Классификация методов оптимизации

Методы оптимизации занимаются построением оптимальных решений для математических моделей. В эту дисциплину не входит само построение математических моделей.

Но именно вид модели определяет методы (или методы), используемые для построения оптимального решения.

В большинстве случаев математическую модель объекта можно представить в виде целевой функции $f(\bar{x})$ или критерия оптимальности (иногда без ограничений), которую нужно максимизировать или минимизировать. Т.е. необходимо найти максимум или минимум поставленной задачи, причем $\bar{x} \in D$ – области возможных значений $\bar{x} = (x_1, x_2, \ldots, x_n)$. Как правило, область допустимых значений D задается. Тогда задача формулируется следующим образом:

$$f(\bar{x}) \to \min(\max), \quad \bar{x} \in D. \tag{1}$$

или

$$f(\bar{x}) \to \min(\max), \tag{1′}$$

при

$$\bar{x} \in D.$$

Область допустимых значений D определяется системой линейных или нелинейных ограничений, накладываемых на $\bar{x}$

$$D = \{\bar{x} : g_j(\bar{x}) \le b_j, \; j = \overline{1, m}\}. \tag{2}$$

В реальных задачах ограничения на область возможных значений переменных модели отсутствуют чрезвычайно редко, потому что, как правило, переменные бывают связаны с некоторым ограниченным ресурсом. Но все-таки с задачами без ограничений сталкиваются. Это бывает в условиях “неограниченных” ресурсов или при наличии условий, не накладывающих ограничений на переменные задачи. В таком случае мы имеем безусловную задачу, задачу без ограничений:

$$f(\bar{x}) \to \min(\max), \quad \bar{x} \in \bar{x}. \tag{3}$$

Сложность задачи зависит от вида критерия $f(\bar{x})$ и функций $q_j(\bar{x})$, определяющих допустимую область. Функции могут быть линейными и нелинейными, быть непрерывнми или принимать дискретные значения. Область возможных значений может быть выпуклой и невыпуклой, несвязной, представлять собой дискретное множество точек. В зависимости от этого задачи могут быть одноэкстремальными или много экстремальными, могут использоваться одни или другие методы поиска решения.

Например, если функции $f(\bar{x})$ и $q_j(\bar{x})$ линейны, имеем задачу линейного программирования и можем использовать для поиска решения методы линейного программирования (варианты симплекс-метода).

Если функции $f(\bar{x})$ и $q_j(\bar{x})$ нелинейны, используем методы нелинейного программирования. Если при этом минимизируем выпуклую $f(\bar{x})$ при выпуклых функциях $q_j(\bar{x})$, то знаем, что задача одноэкстремальна (выпуклое нелинейное программирование).

Если $q_j(\bar{x})$ линейны, а минимизируемая $f(\bar{x})$ представляет собой квадратичную выпуклую функцию, можем использовать алгоритмы квадратичного программирования.

При минимизации вогнутой функции на выпуклой области можем столкнуться с многоэкстремальностью задачи и необходимостью поиска глобального экстремума.

Если на переменные, входящие в задачу, наложено требование целочисленности или дискретности, то используются методы дискретного программирования, среди которых наиболее хорошо разработаны методы решения линейных дискретных задач.

Если система ограничений отсутствует и $f(\bar{x})$ представляет собой нелинейную функцию, то для решния задачи (3), то есть для определения минимума или максимума этой функции используются различные алгоритмы поиска. В зависимости от количества информации о функции $f(\bar{x})$, используемой в алгоритме, могут применяться прямые методы поиска, методы первого или второго порядка.

Прямые методы поиска – методы нулевого порядка, в которых при поиске экстремума используется информация только о самой функции и не используется информация о ее производных. Плюсом таких методов является возможность оптимизации функций, аналитическое представление которых неизвестно, т.е. они определены только алгоритмически.

Методы первого порядка – при поиске решения используют не только информацию о самой функции, но и о ее производных первого порядка. К таким методам относятся различные градиентные алгоритмы.

Методы второго порядка – при поиске решения используют информацию о самой функции и о ее производных первого и второго порядка. Сюда относятся метод Ньютона и его модификации.

Методы одномерного поиска.

Как правило, реально мы сталкиваемся с необходимостью решения многомерных задач. И очень редко практическая задача изначально явля-

ется одномерной. Для многомерных задач мы используем многомерные методы. Но в многомерных методах на этапах поиска с задачами одномерной минимизации в направлении некоторого вектора сталкиваются почти обязательно.

Существует множество методов поиска минимума или максимума функции на отрезке. Наиболее известными из них являются методы дихотомии (деления отрезка пополам), золотого сечения и Фибоначчи. В каждом из этих методов последовательно сокращается интевал, содержащий точку минимума.

Метод дихотомии

Предполагаем, что минимизируемая функция $f(x)$ унимодальна на отрезке $[a_0, b_0]$ и необходимо найти минимум данной функции на заданном отрезке с некоторой точностью ε. Вычисляем две точки согласно следующим формулам:

$$x_1 = \frac{a_0 + b_0 - \varepsilon}{2} \text{ и } x_2 = \frac{a_0 + b_0 + \varepsilon}{2},$$

где $\delta < \varepsilon$. И в каждой из найденных точек вычисляем значения функции: $f(x_1)$ и $f(x_2)$.

Далее сокращаем интервал неопределенности и получаем интервал $[a_1, b_1]$ следующим образом. Если $f(x_1) < f(x_2)$, то $a_1 = a_0$ и $b_1 = x_2$.

В противном случае, если $f(x_1) > f(x_2)$, то $a_1 = x_1$ и $b_1 = b_0$.

Далее по аналогичным формулам вычисляем следующую пару точек x_1 и x_2. С помощью найденных точек определяем новый интервал неопределенности.

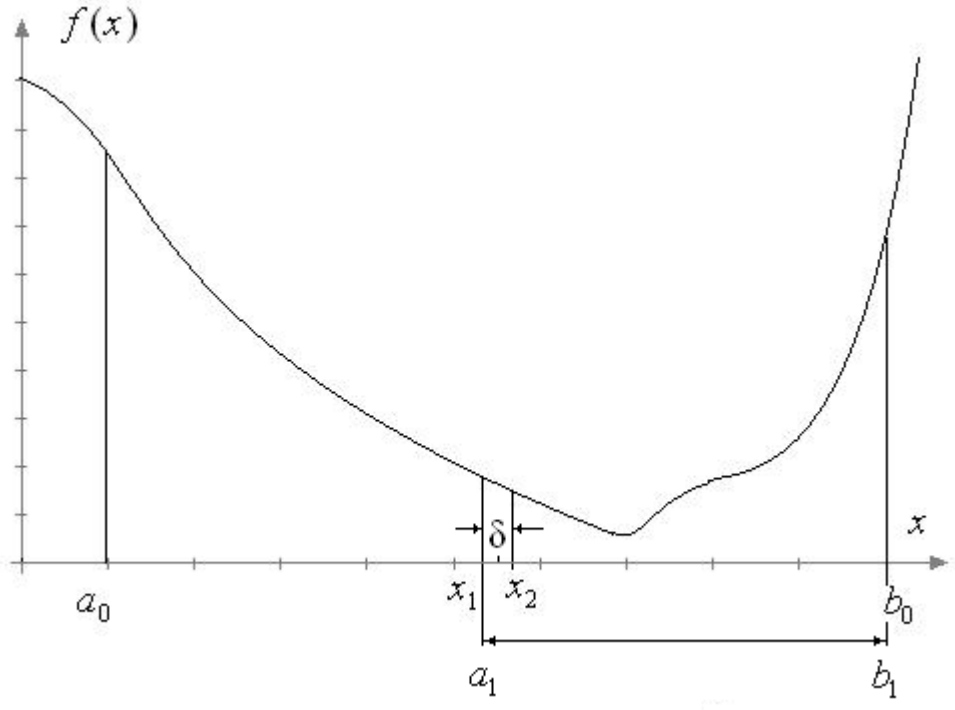


Рис. 1.

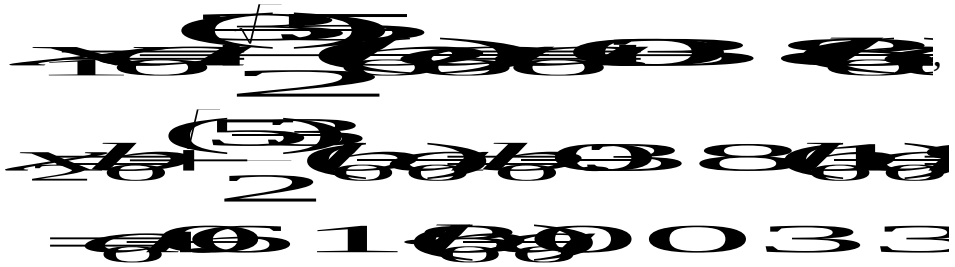
Поиск заканчивается, если длина интервала неопределенности $[a_k, b_k]$ на текущей итерации становится меньше заданной точности: $|b_k - a_k| < \varepsilon$.

В данном методе на каждой итерации минимизируемая функция $f(x)$ вычисляется дважды, а интервал неопределенности сокращается практическ в два раза (при малых $\delta < \varepsilon$).

Метод золотого сечения

Данный метод позволяет найти минимум заданной функции на заданной области $[a_0, b_0]$, как правило, с меньшими вычислительными затратами, чем метод дихотомии.

На первой итерации находим две точки по следующим формулам:



и вычисляем значения функции в них. Обратим внимание, что на первой итерации находим 2 точки и два вычисления $f(x)$.

Сокращаем интервал неопределенности.

1) Если $f(x_1) < f(x_2)$, то $a_1 = a_0$, $b_1 = x_2$, $x_2 = x_1$.

2) В противном случае, если $f(x_1) > f(x_2)$, то $a_1 = x_1$, $b_1 = b_0$, $x_1 = x_2$.

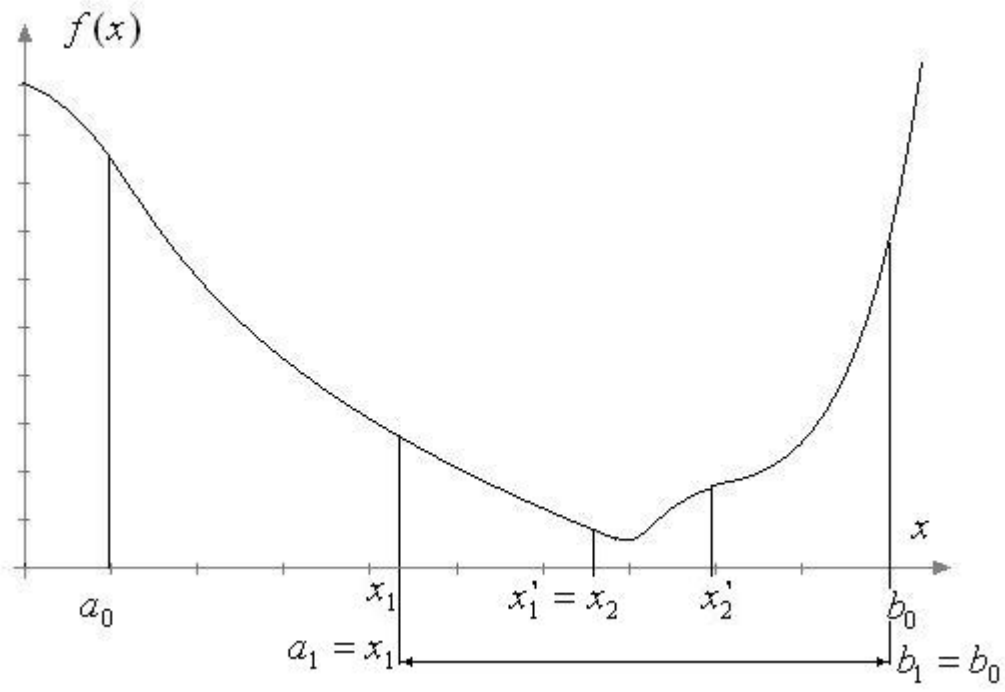


Рис. 2.

На последующих итерациях производим расчет только той точки и ее функции, которую необходимо обновить: в случае 1) вычисляем новое значение x_1 и $f(x_1)$; в случае 2) x_2 и $f(x_2)$.

Поиск прекращается при выполнении $|b_k - a_k| < \varepsilon$.

На i-й итерации интервал неопределенности сокращается до величины $0.618(b_i - a_i)$. Это меньше, чем в 2 раза, но зато всего один раз вычисляем значение $f(x)$ в новой точке.

