что функция $f(x)$ действительно убывает вдоль направления $d^k = - P\nabla f(x^k)$.

На рис. 5.3 и 5.4 иллюстрирован механизм корректировки направлений путем проектирования. Первый рисунок соответствует линейным ограничениям; матрица P проектирует антиградиент $-\nabla f(x^k)$ на боковое ребро множества решений D. Второй рисунок соответствует нелинейным ограничениям; скорректированная с помощью матрицы P точка x^{k+1} уже принадлежит множеству D.

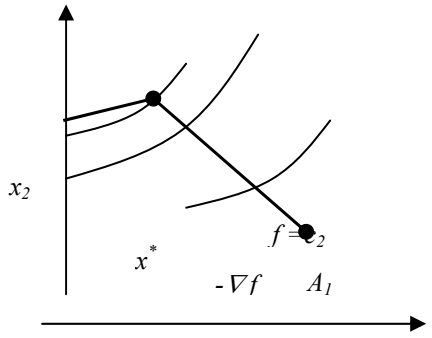

Рис. 5.3. Техника проектирования антиградиента в задаче с линейными ограничениями.

Рис. 5.4 Техника проектирования антиградиента в задаче с нелинейными ограничениями.

Выбор $M^T = (A_1^T, E^T)$ и $P = I - M^T(MM^T)^{-1}M$ означает, что $MP = 0$, т. е.

$$A_1P = 0, \quad EP = 0. \qquad (6.5)$$

следовательно, P проектирует каждую строку матриц A_1 и E в нулевой вектор. Но строками A_1 и E являются градиенты функций активных ограничений $A_1x^k = b_1$ и $Ex^k = e$, следовательно, матрица P проектирует градиенты активных ограничений в нулевой вектор. Эти соображения позволяют строить следующий алгоритм поиска.

Шаг 1. Выбрать начальное приближение x^1, удовлетворяющее ограничениям, т. е. $Ax^1 \le b$, $Ex^1 = e$, представить A и b в виде $A^T = (A_1^T, A_2^T)$, $b^T = (b_1^T, b_2^T)$, где $A_1x^1 = b_1$, $A_2 x^1 < b_2$, положить $k = 1$ и перейти к следующему шагу.

Шаг 2. Положить $M^T = (A_1^T, E^T)$. Если M пустая матрица, т. е. не содержит ни одной строки, то положить $P = I$; в противном случае положить $P = I - M^T(MM^T)^{-1}M$, $d^k = - P\nabla f(x^k)$. Если $d^k \ne 0$, перейти к шагу 3, а в противном случае, т. е. $d^k = 0$, а M пуста, остановиться, если же M непуста, положить

$$\omega = - (MM^T)^{-1}M\nabla f(x^k). \qquad (6.6)$$

Пусть $\omega^T = (u^T, v^T)$. Если $u \ge 0$, то остановиться; точка x^k удовлетворяет условиям теоремы Куна – Таккера, т. е. она является оптимальным решением задачи. Если координаты вектора u не удовлетворяют условию $u \ge 0$, то выбрать отрицательную координату u_j этого вектора, переопределить матрицу A_1, вычеркивая строку, соответствующую координате u_j, и повторить действие шага 2.

Шаг 3. Выбрать величину α_k из условия

$$f(x^k + \alpha d^k) \to \min,$$
$$0 \le \alpha \le \alpha_{max}$$

положить $x^{k+1} = x^k + \alpha_k d^k$, $k := k + 1$ и перейти к шагу 2.

Как и выше (см. формулу (5.8)), величина α_{max} отвечает условиям

$$\alpha_{max} = \min\{ b_i / d_i,\ d_i > 0 \}, \text{ если } d \not\le 0,$$

$$\alpha_{max} = \infty, \text{ если } d \le 0.$$

б) Алгоритм Розена для задачи с нелинейными ограничениями.

Для общей задачи математического программирования (см. (5.1))

$$(D, f): f(x) \to \min, \qquad (6.7)$$
$$q_i(x) \le 0,\ i = 1, \dots, m$$
$$h_i(x) = 0,\ i = 1, \dots, l$$
$$x \in X \subseteq E^n$$

с начальной точкой x^k, $k = 0, 1, 2, \dots$, и множеством активных ограничений $I = \{i / q_i(x^k) = 0\}$ в этой точке в

качестве M выбирается матрица, строками которой являются градиенты

$\nabla q_i(x^k),\ i \in I,$ и $\nabla h_j(x^k), j = 1, \ldots, l,$ т. е. полагается

$$M^T = (\nabla q_i(x^k)^T,\ i \in I, \nabla h_j(x^k)^T, j = 1, \ldots, l), \quad (6.8)$$

а в качестве проекционной матрицы, как и выше, выбирается $P = I - M^T(MM^T)^{-1}M$. Тогда $d^k = - P\nabla f(x^k)$ является направлением спуска.