Метод Фибоначчи

Числа Фибоначчи подчиняются соотношениям:

$$F_{n+2} = F_{n+1} + F_n,$$

где $n = 1, 2, 3\ldots$ и $F_1 = F_2$.

С помощью индукции можно показать, что n-е число Фибоначчи вычисляется по формуле:

$$F_n = \frac{\left[\frac{1+\sqrt{5}}{2}\right]^n - \left[\frac{1-\sqrt{5}}{2}\right]^n}{\sqrt{5}},$$

где $n = 1, 2, 3\ldots$

Из данной формулы видно, что при больших значениях n выполняется соотношение:

$$F_n \approx \frac{\left[\frac{1+\sqrt{5}}{2}\right]^n}{\sqrt{5}},$$

так, что числа Фибоначчи с увеличением n растут очень быстро.

Сам алгоритм метода Фибоначчи очень похож на алгоритм метода золотого сечения. На начальном интервале вычисляются точки по следующим формулам:

$x_1 = a + \frac{F_n}{F_{n+2}}(b - a)$ и $x_2 = a + \frac{F_{n+1}}{F_{n+2}}(b - a)$.

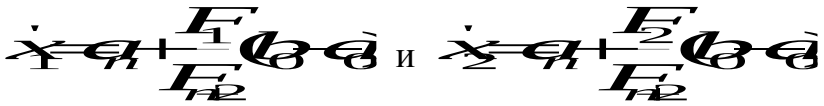
Интервал неопределенности сокращается точно так же, как в методе золотого сечения (см. рис.2). И на новой итерации вычисляется только 1 новая точка и значение функции в ней.

На k-й итерации получаем точку минимума, которая совпадает с одной из точек, которые вычисляются по формулам

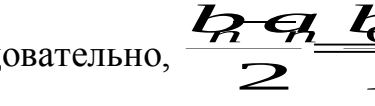
$$x_1 = a_k + \frac{F_{n-k}}{F_{n-k+2}}(b_k - a_k), \quad x_2 = a_k + \frac{F_{n-k+1}}{F_{n-k+2}}(b_k - a_k),$$

и расположены на отрезке $[a_k, b_k]$ симметрично относительно его середины.

Нетрудно заметить, что при $k = n$ точки:



совпадают и делят отрезок $[a_n, b_n]$ пополам.

Следовательно,  .

Отсюда можно выбрать n из условия  .

Таким образом, это условие позволяет до начала работы алгоритма определить число итераций, необходимое для определения минимума с точностью ε при начальной величине интервала $[a_0, b_0]$.

С ростом n из-за того, что F_n / F_{n+2}- бесконечная десятичная дробь, возможно “искажение” метода и потеря интервала с точкой минимума (вследствие погрешностей вычислений).

Следует также отметить, что при практическом применении метод золотого сечения по эффективности, скорости сходимости и точности получаемого решения практически не уступает методу Фибоначчи. А алгоритмически реализация метода золотого сечения является более простой.

При реализации многомерных алгоритмов используются не только рассмотренные выше методы. Применяются различные эвристические алгоритмы, используется интерполяция (аппроксимация) минимизируемой функции более простой с последующим поиском минимума этой интерполирующей функции, например, квадратичной. Зачастую это оказывается очень эффективным приемом.

Прямые методы

Прямые методы или методы нулевого порядка не требуют знания целевой функции в явном виде. Они не требуют регулярности и непрерывности целевой функции и существования производных. Это является существенным достоинством при решении сложных технических и экономических задач.

При реализации прямых методов существенно сокращается этап подготовки решения задачи, так как нет необходимости в определении первых и вторых производных. Прямые методы в основном носят эвристический характер. К прямым методам относится целый ряд алгоритмов, которые отличаются по своей эффективности. Методы предназначены для решения безусловных задач оптимизации

$$\min_{x \in E^n} f(\bar{x}).$$

Алгоритм Гаусса

Это простейший алгоритм, заключающийся в том, что на каждом шаге (каждой итерации) минимизация осуществляется только по одной компоненте вектора переменных $\bar{x}$.

Пусть нам дано начальное приближение $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$. На первой итерации находим значение минимума функции при изменяющейся первой координате и фиксированных остальных компонентах, т.е.

$$x_1^1 = \arg\min_{x_1} f(x_1, x_2^0, \ldots, x_n^0).$$

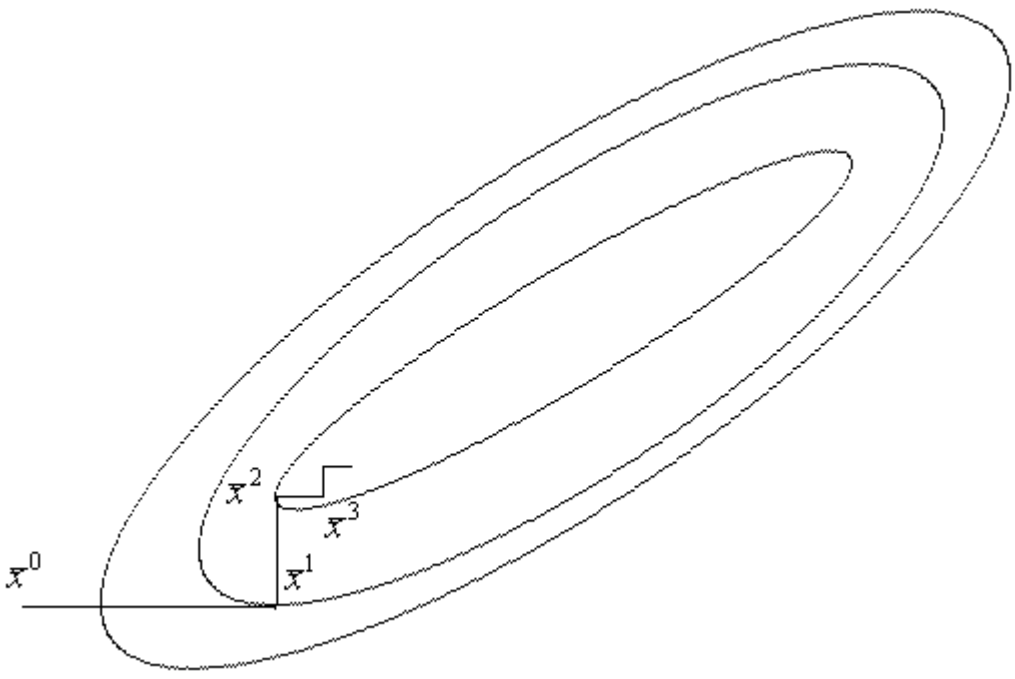


Рис. 1.

В результате получаем новую точку $\bar{x}^1 = (x_1^1, x_2^0, \ldots, x_n^0)$. Далее из точки $\bar{x}^1$ ищем минимум функции, изменяя вторую координату и считая фиксированными все остальные координаты. В результате получаем значение

$$x_2^1 = \arg\min_{x_2} f(x_1^1, x_2, x_3^0, \ldots, x_n^0)$$

и новую точку $\bar{x}^2 = (x_1^1, x_2^1, x_3^0, \ldots, x_n^0)$. Продолжая процесс, после n шагов получаем точку $\bar{x}^n = (x_1^1, x_2^1, \ldots, x_n^1)$, начиная с которой процесс возобновляется.

В качестве условий прекращения поиска можно использовать следующие два критерия:

1) $\left\| f(\bar{x}^{k+1}) - f(\bar{x}^k) \right\| \le \varepsilon$.

2) $\left| x_i^{k+1} - x_i^k \right| \le \varepsilon_i, \ \forall i$.

Метод очень прост, но не очень эффективен. Проблемы могут возникнуть, когда линии уровня сильно вытянуты и "эллипсоиды" ориентированы, например, вдоль прямых вида $x_1 = x_2$. В подобной ситуации поиск быстро застревает на дне такого оврага, а если начальное приближение оказывается на оси "эллипсоида", то процесс так и останется в этой точке.

Хорошие результаты получаются в тех случаях, когда целевая функция представляет собой выпуклую сепарабельную функцию вида

$$f(x) = \sum_{i=1}^{n} f_i(x_i).$$

Алгоритм Хука и Дживса

В данном алгоритме предлагается логически простая стратегия поиска, в которой используются априорные сведения о топологии функции и, в то же время отвергаются уже устаревшая информация об этой функции. В интерпретации Вуда алгоритм включает два основных этапа:

1) исследующий поиск вокруг базисной точки $\bar{x}^k$;
2) поиск по "образцу", т.е. в направлении, выбранном для минимизации.

Задается начальная точка поиска $\bar{x}^0$ и начальное приращение (шаг) $\Delta\bar{x}^0$, после чего начинается исследующий поиск.

Исследующий поиск:

Делаем пробный шаг по переменной x_1, т.е. определяем точку $x_1^0 + \Delta x_1^0$ и вычисляем значение функции для точки $(x_1^0 + \Delta x_1^0, x_2^0, \ldots, x_n^0)$.

Если значение функции в данной точке больше, чем значение функции $f(\bar{x}^0)$, то делаем пробный шаг по этой же переменной, но в противо-

положном направлении. Если значение функции в точке  также больше, чем $f(\bar{x}^0)$, то оставляем точку x_1^0 без изменений. Иначе заменяем точку $\bar{x}^0$ на $\bar{x}'$ или на $\bar{x}''$ в зависимости от того, где значение функции меньше исходного. Из вновь полученной точки делаем пробные шаги по оставшимся координатам, используя тот же самый алгоритм.

Если в процессе исследующего поиска не удается сделать ни одного удачного пробного шага, то $\Delta\bar{x}$ необходимо скорректировать (уменьшить). После чего вновь переходим к исследующему поиску.

Если в процессе исследующего поиска сделан хоть один удачный пробный шаг, то переходим к поиску по образцу.

Поиск по образцу:

После исследующего поиска мы получаем точку $\bar{x}^{01}$. Направление $\bar{x}^{01}-\bar{x}^0$ определяет направление, в котором функция уменьшается. Поэтому проводим минимизацию функции в указанном направлении, решая задачу

В поиске по "образцу" величина шага по каждой переменной пропорциональна величине шага на этапе исследующего поиска. Если удастся сделать удачный шаг в поиске по "образцу", то в результате находим новое приближение  , где

Из точки $\bar{x}^1$ начинаем новый исследующий поиск и т.д.

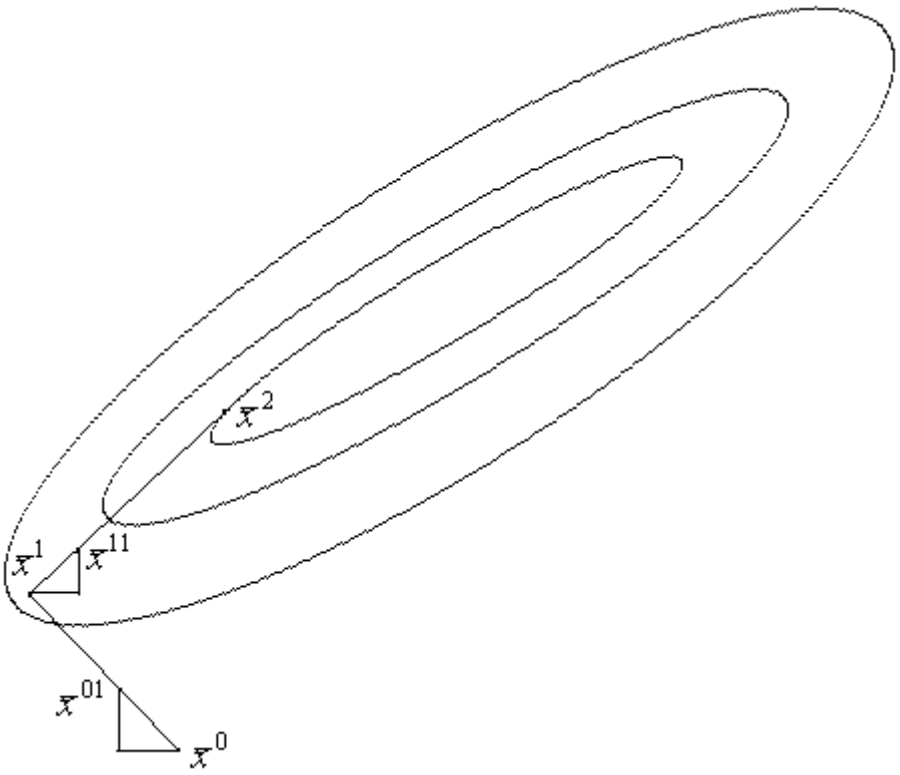


Рис. 2

Существуют модификации алгоритма, в которых в процессе исследующего поиска ищется минимум по каждой переменной или в процессе поиска по образцу ищется не минимум функции, а просто делается шаг в заданном найденном направлении с фиксированным значением параметра λ.