Если $d^k \neq 0,$ то величина шага α_k определяется из условия минимума $f(x^k + \alpha d^k)$ при $0 \leq \alpha \leq \alpha_{max},$ а новая точка строится по формуле $x^{k+1} = x^k + \alpha_k d^k$.

Если же $d^k = 0,$ вычисляется вектор

$$\omega = - (MM^T)^{-1}M\nabla f(x^k).$$

Если в представлении $\omega^T = (u^T, v^T)$ вектор u неотрицателен, т. е. $u \geq 0,$ работа алгоритма прекращается; точка x^k удовлетворяет условиям теоремы Куна – Таккера, т. е. является оптимальным решением задачи. Если координаты вектора u не удовлетворяют условию $u \geq 0,$ то выбирается его отрицательная координата $u_j,$ переопределяется матрица $M,$ путем

вычеркивания строки, соответствующей координате $u_j,$ и повторяется процедура.

Таким образом, в случае нелинейных ограничений в алгоритме Розена изменяется лишь матрица $M,$ т. е. вместо матрицы $M^T = (A_1^T, E^T),$ теперь строится $M^T = (\nabla q_i(x^k)^T, \nabla h_j(x^k)^T),$ где градиенты соответствуют активным ограничениям, во всем остальном алгоритм повторяет действия алгоритма при линейных ограничениях.

Рекомендуется при программировании решения задачи на ЭВМ воспользоваться нижеследующей последовательностью вычислительных действий.

Шаг 1. Ввести исходные данные, выбрать начальное приближение $x^1 \in D,$ положить $k = 1$ и перейти к следующему шагу.

Шаг 2. В текущей точке x^k построить множество M^T из активных ограничений; пусть L – количество строк этой матрицы. Если $L = 0,$ перейти к шагу 3, в противном случае – к шагу 4.

Шаг 3. Вычислить антиградиент целевой функции $S^k = - \nabla f(x^k),$ построить нормированный вектор направления $S^k := S^k/\|S^k\|$, вычислить длину шага $\alpha_k = \arg\min f(x^k + \alpha S^k),\ 0 \leq \alpha \leq \alpha_{max},$ вычислить новое

приближение $x^{k+1} = x^k + \alpha_k S^k$, положить $k: = k + 1$ и повторить действие шага 2.

Шаг 4. Построить матрицу проектирования $P = I - M^T(MM^T)^{-1}M$, вычислить градиент $\nabla f(x^k)$ и построить движения направление $d^k = - P\nabla f(x^k)$. Если $d^k = 0$, перейти к шагу 6. В противном случае – к шагу 5.

Шаг 5. Построить нормированный вектор $S^k = d^k/\|d^k\|$, вычислить оптимальную длину шага $\alpha_k = arg\ min\ f(x^k + \alpha S^k)\ 0, \le \alpha \le \alpha_{max}$, новое приближение $x^{k+1} = x^k + \alpha_k S^k$. Если $\alpha_k < \alpha_{max}$, то в новой точке x^{k+1} число ограничений в виде равенств (активные ограничения) не меняется, следовательно, матрица M^T не меняется, если ограничения линейны, а если нелинейны, то дифференциалы $\nabla q_i(x^k)$ и $\nabla h_j(x^k)$ для активных ограничений, *с* помощью которых образуется эта матрица, нужно вычислить в точке x^{k+1}. Если $\alpha = \alpha_{max}$, то в дополнение к L ограничениям, которые учитывались в точке x^k в виде равенства (и сохраняющиеся при переходе от x^k к x^{k+1}) должны приниматься во внимание другие ограничения в виде равенства. Положить $k: = k + 1$ и перейти к шагу 2.

Шаг 6. Вычислить вектор $\omega = - (MM^T)^{-1}M\nabla f(x^k)$. Если все координаты этого вектора неотрицательны, остановиться; найденная точка x^k является оптимальным решением задачи. В противном случае перейти к шагу 7.

Шаг 7. Выбрать среди координат вектора ω координату с наименьшим отрицательным значением, скорректировать матрицу M^T путем исключения из нее строки с индексом, соответствующей индексу выбранной (наименьшей отрицательной) координаты, и перейти к шагу 2.

Условия сходимости алгоритмов и замкнутости сложных алгоритмических отображений сводятся к следующему. В разделе 5.1 уже шла речь о сходящихся алгоритмических отображениях и построении последовательностей точек $\{x^k\}$ на их основе. В заключение данной главы мы приведет результаты известных теорем, которые гарантируют сходимость численных алгоритмов, построенных на основе замкнутых алгоритмических отображений, а также доказательство замкнутости сложных алгоритмических отображений, лежащих в основе методов возможных направлений,

следовательно, и построенных с их помощью последовательностей.

Теорема о сходимости алгоритма.

Пусть D – непустое замкнутое множество в E^n, $X^* \subset D$ – непустое подмножество решений задачи условной оптимизации, $W: D \to D$ – некоторое *точечно–множественное* отображение, а последовательность $\{x^k\}$ построена на основе отображения $x^{k+1} \in W(x^k)$. Тогда справедливо утверждение:

а) если $x^k \in X^*$, то процесс останавливается. В противном случае $x^{k+1} \in W(x^k)$;

б) последовательность $\{x^k\}$ содержится в компактном подмножестве множества D;

в) $f(x^{k+1}) < f(x^k)$, если $x^k \notin X^*$ и $x^{k+1} \in W(x^k)$;

г) отображение $W(x^k)$ замкнуто на дополнении к X^*,

то алгоритм либо останавливается через конечное число шагов в точке из X^*, либо он порождает бесконечную последовательность $\{x^k\}$, такую, что любая сходящаяся ее

подпоследовательность имеет предел в X^* и $f(x^k) \to f(x^*)$ для некоторого $x^* \in X^*$.

Доказательство. Если $x^k \notin X^*$, то генерируется бесконечная последовательность $\{x^k\}$. Пусть $\{x^k\}_K$ - некоторая сходящаяся подпоследовательность $\{x^k\}$, имеющая своим пределом $x^* \in D$. Тогда из-за непрерывности функции $f(x)$ получим $\lim f(x^k) \to f(x^*)$ для $k \in K$. Это означает, что для любого заданного $\varepsilon > 0$ найдется номер $K_0 \in K$, такой, что

$$f(x^k) - f(x^*) < \varepsilon \qquad (6.9)$$

при $k \geq K_0$ и $k \in K$. В частности, при $k = K_0$ также имеем $f(x^k) - f(x^*) < \varepsilon$. Пусть теперь $k > K_0$. Так как функция $f(x)$ убывает, то $f(x^k) < f(x^{K_0})$, или же, учитывая (6.9),

$$f(x^k) - f(x^*) = f(x^k) - f(x^{K_0}) + f(x^{K_0}) -$$

$$f(x^*) < 0 + \varepsilon = \varepsilon. \qquad (6.10)$$

В последнем выражении величина $\varepsilon > 0$ выбрана произвольно, следовательно,

$$\lim_{k \to \infty} f(x^k) = f(x^*). \qquad (6.11)$$

Остается показать, что $x^* \in X^*$. Допустим, что $x^* \notin X^*$. Рассмотрим последовательность $\{x^{k+1}\}_K$, которая содержится в компактном подмножестве множества D, следовательно, из нее можно выделить другую подпоследовательность $\{x^{k+1}\}_{\bar{K}}$, которая сходится к некоторой точке $\bar{x} \in D$. На основе (6.11) можно заключить, что $f(\bar{x}) = f(x^*)$. Так как отображение W замкнуто и, кроме того, $x^k \to x^*$, $x^{k+1} \in W(x^k)$, $x^{k+1} \to x^*$ для всех $k \in K$, то справедливо также $\bar{x} \in W(x^*)$. Следовательно, $f(\bar{x}) < f(x^*)$, что противоречит равенству $f(\bar{x}) = f(x^*)$. Таким образом, $x^* \in X^*$. С учетом (6.11), это доказательство приводит к условию $f(x^k) \to f(x^*)$ для $x^* \in X^*$.

Во всех алгоритмах, основанных на идее построения *возможных направлений* и последующего

одномерного спуска (или *подъема* в случае задачи на максимум) вдоль этих направлений, по существу мы используем *сложное алгоритмическое отображение,* представляющее собой *композицию двух алгоритмических отображений*, т. е. $W = W_2 \circ W_1$, где символом «$\circ$» обозначено действие *композиции отображений*.

Отображение $W_1 : E^n \times E^n \to E^n$ определяется соотношением $(x, d) \in W_1$, где x – текущее состояние, d – выбираемое направление движения, а отображение W_2: $E^n \times E^n \to E^1$ определяется соотношением $\alpha \in W_2(x, d)$, где α - величина шага, которая выбирается из условия минимума выражения $f(x + \alpha d)$, $0 \le \alpha \le \alpha_{max}$, $x + \alpha d \in D$. Результирующее отображение W при этом определяется соотношением $(d, \alpha) \in W$. Все три отображения являются *точечно-точечными отображениями* в определенном выше смысле, т. е. $x^{k+1} = W(x^k)$.