Алгоритм Розенброка

Этот итерационный метод имеет некоторое сходство с алгоритмом Хука и Дживса. Метод Розенброка также называется методом вращающихся координат. Этот метод существенно эффективнее предыдущих методов, особенно при минимизации функций овражного типа. Общая идея метода заключается в том, что выбирается система ортогональных направлений $\bar{S}_1^0, \bar{S}_2^0, \ldots \bar{S}_n^0$, в каждом из которых последовательно ищется минимальное значение, после чего система направлений поворачивается так, чтобы одна из осей совпала с направлением полного перемещения, а остальные были ортогональны между собой.

Пусть $\bar{x}^0$ - вектор начального приближения; $\bar{S}_1^0, \bar{S}_2^0, \ldots \bar{S}_n^0$ - система ортогональных направлений. На первой итерации это может быть ортонормированная система координат. Начиная с $\bar{x}^0$, последовательно осуществляем минимизацию функции $f(\bar{x})$ в соответствующих направлениях $\bar{S}_1^0, \bar{S}_2^0, \ldots \bar{S}_n^0$, находя последовательные приближения:

$$\bar{x}_1^0 = \bar{x}_0^0 + \lambda_1 \bar{S}_1^0, \quad \lambda_1 = \arg\min_{\lambda} f(\bar{x}_0^0 + \lambda \bar{S}_1^0),$$

….

$$x_n^0 = x_{n+1}^0 + \lambda_n \bar{S}_n, \quad \lambda_n = \arg\min_{\lambda} f\left(x_{n+1}^0 + \lambda \bar{S}_n\right).$$

Следующая итерация начнется с точки $\bar{x}^1 = \bar{x}_n^0$. Если не изменить систему направлений, то будем иметь алгоритм Гаусса. Поэтому после завершения очередного k-го этапа вычисляем новые направления поиска. Ортогональные направления поиска поворачиваются так, чтобы они оказались вытянутыми вдоль "оврага" ("хребта") и, таким образом, исключается взаимодействие переменных ($x_i x_j$). Направления поиска вытягиваются вдоль главных осей квадратичной аппроксимации целевой функции.

Рассмотрим некоторую k-ю итерацию алгоритма Розенброка. В результате минимизации по каждому из ортогональных направлений на данной итерации мы имеем систему параметров $\lambda_1^k, \lambda_2^k, \ldots \lambda_n^k$, с помощью которых определим систему векторов $\bar{A}_1^k, \bar{A}_2^k, \ldots \bar{A}_n^k$, вычисляемых по формулам следующего вида:

$$\bar{A}_1^k = \lambda_1^k \bar{S}_1^k;$$

$$\bar{A}_2^k = \lambda_2^k \bar{S}_2^k + \lambda_1^k \bar{S}_1^k;$$

$$\ldots;$$

$$\bar{A}_n^k = \lambda_n^k \bar{S}_n^k + \ldots + \lambda_1^k \bar{S}_1^k.$$

С помощью системы векторов $\bar{A}_1^k, \bar{A}_2^k, \ldots \bar{A}_n^k$ строим новую систему ортогональных направлений $\bar{S}_1^{k+1}, \bar{S}_2^{k+1}, \ldots \bar{S}_n^{k+1}$. Причем первый вектор направляют так, чтобы он совпал с направлением общего перемещения на k-м шаге, а остальные получаются с помощью процедуры ортогонализации Грама-Шмидта:

$$\bar{S}_1^{k+1} = \frac{\bar{A}_1^k}{\left\|\bar{A}_1^k\right\|};$$

$$\bar{B}_2^k = \bar{A}_2^k - \left(\bar{A}_2^k, \bar{S}_1^{k+1}\right)\bar{S}_1^{k+1};$$

$$\bar{S}_2^{k+1} = \frac{\bar{B}_2^k}{\left\|\bar{B}_2^k\right\|}; \qquad (1)$$

$$\bar{B}_n^k = \bar{A}_n^k - \sum_{i=1}^{n-1}\left(\bar{A}_n^k, \bar{S}_i^{k+1}\right)\bar{S}_i^{k+1};$$

$$\bar{S}_l^{k+1} = \frac{\bar{B}_l^k}{\left\|\bar{B}_l^k\right\|}; \quad l = 2, \ldots n$$

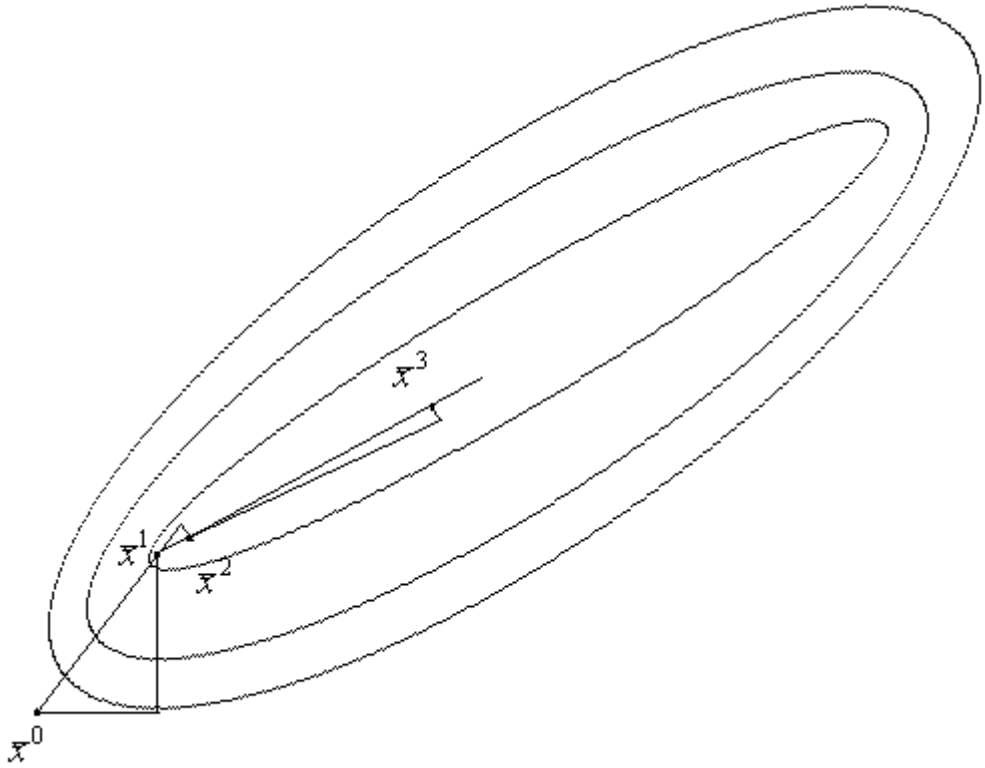


Рис. 3.

Для работы алгоритма необходимо, чтобы ни один из векторов системы $\bar{S}_1^0, \bar{S}_2^0, \ldots \bar{S}_n^0$ не стал нулевым вектором. Для этого в алгоритме следует располагать параметры $\lambda_1^k, \lambda_2^k, \ldots \lambda_n^k$ в порядке убывания по абсолютному значению, т.е. $|\lambda_1^k| > |\lambda_2^k| > \ldots > |\lambda_n^k|$. Тогда если любые m из λ_i^k обращаются в нуль, то отыскиваются новые направления по (1) только для тех $(n-m)$ направлений, для которых $\lambda_i^k \neq 0$, оставшиеся же m направлений остаются неизменными: $\bar{S}_i^{k+1} = \bar{S}_i^k$, $i = \overline{(n-m+1), n}$. Так как первые $(n-m)$ векторов взаимно ортогональны, $\lambda_i^k = 0$, $i = \overline{(n-m+1), n}$, первые $(n-m)$ векторов не будут иметь составляющих в направлениях $\bar{S}_i^{k+1}$, $i = \overline{(n-m+1), n}$. А поскольку эти последние направления взаимно ортогональны, то из этого следует, что все направления являются взаимно ортогональными.

Палмером было показано, что $\bar{B}_{j+1}^k$ и $\left\|\bar{B}_{j+1}^k\right\|$ пропорциональны λ_j^k (при условии, что $\sum_{i=j}^{n} (\lambda_i^k)^2 \neq 0$). Следовательно, при вычислении $\bar{S}_j^{k+1} = \bar{B}_j^k / \left\|\bar{B}_j^k\right\|$, величина λ_j^k сокращается, и, таким образом, $\bar{S}_j^{k+1}$ остается

определенным, если даже $\lambda_j^k = 0$. Имея это в виду, Палмер предложил для вычисления $\bar{S}_j^{k+1}$ следующие соотношения:

$$\bar{A}_i^k = \sum_{j=i}^{n} \lambda_j^k \cdot \bar{S}_j^k, \quad i = 1,..,n,$$

$$\bar{S}_i^{k+1} = \frac{\bar{A}_i^k \left\| \bar{A}_i^k \right\|^2 - \bar{A}_i^k \cdot \left\| \bar{A}_i^k \right\|^2}{\left\| \bar{A}_i^k \right\| \cdot \left\| \bar{A}_i^k \right\| \cdot \left[\left\| \bar{A}_i^k \right\|^2 - \left\| \bar{A}_i^k \right\|^2 \right]^{1/2}}, \quad i = 2,...,n,$$

$$\bar{S}_1^{k+1} = \frac{\bar{A}_1^k}{\left\| \bar{A}_1^k \right\|}.$$

Критерии останова алгоритма могут быть стандартными (т.е. описанными в предыдущих алгоритмах прямых методов).

Симплексный метод Нелдера-Мида или поиск по деформируемому многограннику

В процессе поиска осуществляется работа с регулярными симплексами. Регулярные многогранники в пространстве E^n называются **симплексами**. Для $n = 2$ регулярный симплекс представляет собой равносторонний треугольник; при $n = 3$ - тетраэдр и т.д.

Координаты вершин регулярного симплекса в n-мерном пространстве могут быть определены следующей матрицей D, в которой столбцы представляют собой вершины симплекса, пронумерованные от 1 до $(n+1)$, а строки – координаты вершин, $i = \overline{1,n}$. Матрица имеет размерность $n \times (n+1)$:

$$D = \begin{bmatrix} 0 & d_1 & d_2 & \dots & d_2 \\ 0 & d_2 & d_1 & \dots & d_2 \\ 0 & d_2 & d_2 & \dots & d_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & d_2 & d_2 & \dots & d_1 \end{bmatrix}_{n \times (n+1)},$$

где:

$$d_1 = \frac{t}{n\sqrt{2}}\left(\sqrt{n+1} + n - 1\right); \quad d_2 = \frac{t}{n\sqrt{2}}\left(\sqrt{n+1} - 1\right);$$

t – расстояние между вершинами.