В общем случае сложное отображение W может оказаться незамкнутым, что создает проблему, связанную с сходимостью алгоритмов. Вопрос *замкнутости сложного отображения* решается на основе следующих соображений общей теории[Б,Ш].

Теорема о замкнутости алгоритмического отображения.

Пусть X, Y, Z непустые замкнутые множества в E^n, E^p и E^g соответственно. Рассмотрим точечно-множественные отображения $B: X \to Y$, $C: Y \to Z$ и $A = C \circ B: X \to Z$. Предположим далее, что B замкнуто в точке $\bar{x} \in X$, C замкнуто на множестве $B(\bar{x})$, и если последовательности $\{x^k\}$ и $\{y^k\}$ сходятся к точкам $\bar{x}$ и $\bar{y}$ соответственно, то существует сходящаяся подпоследовательность $\{y^k\}$. Тогда отображение A замкнуто в $\bar{x}$.

Доказательство. Пусть $x^k \to \bar{x}$, $z^k \in A(x^k)$ и имеет место $z^k \to \bar{z}$. Нужно показать, что справедливо включение $\bar{z} \in A(\bar{x})$, т. е. что A замкнуто в $\bar{x}$. По определению A для каждого индекса k существует $y^k \in B(x^k)$, такое, что $z^k \in C(y^k)$. По предположению, существует подпоследовательность $\{y^k\}$, сходящуюся к точке $\bar{y}$. Но B замкнуто в $\bar{x}$, так что $\bar{y} \in B(\bar{x})$, кроме того, из замкнутости C на $B(\bar{x})$ следует также и замкнутость C в $\bar{y} \in B(\bar{x})$, а значит, $\bar{z} \in C(\bar{y}) \subset C \circ B(\bar{x}) = A(\bar{x})$. Итак условие существования сходящейся

подпоследовательности является весьма существенным для условия замкнутости алгоритмического отображения A в точке $\bar{x}$.

В терминах алгоритмов математического программирования принятые в теореме обозначения означают:

$$W_1: E^n \times E^n \to E^n, (x, d) \in W_1(x);$$

$$W_2: E^n \times E^n \to E^1, y \in W_2(x, d);$$

$$W = W_2 \circ W_1, (d, \alpha) \in W.$$

ПРИЛОЖЕНИЯ

1. Задача выпуклого программирования

Теорема. Пусть множество $D \subset E^n$, $D \neq \varnothing$, выпукло, функция $f(x)$ определена на D и точка x^* является локальным оптимальным решением задачи

$$f(x) \to min. \qquad (П1.1)$$
$$x \in D$$

а) Если $f(x)$ – выпуклая функция, то x^* - глобальное оптимальное решение;

б) Если $f(x)$ – строго выпуклая функция, то x^* - единственное глобальное решение.

Доказательство. Обозначим через $N_\varepsilon(x)$ – некоторую ε - окрестность точки x^*. Пусть точка x^* не является глобальным решением задачи (П1.1), т. е. неверно, что имеет место условие

$$f(x^*) \le f(x), \ \forall x \in D \cap N_\varepsilon(x). \qquad (П1.2)$$

Тогда существует точка $x^{**} \in D$, для которой справедливо условие $f(x^{**}) < f(x^*)$. Так как функция $f(x)$ по определению выпукла, то для всех $\alpha \in (0, 1)$ имеет место условие

$$f(\alpha x^{**} + (1 - \alpha)x^*) \le \alpha f(x^{**}) + (1 - \alpha)f(x^*) <$$
$$\alpha f(x^*) + (1 - \alpha)f(x^*) = f(x^*). \qquad (П1.3)$$

Но при достаточно малом значении α точка $\alpha x^{**} + (1 - \alpha)x^*$ будет принадлежать множеству $D \cap N_\varepsilon(x)$, т. е. $\alpha x^{**} + (1 - \alpha)x^* \in D \cap N_\varepsilon(x)$, что входит в противоречие с определением точки x^* как локального оптимального решения. Следовательно, x^* является глобальным оптимумом функции $f(x)$ на D, и пункт а) доказан. Докажем теперь справедливость пункта б). Пусть x^* - не единственное глобальное решение, и существует другая точка $x^{**} \in D$, $x^{**} \neq x^*$, такая, что $f(x^{**}) = f(x^*)$. Так как по определению $f(x)$ – строго выпуклая функция, то имеет место условие

$$f(\alpha x^{**} + (1 - \alpha)x^*) < \alpha f(x^{**}) + (1 - \alpha)f(x^*) = f(x^*), \qquad (П1.4)$$

но это противоречит условию глобального оптимума точки x^*. Остается утверждать, что эта точка единственная, другими словами, условие б) верно.

Итак, задача выпуклого программирования имеет только глобальные оптимальные решения, и если целевая функция строго выпукла, то глобальное решение единственно. Аналогичное утверждение справедливо и для

задачи вогнутого программирования, когда область допустимых решений выпукло, а целевая функция вогнута, тогда функция с отрицательным знаком будет выпукла, и мы вновь получим задачу выпуклого программирования.

Задача линейного программирования, как отмечалось в первой части настоящего пособия, относится к классу задач выпуклого программирования, поэтому она может иметь только глобальные решения, как это непосредственно следует из центральной теоремы математического программирования – теоремы Вейерштрасса.

2. Условие для направления спуска

Теорема. Пусть функция $f(x)$ определена на множестве $D \subset E^n$, $D \neq \varnothing$, и $x^k \in D$, существует направление d, такое, что в точке x^k выполняется условие

$$\nabla f(x^k)^T d < 0, \qquad (П2.1)$$

где $\nabla f\ x^k)$ – градиент функции $f(x)$ в точке x^k. Тогда существует величина $\delta > 0$, такая, что

$$x^k + \alpha d \in D, \quad \forall \alpha \in (0, \delta). \qquad (П2.2)$$

Направление d называется направлением спуска (убывания) функции $f(x)$ в точке $x^k \in D$. Для точек этого направления $x^k + \alpha d)$ справедливо разложение

$$f(x^k + \alpha d) = f(x^k) + \alpha \nabla f(x^k)^T d + \alpha^2 // d // \theta(x^k, \alpha d), \qquad (П3.3)$$

где $lim\ \theta(x^k, \alpha d) = 0, \ \alpha \to 0$.

3. Геометрическое условие оптимальности

Теорема. Рассмотрим задачу математического программирования

$$f(x) \to min. \qquad (П3.1)$$
$$q_i(x) \leq 0, i = 1, \ldots, m$$
$$x \in X \subseteq E^n$$

Пусть $x^k \in D$ – некоторая допустимая точка задачи, $I = \{i / q_i(x^k) = 0\}$ – множество активных ограничений в точке x^k, функции $f(x)$ и $q_i(x)$, $i \in I$, дифференцируемы в точке x^k, а функции $q_i(x)$,), $i \notin I$, непрерывны в точке x^k, $F_0 = \{d / \nabla f(x^k)^T d < 0\}$, $G_0 = \{d / \nabla q_i(x^k)^T d < 0\}$. Если x^k – точка локального оптимума задачи (П3.1), то имеет место условие

$$F_0 \cap G_0 = \varnothing. \qquad (П3.2)$$

Литература

1. Беллман Р. Динамическое программирование: Пер. с англ. –М.: ИЛ, 1960.
2. Беллман Р., Калаба Р. Динамическое программирование и современная теория управления: Пер. с англ. –М.: Наука, 1969. –120 с.
3. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования: Пер. с англ. –М.: Наука, 1965. –450 с.
4. Рихтер К. Динамические задачи дискретной оптимизации. –М.: Радио и связь, 1985. –136 с.
5. Дегтярев Ю.И. Системный анализ и исследование операций. – М.: Высшая школа, 1996. –335 с.
6. Базара М., Шетти К. Нелинейное программирование. Теория и алгоритмы. – М.: МИР, 1982. –580 с.
7. Таха Х. Введение в исследование операций. – М.: МИР, 2002. –980 с.
8. Исследование операций. В 2-х томах./Под ред. Дж. Моудера и С. Элмаграби. Т.1. Методологические основы и математические методы. –М.: МИР, 1981. –712 с., Т.2. Модели и применение. – М.: МИР, 1981. –677 с..
9. Интрилигатор М. Математические методы оптимизации и экономическая теория: Пер. с англ. –М.: Прогресс, 1975. –606 с.
10. Банди Б. Методы оптимизации. Вводный курс. – М.: Радио и связь, 1988. –128 с.
11. Ашманов С.А., Тимохов А.В. Теория оптимизации в задачах и упражнениях. М.: Наука, 1991. –448 с.