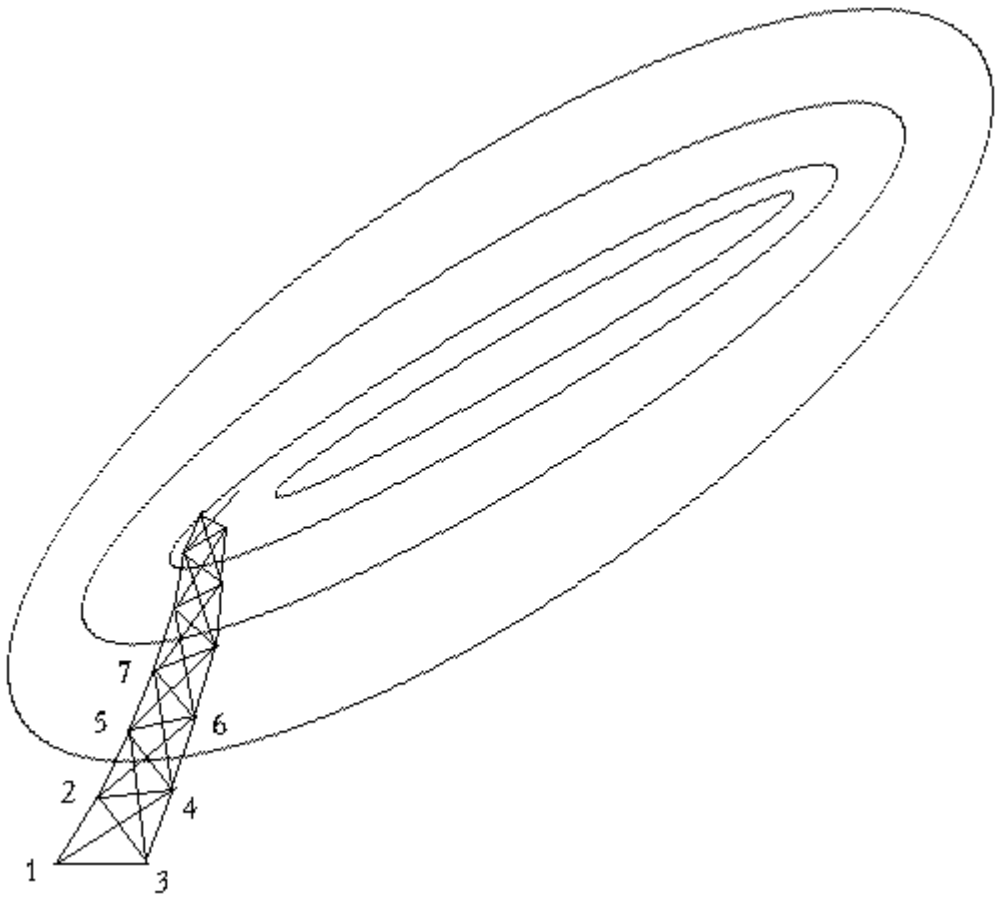


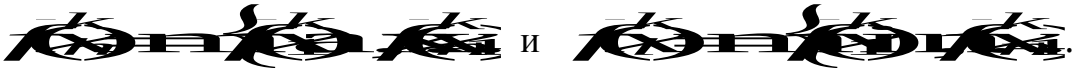
Рис.4.

В самом простом виде симплексный алгоритм заключается в следующем. Строится регулярный симплекс. Из вершины, в которой $f(\bar{x})$ максимальна (т.1) проводится проектирующая прямая через центр тяжести симплекса. Затем т.1 исключается и строится новый *отраженный* симплекс из оставшихся старых точек и одной новой, расположенной на проектирующей прямой на надлежащем расстоянии от центра тяжести.

Продолжение этой процедуры, в которой каждый раз исключается вершина, где целевая функция максимальна, а также использование правил уменьшения размера симплекса и предотвращение циклического движения в окрестности экстремума позволяет достаточно эффективно определять минимум для "хороших" функций. Но для "овражных" функций такой поиск неэффективен.

В симплексном алгоритме Нелдера и Мида минимизация функций n переменных осуществляется с использованием деформируемого многогранника.

Будем рассматривать k-ю итерацию алгоритма. Путь $\bar{x}^k = [x_1^k, x_2^k, \ldots, x_n^k]$, $i = 1 \ldots (n+1)$, является i-й вершиной в E^n на k-м этапе поиска, $k = 0, 1, 2, \ldots$, и пусть значения целевой функции в вершине $f(\bar{x}_i^k)$. Отметим вершины с минимальным и максимальным значениями. И обозначим их следующим образом:



Многогранник в E^n состоит из $n+1$ вершин $\bar{x}_1^k, \bar{x}_2^k, \ldots, \bar{x}_{n+1}^k$. Обозначим через $\bar{x}_{n+2}^k$ - центр тяжести вершин без точки $\bar{x}_h^k$ с максимальным значением функции. Координаты этого центра вычисляются по формуле:

$$x_{n+2,j}^k = \frac{1}{n}\left[\sum_{i=1}^{n+1} x_{ij}^k - x_{hj}^k\right], \quad j = 1, \ldots, n. \qquad (1)$$

Начальный многогранник обычно выбирается в виде регулярного симплекса (с вершиной в начале координат). Можно начало координат поместить в центр тяжести. Процедура отыскания вершин в E^n, в которых $f(\bar{x})$ имеет лучшее значение, состоит из следующих операций: 1) отражения; 2) растяжения; 3) сжатия; 4) редукции.

1. Отражение. Отражение – проектирование точки $\bar{x}_h^k$ через центр тяжести $\bar{x}_{n+2}^k$ в соответствии со следующим соотношением:

$$\bar{x}_{n+3}^k = \bar{x}_{n+2}^k + \alpha(\bar{x}_{n+2}^k - \bar{x}_h^k), \qquad (2)$$

где $\alpha > 0$ - коэффициент отражения.

Вычисляем значение функции в найденной точке $f(\bar{x}_{n+3}^k)$. Если значение функции в данной точке $f(\bar{x}_{n+3}^k) < f(\bar{x}_l^k)$, то переходим к четвертому пункту алгоритма – операции редукции.

Если $f(\bar{x}_{n+3}^k) < f(\bar{x}_l^k) \wedge f(\bar{x}_{n+3}^k) < f(\bar{x}_s^k)$, то выполняем операцию растяжения.

В противном случае, если $f(\bar{x}_{n+3}^k) < f(\bar{x}_l^k) \wedge f(\bar{x}_{n+3}^k) \geq f(\bar{x}_s^k)$, то выполняется операция сжатия.

2. Растяжение. Эта операция заключается в следующем. Если $f(\bar{x}_{n+3}^k) < f(\bar{x}_s^k)$ (меньше минимального значения на k-м этапе), то вектор $(\bar{x}_{n+3}^k - \bar{x}_{n+2}^k)$ растягивается в соответствии с соотношением

$$\bar{x}_{n+4}^k = \bar{x}_{n+2}^k + \gamma(\bar{x}_{n+3}^k - \bar{x}_{n+2}^k), \qquad (3)$$

где $\gamma > 0$ - коэффициент растяжения.

Если $f(\bar{x}_{n+4}^k) < f(\bar{x}_s^k)$, то $\bar{x}_l^k$ заменяется на $\bar{x}_{n+4}^k$ и процедура продолжается с операции 1) при $k = k+1$. В противном случае $\bar{x}_l^k$ заменяется на $\bar{x}_{n+3}^k$ и переходим к операции отражения 1).

3. Сжатие. Если $f(\bar{x}_{n+3}^k) > f(\bar{x}_i^k)$ для $\forall i \; i \neq h$, то вектор $(\bar{x}_h^k - \bar{x}_{n+2}^k)$ сжимается в соответствии с формулой

$$\bar{x}_{n+5}^k = \bar{x}_{n+2}^k + \beta(\bar{x}_h^k - \bar{x}_{n+2}^k),$$

где $0<\beta<1$ - коэффициент сжатия. После этого, точка $\bar{x}_h^k$ заменяется на $\bar{x}_{n+5}^k$, и переходим к операции отражения 1) с $k=k+1$. Заново ищется $\bar{x}_h^{k+1}$.

4. Редукция. Если $f(\bar{x}_{n+3}^k) > f(\bar{x}_h^k)$, то все векторы $\left(\bar{x}_i^k - \bar{x}_l^k\right)$, где $i=\overline{1,(n+1)}$ уменьшаются в два раза с отсчетом от точки $\bar{x}_l^k$ по формуле

$$\bar{x}_i^k = \bar{x}_l^k + 0{,}5\left(\bar{x}_i^k - \bar{x}_l^k\right), \quad i=\overline{1,(n+1)}$$

и переходим к операции отражения (на начало алгоритма с $k=k+1$).

В качестве критерия останова могут быть взяты те же критерии, что и в остальных алгоритмах. Можно также использовать критерий останова следующего вида:

$$\left\{\frac{1}{n+1}\sum_{i=1}^{n+1}\left[f(\bar{x}_i^k) - f(\bar{x}_l^k)\right]^2\right\}^{\frac{1}{2}} \le \varepsilon.$$

Выбор коэффициентов α, β, γ обычно осуществляется эмпирически. После того как многогранник подходящим образом промассштабирован, его размеры должны поддерживаться неизменными пока изменения в топологии задачи не потребуют многогранника другой формы. Чаще всего выбирают $\alpha = 1$, $0{,}4 \le \beta \le 0{,}6$, $2 \le \gamma \le 3$.

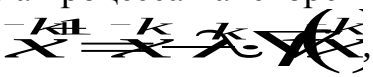
Методы первого порядка

К методам первого порядка относятся алгоритмы, в которых в процессе поиска кроме информации о самой функции используется информация о производных первого порядка. К группе таких методов относятся различные градиентные методы.

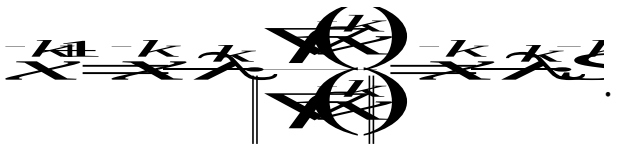
Алгоритм наискорейшего спуска

Градиент функции в любой точке показывает направление наибольшего локального увеличения $f(\bar{x})$. Поэтому при поиске минимума $f(\bar{x})$, следует двигаться в направлении противоположном направлению градиента $\nabla f(\bar{x})$ в данной точке, то есть в направлении *наискорейшего спуска*.

Итерационная формула процесса наискорейшего спуска имеет вид

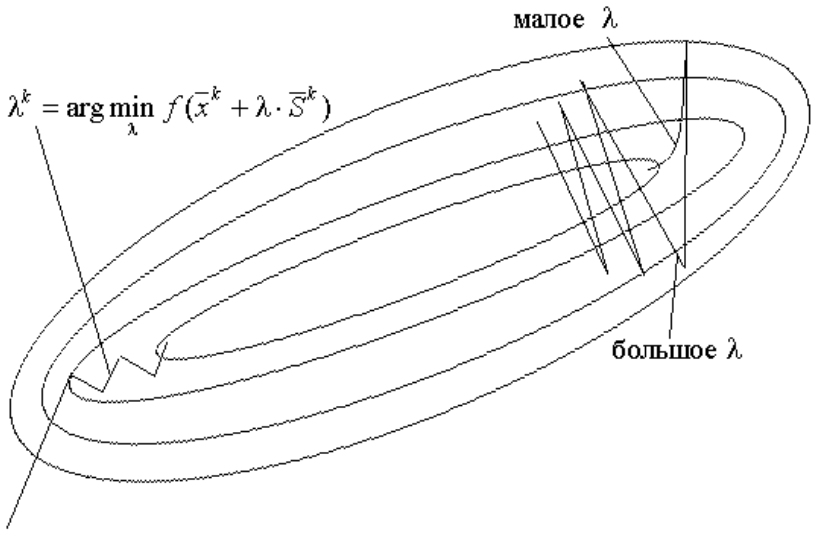


или



Очевидно, что в зависимости от выбора параметра λ траектории спуска будут существенно различаться. При большом значении λ траектория спуска будет представлять собой колебательный процесс, а при слишком больших λ процесс может расходиться. При выборе малых λ траектория спуска будет плавной, но и процесс будет сходиться очень медленно. Обычно λ выбирают из условия

решая одномерную задачу минимизации с использованием некоторого одномерного метода. В этом случае получаем алгоритм наискорейшего спус-



ка.

Рис.

Если λ определяется в результате одномерной минимизации, то градиент в точке очередного приближения будет ортогонален направлению предыдущего спуска $\nabla f(\bar{x}^{k+1}) \perp \bar{S}^k$.

Вообще говоря, процедура наискорейшего спуска может закончиться в стационарной точке любого типа, в которой $\nabla f(\bar{x}) = \bar{0}$. Поэтому следует проверять, не завершился ли алгоритм в седловой точке.

Эффективность алгоритма зависит от вида минимизируемой функции. Алгоритм наискорейшего спуска сойдется за одну итерацию при любом начальном приближении для функции $f(x) = x_1^2 + x_2^2$, но сходимость будет очень медленной, например, в случае функции вида $f(x) = x_1^2 + 16x_2^2$. В тех ситуациях, когда линии уровня минимизируемой функции представляет собой прямолинейный или, хуже того, криволинейный "овраг" эффективность алгоритма оказывается очень низкой. Очевидно, что хорошие результаты может давать предварительное масштабирование функций.

Процесс наискорейшего спуска обычно быстро сходится вдали от точки экстремума и медленно в районе экстремума. Поэтому метод наискорейшего спуска нередко используют в комбинации с другими алгоритмами.

Метод сопряженных градиентов

В алгоритме наискорейшего спуска на каждом этапе поиска используется только текущая информация о функции $f(\bar{x}^k)$ и градиенте $\nabla f(\bar{x}^k)$. В алгоритмах сопряженных градиентов используется информация о поиске на предыдущих этапах спуска.