Термины

***Альтернатива**:* конкретный вариант решений или действий.

***Анализ**:* разбиение (расчленение, декомпозиция) объекта исследования на составные части с целью изучения его свойства, связи и отношения. Метод разложения объектов (систем, явлений) и оперирование с упрощенными их компонентами и условиями носит название *аналитического метода*.

Задача математического программирования: задача, состоящая в выборе из заданного допустимого множества значений переменных таких значений, при которых достигается максимум или минимум заданной целевой функции.

Задача синтеза: Задача определения управления по так называемому замкнутому контуру, называется задачей синтеза. Управление по замкнутому контуру означает, что оптимальное управление определяется как функция текущих фазовых координат и времени, т. е. $\{u^*(t)\} = \{u^*(x(t), t)\}$. Если же оптимальное управление определяется как функция времени, т. е. $\{u^*(t)\}$, управление называется по открытому контуру.

Задача стохастического управления: задача управления, которая содержит случайные переменные с фиксированными распределениями.

Задача адаптивного управления: задача управления, которая содержит неопределенности относительно начальных условий на параметры, функций или множества, которые уменьшаются или полностью устраняются по мере развертывания процесса управления.

***Имитационное моделирование*:** процесс конструирования модели реальной системы и постановки

эксперимента на этой модели с целью либо понять поведение системы, либо оценить различные стратегии, обеспечивающие функционирование данной системы в рамках ограничений, накладываемых некоторыми критериями.

Критерий: конкретный показатель привлекательности или полезности различных вариантов решений для участников процесса выбора. Эти показатели называются также признаками, факторами или атрибутами. *Всесторонность* критерия означает, что, зная его значение в определенной ситуации, мы полностью понимаем, в какой степени достигается поставленная цель. *Измеримость* критерия означает, что мы можем установить путем измерений точное (или точечное) значение критерия, с помощью которого оцениваем степень предпочтения или полезности каждой конкретной альтернативы. Сложные проблемы (или решения) обычно оцениваются с помощью *набора критериев*, который должен быть: *полным*, чтобы охватить все значимые аспекты проблемы, *действенным,* чтобы быть с пользой применен в анализе; *разложимым,* чтобы процесс оценивания можно было упростить, разбив его на части; *неизбыточным,* чтобы не дублировать учет различных аспектов и последствий; *минимальным,* чтобы размерность проблемы оставалась по возможности минимальной, т.к. с ростом числа критериев возрастают трудности получения необходимой для анализа и принятия решение информация.

Лицо, принимающее решение (ЛПР): человек, осуществляющий выбор наилучшего варианта решений в конкретной проблемной ситуации или ситуации выбора.

Математическое программирование: раздел теории оптимизации, связанный с исследованием и

решением задачи максимизации или минимизации одной или нескольких функций *скалярной* или *векторной переменной* на подмножестве *конечномерного векторного пространства*, заданного множеством ограничений типа равенства и/или неравенства. Важными разделами математического программирования являются *линейное, нелинейное, квадратичное, целочисленное, динамическое, стохастическое, программирование,* ряд других прикладных задач. Задачи математического программирования содержат целевые функции и ограничения, заданные в той или иной форме. Ключевым понятием в данной области знаний является понятие *выпуклости* функций и множеств. Соответствующий раздел теории оптимизации называется выпуклым программированием

Модель: представление объекта (системы, идеи и т.д.) в некоторой форме, отличной от формы их реального существования. Модели служат средством, помогающим нам в *объяснении, понимании или совершенствовании* систем и их компонентов.

Неформальная информация: информация, получаемая от ЛПР относительно конкретного свойства или качества сравниваемых друг с другом альтернатив.

Полезность: воображаемая мера психологической и потребительской ценности различных решений, действий или подходов.

Решение: выбор альтернативы.

Поддержка принятия решений: совокупность процедур обеспечения ЛПР необходимой информацией и рекомендациями, облегчающими процесс принятия решений.

Принятие решений: особый вид человеческой деятельности, направленный на выбор наилучшего варианта решений или действий.

Процесс принятия решений: процесс, содержащий поиск необходимой информации, построение (или поиск) альтернатив, выбор наилучшей альтернативы.