Направление поиска на текущем шаге $\bar{S}^k$ строится как линейная комбинация наискорейшего спуска на данном шаге $-\nabla f(\bar{x}^k)$ и направлений спуска на предыдущих шагах $\bar{S}^0, \bar{S}^1, \ldots \bar{S}^{k-1}$. Веса в линейной комбинации выбираются таким образом, чтобы сделать эти направления сопряженными. В этом случае квадратичная функция будет минимизироваться за n шагов алгоритма.

При выборе весов используется только текущий градиент и градиент в предыдущей точке.

В начальной точке $\bar{x}^0$ направление спуска $\bar{S}^0 = -\nabla f(\bar{x}^0)$:

$$\bar{x}^1 = \bar{x}^0 + \lambda \bar{S}^0, \qquad (1)$$

где λ^0 выбирается из соображений минимальности целевой функции в данном направлении  . Новое направление поиска

$$\bar{S}^1 = -\nabla f(\bar{x}^1) + \omega_1 \bar{S}^0, \tag{2}$$

где ω_1 выбирается так, чтобы сделать направления $\bar{S}^1$ и $\bar{S}^0$ сопряженными по отношению к матрице H:

$$(\bar{S}^0)^T H \bar{S}^1 = 0. \tag{3}$$

Для квадратичной функции справедливы соотношения:

 ,

где $\Delta\bar{x} = \bar{x} - \bar{x}^0$,

 .

Если положить $\bar{x} = \bar{x}^1$, тогда $\bar{x}^1 - \bar{x}^0 = \lambda^0 \bar{S}^0$ и

 . (4)

Воспользуемся (4), чтобы исключить $(\bar{S}^0)^T$ из уравнения (3). Для квадратичной функции $H = H^T$, поэтому транспонировав (4) и умножив справа на H^{-1}, получим  и далее

 .

Следовательно, для сопряженности $\bar{S}^0$ и $\bar{S}^1$

 .

Вследствие изложенных ранее свойств сопряженности все перекрестные члены исчезают. Учитывая, что $\bar{S}^0 = -\nabla f(\bar{x}^0)$, получим для ω_1 следующее соотношение:

$$\omega_1 = \frac{\nabla f(\bar{x}^1)^T \nabla f(\bar{x}^1)}{\nabla f(\bar{x}^0)^T \nabla f(\bar{x}^0)} = \frac{\|\nabla f(\bar{x}^1)\|^2}{\|\nabla f(\bar{x}^0)\|^2}. \tag{5}$$

Направление поиска $\bar{S}^2$ строится в виде линейной комбинации векторов $-\nabla f(\bar{x}^2)$, $\bar{S}^0$, $\bar{S}^1$, причем так, чтобы оно было сопряженным с $\bar{S}^1$. Если распространить сделанные выкладки на $\bar{S}^2, \bar{S}^3, \ldots$, опуская их содержание и учитывая, что $(\bar{S}^k)^T \nabla f(\bar{x}^{k+1}) = 0$ приводит к $\nabla f(\bar{x}^k) \nabla f(\bar{x}^{k+1}) = 0$, можно получить общее выражение для ω_k:

$$\omega_k = -\frac{\left\|\nabla f\left(\bar{x}^k\right)\right\|^2}{\left\|\nabla f\left(\bar{x}^{k-1}\right)\right\|^2}. \qquad (6)$$

Все весовые коэффициенты, предшествующие ω_k, что особенно интересно, оказываются нулевыми.

Полностью алгоритм описывается следующей последовательностью действий:

(a) В точке начального приближения $\bar{x}^0$ вычисляется $\bar{S}^0 = -\nabla f\left(\bar{x}^0\right)$.

(b) На k-м шаге с помощью одномерного поиска в направлении $\bar{S}^k$ определяется минимум функции, то есть решается задача

$$\lambda_k = \arg\min_{\lambda} f\left(\bar{x}^k + \lambda \bar{S}^k\right)$$

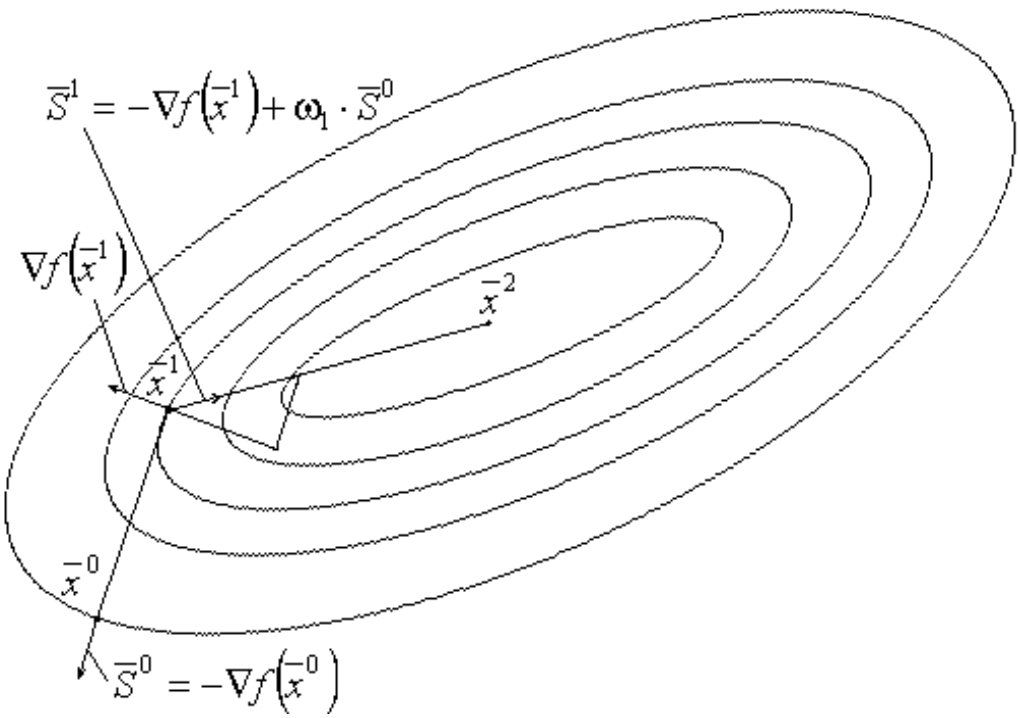
и находится очередное приближение $\bar{x}^{k+1} = \bar{x}^k + \lambda_k \bar{S}^k$.

(c) Вычисляется $f\left(\bar{x}^{k+1}\right)$ и $\nabla f\left(\bar{x}^{k+1}\right)$.

(d) Определяется направление $\bar{S}^{k+1} = -\nabla f\left(\bar{x}^{k+1}\right) + \omega_{k+1}\bar{S}^k$.

(e) Алгоритм заканчивается, если $\left\|\bar{S}^k\right\| < \varepsilon$, где ε – заданная величина.

После $n+1$ итераций ($k = n$), если не произошел останов алгоритма, процедура циклически повторяется с заменой $\bar{x}^0$ на $\bar{x}^{n+1}$ и возвратом на первый пункт алгоритма. Если исходная функция является квадратичной, то $(n+1)$-е приближение даст точку экстремума данной функции. Описанный алгоритм с построением ω_k по формулам (6) соответствует методу



сопряженных градиентов **Флетчера-Ривса.**

Рис.

В модификации **Полака-Рибьера** (Пшеничного) метод сопряженных градиентов отличается только вычислением

(7)

В случае квадратичных функций обе модификации примерно эквивалентны, однако в модификации Полака-Рибьера алгоритм иногда оказывается несколько эффективнее.

Многопараметрический поиск

Милем и Кентреллом предложен метод поиска, основанный на использовании двух подбираемых параметров для минимизации $f(\bar{x})$ в каждом направлении поиска. В этом алгоритме последовательность действий определяется формулой:

(1)

где $\Delta\bar{x}^{k+1} = \bar{x}^{k} - \bar{x}^{k-1}$.

На каждом шаге решается задача минимизации по двум параметрам:

После чего находится очередное приближение по формуле (1). При этом можно показать, что

На первом шаге $\Delta\bar{x}^{k-1} = 0$, а $\bar{x}^{0}$ должно быть задано. На k-м шаге:

- Вычисляется $\bar{x}^{k}$, $\nabla f\left(\bar{x}^{k}\right)$ и $\Delta\bar{x}^{k+1} = \bar{x}^{k} - \bar{x}^{k-1}$.
- Пользуясь одним из эффективных методов, например, методом Ньютона находятся с требуемой точностью λ_0^k и λ_1^k.
- По соотношению (1) вычисляют $\bar{x}^{k+1}$ и переходят к пункту 1.
- Каждый $(n+1)$-й шаг начинается с $\Delta\bar{x}^{k-1} = 0$.
- Процесс заканчивается, когда $\left|\Delta f(\bar{x})\right| < \varepsilon$.

На квадратичных функциях алгоритм по эффективности близок к методу сопряженных градиентов.

Крэгг и Леви распространили данный метод на случай большего числа параметров. На каждом шаге очередное приближение находится как

при $m \leq n-1$, а, следовательно, на каждом шаге при минимизации $f(\bar{x})$ в заданном направлении решается задача вида

Достоинства и недостатки такого подхода очевидны.

Методы второго порядка

В методах второго порядка при поиске минимума используют информацию о функции и ее производных до второго порядка включительно. К этой группе относят метод Ньютона и его модификации.

В основе метода лежит квадратичная аппроксимация $f(\bar{x})$, которую можно получить, отбрасывая в рядах Тейлора члены третьего и более высокого порядков:

 , (1)

где  – матрица Гессе, представляющая собой квадратную матрицу вторых частных производных $f(\bar{x})$ в точке $\bar{x}^k$.

Направление поиска $\bar{S}^k$ в методе Ньютона определяется следующим образом. Если заменить в выражении (1) $\bar{x}$ на $\bar{x}^{k+1}$ и обозначить  , то получим

 , (2)

Минимум функции $f(\bar{x})$ в направлении $\Delta\bar{x}^k$ определяется дифференцированием $f(\bar{x})$ по каждой из компонент $\Delta\bar{x}$ и приравниванием к нулю полученных выражений

 . (3)

Это приводит к

 , (4)

 (5)

В данном случае и величина шага и направление поиска точно определены.

Если $f(\bar{x})$ - квадратичная функция (выпуклая вниз), то для достижения минимума достаточно одного шага.

Но в общем случае нелинейной функции $f(\bar{x})$ за один шаг минимум не достигается. Поэтому итерационную формулу (5) обычно приводят к виду:

 , (6)

где λ^k - параметр длины шага, или к виду

 (7)

Направление поиска определяется вектором

Итерационный процесс (6) или (7) продолжается до тех пор, пока не будет выполнен некоторый критерий останова.

Условием, гарантирующим сходимость метода Ньютона в предположении, что функция $f(\bar{x})$ дважды дифференцируема, заключается в том, что матрица $H^{-1}(\bar{x}^k)$ должна быть положительно определенной.

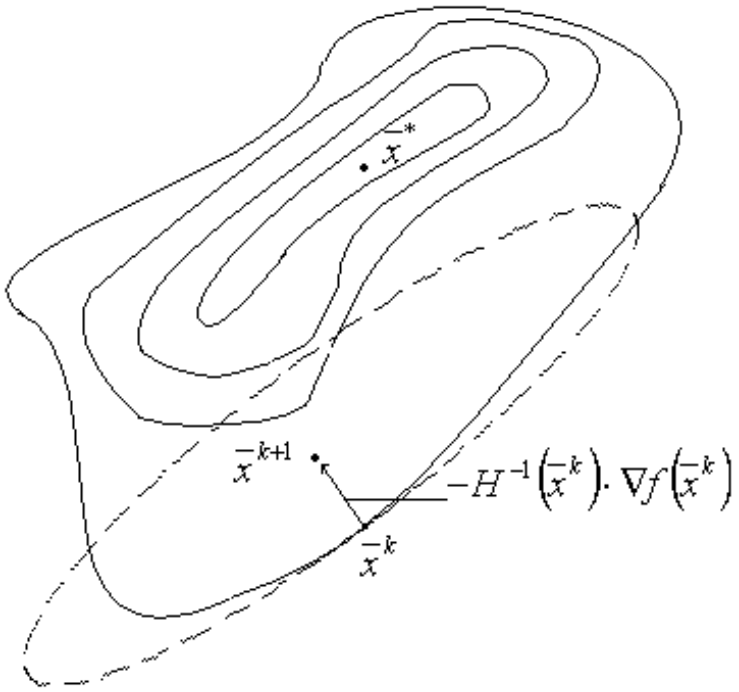
Иногда определенную сложность вызывает вычисление на каждом шаге матрицы $H^{-1}(\bar{x}^k)$. Тогда вместо метода Ньютона используют его модификацию. Суть модифицированного метода Ньютона заключается в том, что при достаточно хорошем начальном приближении вычисляется матрица $\left[\nabla^2 f(\bar{x}^0)\right]^{-1}$ и в дальнейшем на всех итерациях вместо $\left[\nabla^2 f(\bar{x}^k)\right]^{-1}$ используется $\left[\nabla^2 f(\bar{x}^0)\right]^{-1}$.

Очередные приближения определяются соотношением

(8)

Естественно, что число итераций, необходимое для достижения минимума, обычно возрастает, но в целом процесс может оказаться экономичнее.

Рис.



Градиентные методы, в частности метод наискорейшего спуска, обладают линейной скоростью сходимости. Метод Ньютона обладает квадратичной скоростью сходимости.

Применение метода Ньютона оказывается очень эффективным при условии, что выполняются необходимые и достаточные условия его сходимости. Однако само исследование необходимых и достаточных условий сходимости метода в случае конкретной $f(\bar{x})$ может быть достаточно сложной задачей.

Статистические методы поиска

Статистические методы или ***методы случайного поиска*** получили достаточно широкое распространение при построении оптимальных решений в различных приложениях. Это объясняется в первую очередь тем, что с ростом размерности задач резко снижается эффективность регулярных методов поиска (детерминированных): так называемое “проклятие размерности”. Во-вторых, зачастую информация об оптимизируемом объекте слишком мала для того, чтобы можно было применить детерминированные методы. Достаточно часто статистические алгоритмы используют при поиске оптимального решения в системах управления, когда отклик системы можно получить только при задании управляющих воздействий $\bar{x}$ на ее входах. В таких ситуациях статистические алгоритмы могут оказаться значительно эффективнее детерминированных.

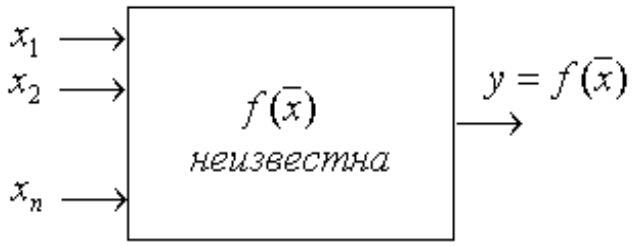


Рис.

Наибольший эффект применение статистических методов приносит при решении задач большой размерности или при поиске глобального экстремума.

Под случайными или статистическими методами поиска будем понимать методы, использующие элемент случайности либо при сборе информации о целевой функции при пробных шагах, либо для улучшения значений функции при рабочем шаге. Случайным образом может выбираться направление спуска, длина шага, величина штрафа при нарушении ограничения и т.д.

Статистические алгоритмы обладают рядом достоинств:

(a) простота реализации и отладки программ;
(b) надежность и помехоустойчивость;
(c) универсальность;
(d) возможность введения операций обучения в алгоритм поиска;
(e) возможность введения операций прогнозирования оптимальной точки (оптимального решения).

А основными недостатками являются большое количество вычислений минимизируемой функции и медленная сходимость в районе экстремума.

Принято считать, что преимущество статистических методов проявляется с ростом размерности задач, так как вычислительные затраты в детерминированных методах поиска с ростом размерности растут быстрее, чем в статистических алгоритмах.

Простой случайный поиск

Пусть нам необходимо решить задачу минимизации функции $f(\bar{x})$ при условии, что $\bar{x} \in [\bar{A}, \bar{B}]$.

В данной области по равномерному закону выбираем случайную точку $\bar{x}_1$ и вычисляем в ней значение функции $y_1 = f(\bar{x}_1)$. Затем выбираем таким же образом случайную точку $\bar{x}_2$ и вычисляем $y_2 = f(\bar{x}_2)$. Запоминаем минимальное из этих значе-

ний и точку, в которой значение функции минимально. Далее генерируем новую точку. Делаем N экспериментов, после чего лучшую точку берем в качестве решения задачи (в которой функция имеет минимальное значение среди всех случайно сгенерированных).

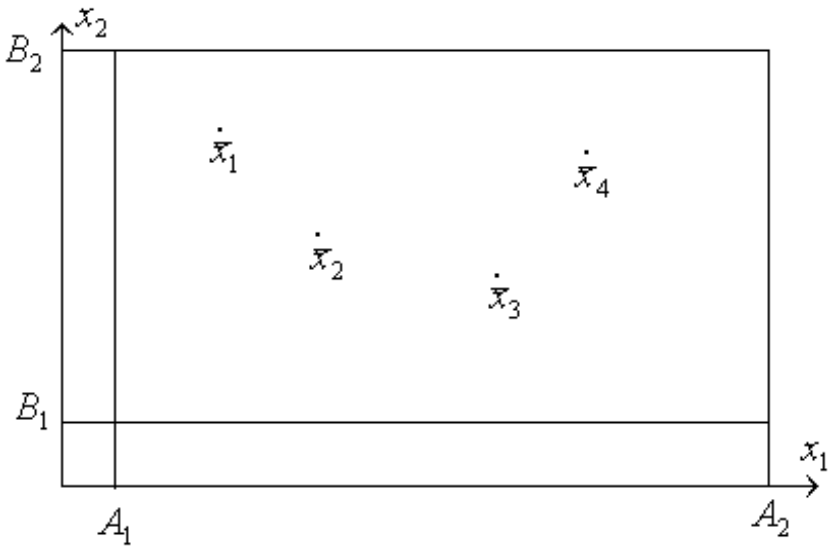


Рис.

Оценим число экспериментов, необходимое для определения решения (точки минимума) с заданной точностью. Пусть n - размерность вектора переменных. Объем n-мерного прямоугольника, в котором ведется поиск минимума,

$$V=\prod_{i=1}^{n}(B_i-A_i).$$

Если необходимо найти решение с точностью ε_i, $i=\overline{1,n}$, по каждой из переменных, то мы должны попасть в окрестность оптимальной точки с объемом

$$V_\varepsilon=\prod_{i=1}^{n}\varepsilon_i.$$

Вероятность попадания в эту окрестность при одном испытании равна $P_\varepsilon=\frac{V_\varepsilon}{V}$.

Вероятность непопадания равна $1-P_\varepsilon$. Испытания независимы, поэтому вероятность непопадания за N экспериментов равна $(1-P_\varepsilon)^N$.

Вероятность того, что мы найдем решение за N испытаний: $P=1-(1-P_\varepsilon)^N$.

Отсюда нетрудно получить оценку необходимого числа испытаний N для определения минимума с требуемой точностью:

$$N\geq\frac{\ln(1-P)}{\ln(1-P_\varepsilon)}.$$

Опираясь на заданную точность ε_i, $i=\overline{1,n}$, величину V, можно определить P_ε и, задаваясь вероятностью P, посмотреть, как меняется требуемое количество экспериментов N в зависимости от P_ε и P (см. табл.).

Таблица

P_ε \ P	0.8	0.9	0.95	0.99	0.999
0.1	16	22	29	444	66
0.025	64	91	119	182	273
0.01	161	230	299	459	688
0.005	322	460	598	919	1379
0.001	1609	2302	2995	4603	6905

При решении экстремальных задач на областях со сложной геометрией обычно вписывают эту область в n-мерный параллелепипед. А далее генерируют в этом n-мерном параллелепипеде случайные точки по равномерному закону, оставляя только те, которые попадают в допустимую область.

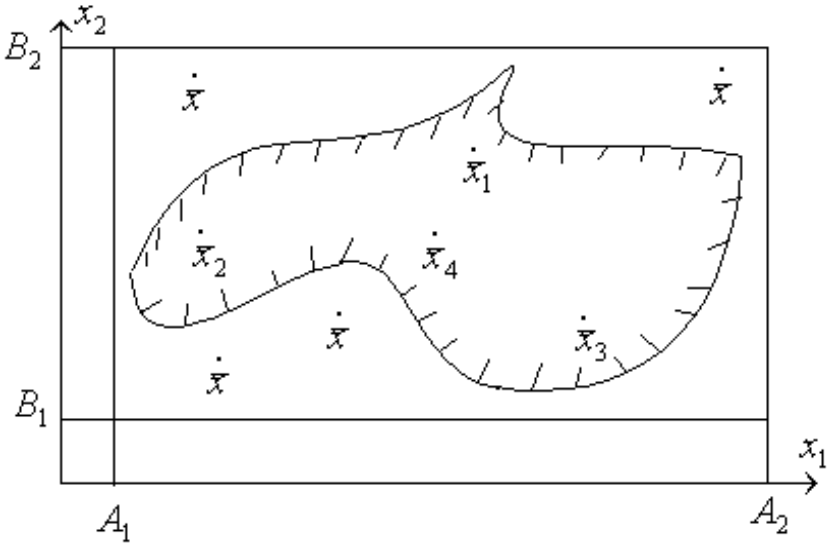


Рис.

Различают направленный и ненаправленный случайный поиск.

(f) **Ненаправленный случайный поиск.** При таком поиске все последующие испытания проводят совершенно независимо от результатов предыдущих. Сходимость такого поиска очень мала, но имеется важное преимущество, связанное с возможностью решения многоэкстремальных задач (искать глобальный экстремум). Примером является рассмотренный простой случайный поиск.

(g) **Направленный случайный поиск.** В этом случае отдельные испытания связаны между собой. Результаты проведенных испытаний используются для формирования последующих. Сходимость таких методов, как правило, выше, но сами методы обычно приводят только к локальным экстремумам.

Простейшие алгоритмы направленного случайного поиска

Алгоритм парной пробы. В данном алгоритме четко разделены пробный и рабочий шаги.

Пусть $\bar{x}^k$ – найденное на k-м шаге наименьшее значение минимизируемой функции $f(\bar{x})$. По равномерному закону генерируется случайный единичный вектор

$\bar{\xi}$ и по обе стороны от исходной точки $\bar{x}^k$ делаются две пробы: проводим вычисление функции в точках $\bar{x}_{1,2}^k = \bar{x}^k \pm g\bar{\xi}$, где g -величина пробного шага.

Рабочий шаг делается в направлении наименьшего значения целевой функции. Очередное приближение определяется соотношением

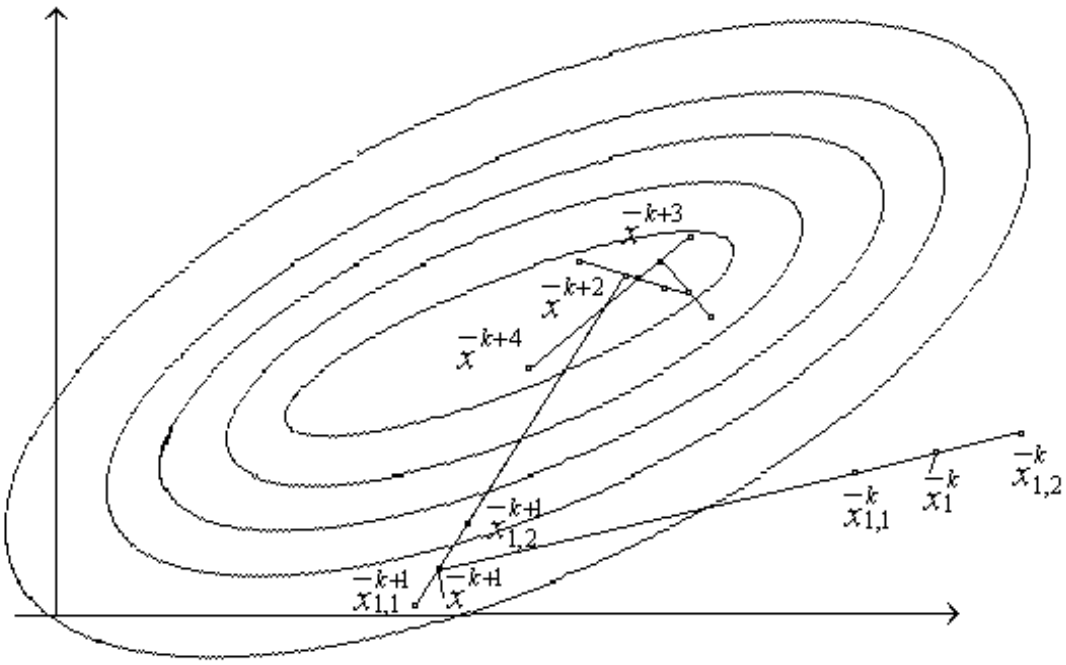


Рис.