Принцип максимума Понтрягина: согласно этому принципу, оптимальное управление $\{u^*(t)\} \in U$ максимизирует функцию Гамильтона $H(x, u, y, t) = I(x, u, t) + y^T f(x, u, t)$, где $I(x, u, t)$ – подынтегральная функция в выражении целевого функционала, $y(t)$ – *вектор сопряженных переменных* (по аналогии с неопределенными множителями Лагранжа в задаче нелинейного программирования), $f(x, u, t)$ – вектор – функция закона движения (см. *закон движения)*. Порядок решения задачи оптимального управления с помощью принципа максимума таков: сначала вводится вектор $y(t) = (y_1(t), \ldots, y_n(t))^T$, далее строится функция Гамильтона и отыскиваются функции $\{u(t)\}$, $\{x(t)\}$ и $\{y(t)\}$ из условия максимума функции Гамильтона, удовлетворяющие каноническим уравнениям для координат $x(t)$ $y(t)$ $\dot{x} = \partial H / \partial y, x(t_0) = x_0, \dot{y} = -\partial H / \partial x, y(t_1) = \partial F / \partial x_1$. Эти условия являются необходимыми для существования максимума. Они становятся достаточными, если функция Гамильтона линейна относительно управляющих параметров или если максимум функции Гамильтона представляет собой вогнутую функцию относительно фазовых координат.

Принцип оптимальности: согласно этому принципу, оптимальное поведение обладает тем свойством, что каковы бы не были первоначальное состояние и решение (или управление) в начальный момент времени, последующие решения должны быть

оптимальны по отношению к состоянию, в которое переходит процесс (или систем) после предыдущих решений. Арису принадлежит следующая интерпретация этого принципа: «Если вы не используете наилучшим образом то, чем вы располагаете, то вы никогда не распорядитесь наилучшим образом и тем, что вы могли бы иметь в дальнейшем».

Рациональная модель выбора: модель процесса принятия решений, включающая диагностику проблемы, формулирование целей и критериев для принятия решений, выявление альтернатив, окончательный выбор альтернативы, ее реализацию, контроль и корректировку результатов.

Решающее правило: правило, позволяющее ЛПР определить превосходство одной альтернативы по сравнению с другими альтернативами заданной совокупности.

Риски*:* возможность возникновения условий, которые приведут к негативным последствиям. Для организации термин «*риск»* подразумевает любое событие или действие, которое может неблагоприятно отразиться на достижение организацией своих деловых целей и помешать ей успешно реализовать свою стратегию. Различают *внешние, внутренние, процессные (или операционные), финансовые, рыночные, информационные и т.д. риски. Комплексный подход к управлению рисками* предполагает включение их в общий стратегический план управления организацией с последующей их *идентификацией, оцениванием и управлением.* Эффективными мерами управления рисками считаются: *принятие, передача, уменьшение.* Фразу *«Общество высокого риска не исчезает. Наоборот, риск будет возрастать»* вряд ли следует отнести к разряду *«Законов*

Мерфи» как своеобразное проявление *второго закона термодинамики.*

***Система**:* совокупность объектов произвольной природы, объединенных некоторой формой регулярного взаимодействия или взаимозависимости для выполнения заданной функции или достижения определенной цели. Различают *естественные и искусственные, простые и сложные, открытые и замкнутые, детерминированные и стохастические, статические и динамические* системы.

Системный анализ: наука о комплексном, системном подходе к изучению объектов и явлений окружающей нас действительности. В концепцию системного анализа объединены понятия *системы, информации* и *управления*, что позволяет выделить важный для современной практики класс *управляющих,* или *кибернетических, систем.* Эти системы обладают такими полезными свойствами, как связи с внешним окружением (материальные, энергетические и информационные), наличие управляющих и исполнительных органов, наличие замкнутой цепи обратной связи, использование информации для идентификации и управления.

Системный подход: подход к изучению объекта (или явления) как совокупности взаимосвязанных и взаимодействующих компонентов, который одновременно является составной частью другого объекта (или системы) – так называемой *метасистемы.* Таким образом, согласно системному подходу, изучаемый объект рассматривается и как организованное целое, и как часть другого объекта (системы более высокого порядка).

***Системы поддержки принятия решений (СППР)**:* Программно-аппаратные системы, предназначенные для помощи ЛПР при управлении

сложными объектами в условиях жестких временных ограничений. СППР – это интеллектуализированные системы, основанные на применении баз знаний и машинного вывода на знаниях (так называемые *экспертные системы*). Важной разновидностью СППР являются системы реального времени (СРВ), работающие в ритме управляемых объектов.

Системы управления: системы, в которых имеют место процессы обработки информации и управления. Системы управления состоят из *объекта управления* и *управляющей подсистемы.* В *организационных системах управления* их структурными компонентами являются *объект и субъект управления*.