Особенностью данного алгоритма является его повышенная тенденция к “блужданию”. Даже найдя экстремум, алгоритм может увести процесс поиска в сторону.

Алгоритм наилучшей пробы. На k -м шаге мы имеем точку $\bar{x}^k$. Генерируется m случайных единичных векторов $\bar{\xi}_1, .., \bar{\xi}_m$. Делаются пробные шаги в направлениях $g\bar{\xi}_1 .. g\bar{\xi}_n$ и в точках $\bar{x}^k + g\bar{\xi}_1$, $\bar{x}^k + g\bar{\xi}_n$ вычисляются значения функции. Выбирается тот шаг, который приводит к наибольшему уменьшению функции: $\bar{\xi}^* = \arg\min_{i=1..n} f(\bar{x}^k + g\bar{\xi}_i)$. И в данном направлении делается шаг

$$\Delta\bar{x}^k = \lambda \cdot \bar{\xi}^*.$$

Параметр λ может определяться как результат минимизации по направлению, определяемому наилучшей пробой, или выбираться по определенному закону.

С увеличением числа проб выбранное направление приближается к направлению $-\nabla f(\bar{x})$.

Если функция $f(\bar{x})$ близка к линейной, то есть возможность ускорить поиск, выбирая вместе с наилучшей и наихудшую пробу. Тогда рабочий шаг можно делать или в направлении наилучшей, или в направлении, противоположном наихудшей пробе.

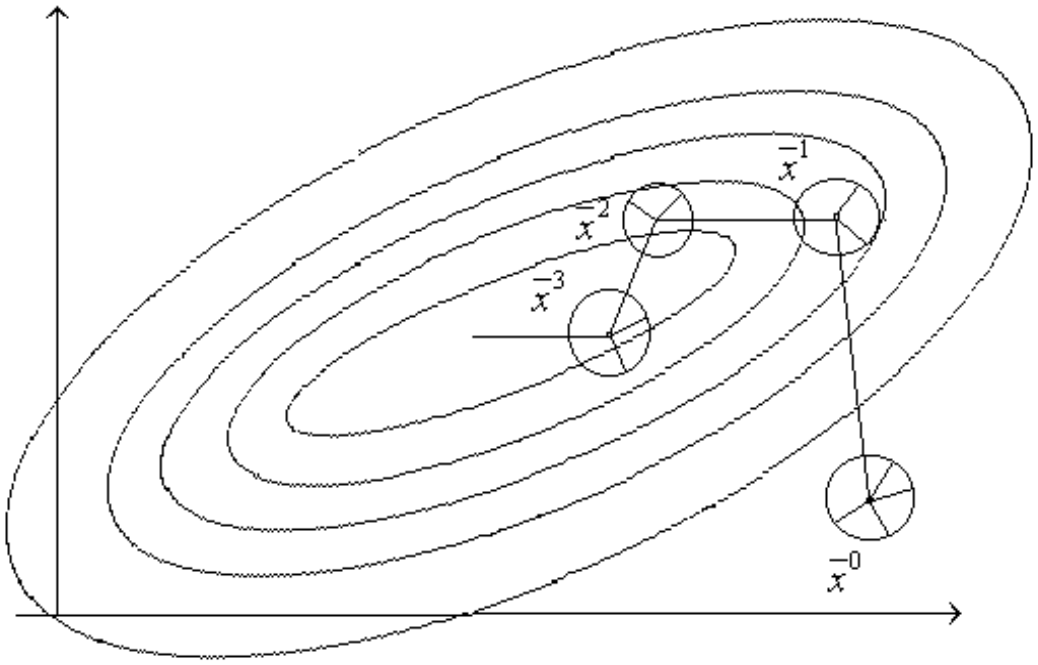


Рис.

Метод статистического градиента. Из исходного состояния $\bar{x}^k$ делается m независимых проб $g\bar{\xi}_1 \dots g\bar{\xi}_n$ в m случайных направлениях, а затем вычисляются соответствующие значения минимизируемой функции в этих точках. Для каждой пробы запоминаем приращения функции

$$\Delta f_j = f(\bar{x}^k + g\bar{\xi}_j) - f(\bar{x}^k).$$

После этого формируем векторную сумму

$$\Delta \bar{f} = \sum_{j=1}^{m} \bar{\xi}_j \cdot \Delta f_j .$$

В пределе при $m \to \infty$ направление $\Delta \bar{f}$ совпадает с направлением градиента целевой функции. При конечном m вектор $\Delta \bar{f}$ представляет собой статистическую оценку направления градиента. В направлении $\Delta \bar{f}$ делается рабочий шаг и, в результате, очередное приближение определяется соотношением

$$\bar{x}^{k+1} = \bar{x}^k - \lambda \frac{\Delta \bar{f}}{\|\Delta \bar{f}\|}.$$

При выборе оптимального значения λ, которое минимизирует функцию в заданном направлении, мы получаем статистический вариант метода наискорейшего спуска. Существенное преимущество перед детерминированными алгоритмами заключается в возможности принятия решения о направлении рабочего шага при $m < n$. При $m = n$ и неслучайных ортогональных рабочих шагах, направленных вдоль осей координат, алгоритм вырождается в градиентный метод.

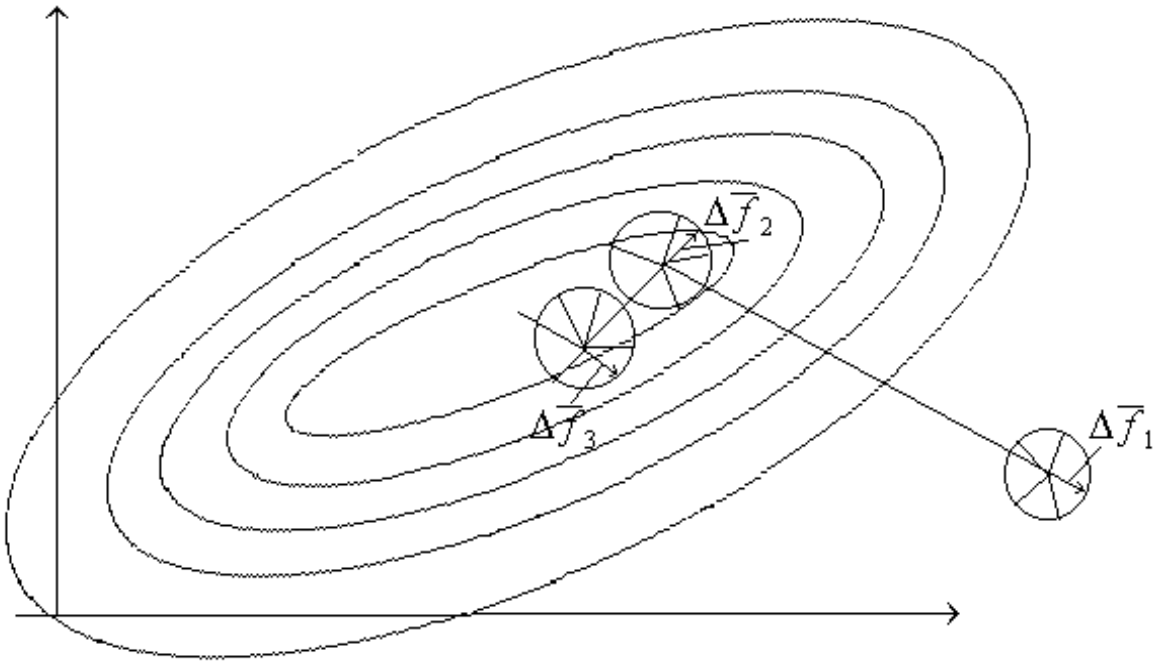


Рис.

Алгоритм наилучшей пробы с направляющим гиперквадратом. Внутри допустимой области строится гиперквадрат. В этом гиперквадрате случайным образом разбрасывается m точек $\overline{x}_1, \ldots, \overline{x}_m$, в которых вычисляются значения функции. Среди построенных точек выбираем наилучшую. Таким образом, на 1-м этапе координаты случайных точек удовлетворяют неравенствам $a_i^1 \le x_i \le b_i^1$, $i = \overline{1,n}$, и $\overline{x}^1 = \arg\min_{j=\overline{1,m}} f(\overline{x}_j)$ – точка с минимальным значением целевой функции.

Опираясь на эту точку, строим новый гиперквадрат. Точка, в которой достигается минимум функции на k-м этапе, берется в качестве центра нового гиперквадрата на $(k+1)$-м этапе.

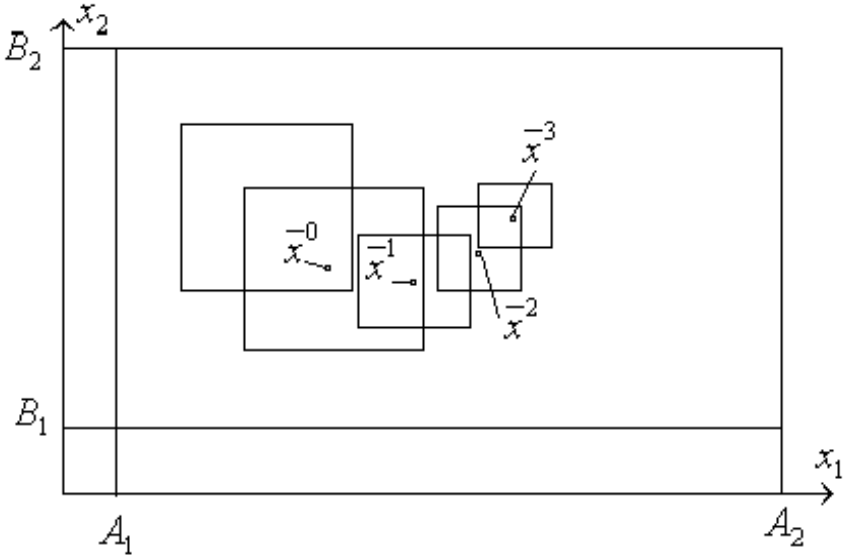


Рис.

Координаты вершин гиперквадрата на $(k+1)$-м этапе определяются соотношениями

$$a_i^{k+1} = x_i^{k+1} - \frac{b_i - a_i}{2}, \quad b_i^{k+1} = x_i^{k+1} + \frac{b_i - a_i}{2},$$

где $\bar{x}^k$ – наилучшая точка в гиперквадрате на k-м этапе.

В новом гиперквадрате выполняем ту же последовательность действий, случайным образом разбрасывая m точек. В результате осуществляется направленное перемещение гиперквадрата в сторону уменьшения функции.

В алгоритме с обучением стороны гиперквадрата могут регулироваться в соответствии с изменением по некоторому правилу параметра α, определяющего стратегию изменения стороны гиперквадрата. В этом случае координаты вершин гиперквадрата на $(k+1)$-м этапе будут определяться соотношениями

$$a_i^{k+1} = x_i^{k+1} - \frac{b_i - a_i}{2\alpha}, \quad b_i^{k+1} = x_i^{k+1} + \frac{b_i - a_i}{2\alpha}.$$

Хорошо выбранное правило регулирования стороны гиперквадрата приводит к достаточно эффективному алгоритму поиска.

В алгоритмах случайного поиска вместо направляющего гиперквадрата могут использоваться направляющие гиперсферы, направляющие гиперконусы.

Алгоритмы глобального поиска

Случайный поиск приобретает решающее значение при решении многоэкстремальных задач и оптимизации сложных объектов. В общем случае решение многоэкстремальных задач без элемента случайности практически невозможно.

Рассмотрим некоторые подходы к поиску глобального экстремума.

Алгоритм 1. В допустимой области D случайным образом выбирают точку $\bar{x}_1 \in D$. Приняв эту точку за исходную и используя некоторый детерминированный метод или алгоритм направленного случайного поиска, осуществляется спуск в точку локального минимума $\bar{x}_1^* \in D$, в области притяжения которого оказалась точка $\bar{x}_1$.

Затем выбирается новая случайная точка $\bar{x}_2 \in D$ и по той же схеме осуществляется спуск в точку локального минимума $\bar{x}_2^* \in D$, и т.д.