Стационарные точки: точки области допустимых решений задачи, в которых все частные производные первого порядка целевой функции равны нулю. Из этого определения следует, что все точки локального максимума или минимума целевой функции являются стационарными точками этой функции. В стационарных точках матрица Гессе, состоящая из частных производных второго порядка, является положительно определенной или полуопределенной для точек минимума, и отрицательно определенной или полуопределенной для точек минимума (условия второго порядка)

Теорема Вейерштрасса: фундаментальная теорема математического программирования, формулирующая условия существования глобального максимума или минимума целевой функции. Согласно этой теореме, непрерывная функция, определенная на компактном (т. е. ограниченном по расстоянию и замкнутом) множестве, достигает на внутренней или граничной его точке своего глобального максимума или минимума.

Теория систем: междисциплинарная наука, изучающая законы и принципы создания, функционирования и взаимодействия систем и явлений в природе и обществе.

Уравнение движения: система дифференциальных уравнений, описывающий скорости изменения фазовых координат во времени $dx(t)/dt = f(x(t),u(t), t)$, где $u(t)$ – вектор управления, $x(t)$ - вектор фазовых координат, t – время, $f(x, u, t)$ – заданная вектор-функция. Уравнение движения называется *автономным,* если вектор-функция $f(x, u, t)$ явно не зависит от времени.

Управление: целенаправленное воздействие на объект управления для достижения в нем желательного состояния (или целей). Любая задача управления предполагает формулирование (или определение) следующих трех взаимосвязанных компонентов: начального (или текущего) состояния системы – S_0, ее целевого состояния – S_G и средства или стратегии - S, необходимого для преобразования состояния системы, в котором она находится в настоящее время, в целевое состояние, в котором она окажется в будущем. Эту тройку можно символически представить в виде $C = <S_0, S, S_G>$, где символ C символизирует управление (control).

Фазовые координаты: последовательность $x_1(t), \ldots, x_n(t)$ вещественных чисел, зависящих от времени t, которые характеризуют состояние системы в фиксированный момент времени t. Вектор $x(t) = (x_1(t), \ldots, x_n(t))^T$ называется *фазовым вектором (или фазовой точкой)*. Множество значений этого вектора для промежутка времени $t \in [t_0, t_1]$ называется *фазовой траекторией*.

Функция полезности: функция, определяющая полезность для ЛПР каждой оценки альтернативы из возможного диапазона изменения критерия выбора.

Целевая функция: функция, с помощью которой оцениваются различные альтернативы с точки зрения достижения целей (показатель степени достижения целей).

Цель: конкретные конечные состояния отдельных характеристик организации (системы), достижение которых желательно для организации, другими словами, цель есть конкретное желательное состояние организации в будущем. Цели должны быть *измеримыми, четко сформулированными и однозначными, реалистичными и достижимыми, гибкими*, *согласованными* в смысле краткосрочной и долгосрочной перспективы.

Целевой функционал: функция, координаты которой являются функциями времени. Целевой функционал задачи оптимального управления представляет собой отображение управлений (функций времени) на точки вещественной прямой и записывается в виде $J = J\{u(t)\}$, где $u(t)$ – вектор управления, зависящий от времени t.

Экстремаль: любая траектория движения $\{x(t)\}$, которая удовлетворяет уравнению Эйлера вариационной задачи и граничным условиям $x(t_0) = x_0$, $x(t_1) = x_1$, $t \in [t_0, t_1]$.

Содержание

Глава 4. Детерминированные модели

динамического программирования

Введение ……..…3
4.1. Многошаговые процессы и функциональное
уравнение динамического программирования… 5
4.2. Задача инвестирования……..21
4.3. Задача оптимального резервирования…….33
4.4. Задача распределения двух видов ресурсов……44
4.5. Транспортная задача с нелинейной функцией
затрат ……55
4.6. Задача оптимального управления……..…73
4.7. Связи с вариационной задачей и принципом
максимума …….…....81
4.8. Задачи и вопросы по ДП ……..107

Глава 5. Численные методы оптимизации
5.1. Стратегия поиска и условия сходимости……117
5.2. Методы безусловной оптимизации…….…….120
5.3. Методы условной оптимизации ……..136
5.4. Методы штрафных и барьерных функций……..149
5.5. Методы возможных направлений……156
5.6. Метод проекции Розена…….166
Приложения
1**.** Задача выпуклого программирования решения …..185
2. Условие для направления спуска……..…….187
3. Геометрическое условие оптимальности……..…....188
Литература ……..189
Термины ……..190

Св. план 2008 г., поз. 244.

Саркисян Рафаэль Еремович

Системный анализ и принятие решений.
Часть 2. Детерминированные методы
динамического программирования.
Численные методы оптимальности

Учебное пособие

Подписано в печать Тираж 100 экз.
Усл. –печ. л. Заказ № Формат

127994 Москва, ул. Образцова, 15
Типография МИИТа.