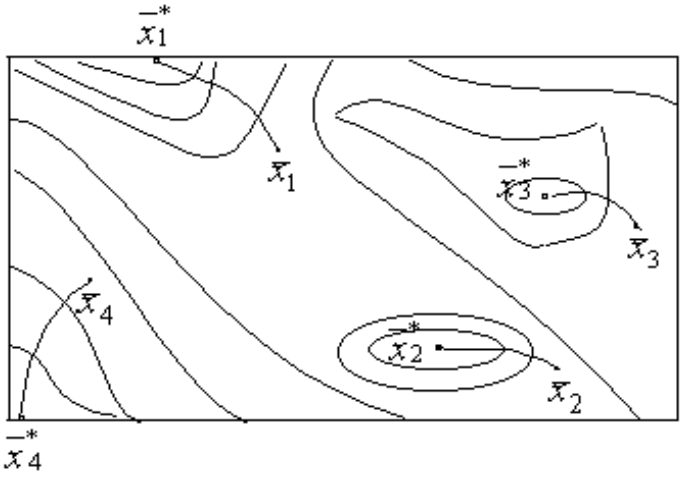


Рис.

Поиск прекращается, как только некоторое заданное число m раз не удается найти точку локального экстремума со значением функции, меньшим предыдущих.

Алгоритм 2. Пусть получена некоторая точка локального экстремума $\bar{x}_1^* \in D$. После этого переходим к *ненаправленному случайному* поиску до получения точки $\bar{x}_2$ такой, что $f(\bar{x}_2) < f(\bar{x}_1^*)$.

Из точки $\bar{x}_2$ с помощью детерминированного алгоритма или направленного случайного поиска получаем точку локального экстремума $\bar{x}_2^*$, в которой заведомо выполняется неравенство $f(\bar{x}_2^*) < f(\bar{x}_1^*)$.

Далее с помощью случайного поиска определяем новую точку $\bar{x}_3$, для которой справедливо неравенство $f(\bar{x}_3) < f(\bar{x}_2^*)$, и снова спуск в точку локального экстремума $\bar{x}_3^*$, и т.д.

Поиск прекращается, если при генерации некоторого предельного числа новых случайных точек не удается найти лучшей, чем предыдущий локальный экстремум, который тогда и принимается в качестве решения.

Алгоритм 3. Пусть $\bar{x}_1^0$ – некоторая исходная точка поиска в области D, из которой осуществляется спуск в точку локального экстремума $\bar{x}_1^*$ со значением $f(\bar{x}_1^*)$. Далее из точки $\bar{x}_1^*$ движемся либо в случайном направлении, либо в направлении $\bar{x}_1^* - \bar{x}_1^0$ до тех пор, пока функция снова не станет убывать (выходим из области притяжения $\bar{x}_1^*$).

Полученная точка $\bar{x}_2^0$ принимается за начало следующего спуска. В результате находим новый локальный экстремум $\bar{x}_2^*$ со значением функции $f(\bar{x}_2^*)$.

Если $f(\bar{x}_2^*) < f(\bar{x}_1^*)$, точка $\bar{x}_1^*$ забывается и ее место занимает точка $\bar{x}_2^*$. Если $f(\bar{x}_2^*) \geq f(\bar{x}_1^*)$, то возвращаемся в точку $\bar{x}_1^*$ и движемся из нее в новом случайном направлении.

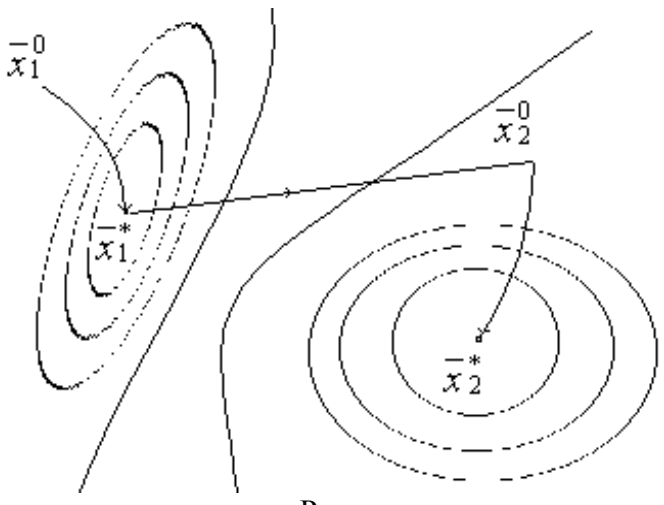


Рис.

Процесс прекращается, если не удается найти лучший локальный минимум после заданного числа попыток или не удается найти “случайного” направления, в котором функция снова начинает убывать.

Такой подход позволяет найти глобальный экстремум в случае многосвязных допустимых областей.

Алгоритм 4. В допустимой области D разбрасываем m случайных точек и выбираем из них наилучшую, то есть ту, в которой значение функции минимально. Из выбранной точки осуществляем локальный спуск. А далее вокруг траектории спуска образуем *запретную область*.

В оставшейся области случайным образом разбрасываем новую совокупность случайных точек и из лучшей точки осуществляем спуск в точку локального экстремума. Вокруг новой траектории также строим запретную область и т.д.

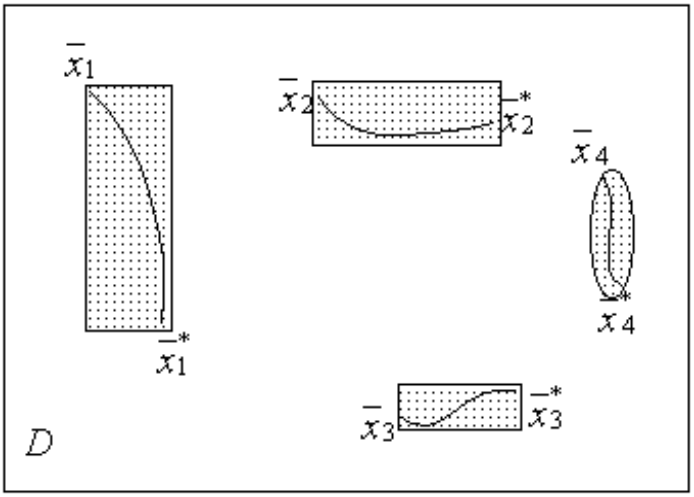


Рис.

Поиск прекращается, если в течение заданного числа попыток не удается найти лучшего локального экстремума.

Замечание: Комбинация случайного поиска с детерминированными методами применяется не только для решения многоэкстремальных задач. Часто к такой комбинации прибегают в ситуациях, когда детерминированные методы сталкиваются с теми или иными трудностями (застревают на дне узкого оврага, в седловой точке и т.п.). Шаг в случайном направлении порой позволяет преодолеть такую тупиковую ситуацию для детерминированного алгоритма.

Линейное программирование

Основные определения и теоремы

Определение 1. Задача, в которой требуется минимизировать (или максимизировать) линейную форму

$$\sum_{i=1}^{n} c_i x_i \to \min (\max)$$

при условии, что

$$\sum_{j=1}^{n} a_{ij} \cdot x_i \le b_j,\ j = \overline{1, m},$$

или

$$\sum_{i=1}^{n} a_{ij} \cdot x_i = b_j,\ j = \overline{(m+1), r},$$

и

$$x_i > 0,\ i = \overline{1, n},$$

называется задачей линейного программирования в произвольной форме записи.

Определение 2. Задача в матричной форме вида

$$\left.\begin{array}{l} \bar{c}^{T} \cdot \bar{x} \to \min (\max) \\ A\bar{x} \le \bar{b}, \\ \bar{x} \ge \bar{0}, \end{array}\right\} \qquad (1)$$

называется симметричной формой записи задачи линейного программирования.

Определение 3. Задача линейного программирования вида

$$\left.\begin{array}{l} \bar{c}^{T} \cdot \bar{x} \to \min (\max) \\ A\bar{x} = \bar{b}, \\ \bar{x} \ge \bar{0}, \end{array}\right\} \qquad (2)$$

называется канонической формой записи задачи линейного программирования.

Любую задачу линейного программирования можно привести к канонической форме. Например, если система ограничений задана в виде

$$A \cdot \bar{x} \le \bar{b},$$

то можно, введя дополнительные переменные, привести ее к виду

$$A\bar{x} + E\bar{y} = \bar{b},\ \bar{x} \ge 0,\ \bar{y} \ge 0,$$

где $\bar{y} = [x_{n+1}, \ldots, x_{n+m}]$. Если же ограничения в задаче заданы в виде

$$A \cdot \bar{x} \ge \bar{b},$$

то

$$A\bar{x} - E\bar{y} = \bar{b},\ \bar{x} \ge 0,\ \bar{y} \ge 0.$$

Рассмотрим задачу с ограничениями $A \cdot \bar{x} \le \bar{b}$. Эту систему ограничений можно представить в виде системы

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{n1} = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + x_{n2} = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n + x_{nm} = b_m \end{cases}.$$

Введем следующие обозначения:

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix}, \bar{A}_1 = \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ \ldots \\ a_{m,1} \end{bmatrix}, \ldots, \bar{A}_n = \begin{bmatrix} a_{1,n} \\ a_{2,n} \\ \ldots \\ a_{m,n} \end{bmatrix}, \bar{A}_{n+1} = \begin{bmatrix} 1 \\ 0 \\ \ldots \\ 0 \end{bmatrix}, \ldots, \bar{A}_{n+m} = \begin{bmatrix} 0 \\ 0 \\ \ldots \\ 1 \end{bmatrix},$$

$$\bar{A}_0 = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{bmatrix}.$$

Тогда задачу линейного программирования можно записать в виде:

$$\sum_{i=1}^{n} c_i x_i \to \min$$

$$\bar{x} \geq \bar{0}.$$

Векторы $\bar{A}_i$ называются векторами условий, а сама задача линейного программирования называется расширенной по отношению к исходной.

Пусть D и D_1 - допустимые множества решений исходной и расширенной задач линейного программирования соответственно. Тогда любой точке множества D_1 соответствует единственная точка множества D и наоборот. В общем случае, допустимое множество D исходной задачи есть проекция множества D_1 расширенной задачи на подпространство исходных переменных.

Определение 4. Набор чисел $\bar{x} = [x_1 x_2 \ldots x_n]$, удовлетворяющий ограничениям задачи линейного программирования, называется ее **планом.**

Определение 5. Решением задачи линейного программирования называется ее план, минимизирующий (или максимизирующий) линейную форму.

Введем понятие базисного решения. Из матрицы расширенной задачи $A_p = [\bar{A}_1 \bar{A}_2 \ldots \bar{A}_n]$ выберем m линейно независимых векторов-столбцов, которые обозначим как матрицу $B_{m\times m}$, а через $D_{m\times n}$ – обозначим матрицу из оставшихся столбцов. Тогда $A_p = [BD]$, и ограничения расширенной задачи линейного программирования можно записать в виде:

$$A_p \bar{x} = B\bar{x}_B + D\bar{x}_D. \qquad (3)$$

Очевидно, что столбцы матрицы B образуют базис m-мерного пространства. Поэтому вектор $\bar{A}_0$ и любой столбец матрицы D можно представить в виде линейной комбинации столбцов матрицы B.

Умножим (3) на B^{-1} слева

$$B^{-1}\bar{x}_B = B^{-1}\bar{A}_0 - B^{-1}D\bar{x}_D, \tag{4}$$

и найдем отсюда $\bar{x}_B$:

$$\bar{x}_B = B^{-1}\bar{A}_0 - B^{-1}D\bar{x}_D. \tag{5}$$

Придавая $\bar{x}_D$ различные значения, будем получать различные решения $\bar{x}_B$.

Если положить $\bar{x}_D = \bar{0}$, то

$$\bar{x}_B = B^{-1}\cdot\bar{A}_0. \tag{6}$$

Решение (6) называют ***базисным решением*** системы из m уравнений с $m+n$ неизвестными.

Если полученное решение содержит только положительные компоненты, то оно называется ***базисным допустимым***.

Особенность допустимых базисных решений состоит в том, что они являются крайними точками допустимого множества D_1 расширенной задачи.

Если среди компонент $\bar{x}_B$ нет нулевых, то базисное допустимое решение называется ***невырожденным***.

Определение 6. План $\bar{x}$ задачи линейного программирования будем называть опорным, если векторы условий $\bar{A}_i$ с положительными коэффициентами линейно независимы.

То есть, опорный план – это базисное допустимое решение расширенной системы, угловая точка многогранника решений.

Определение 7. Опорное решение называется **невырожденным**, если оно содержит m положительных компонент (по числу ограничений).

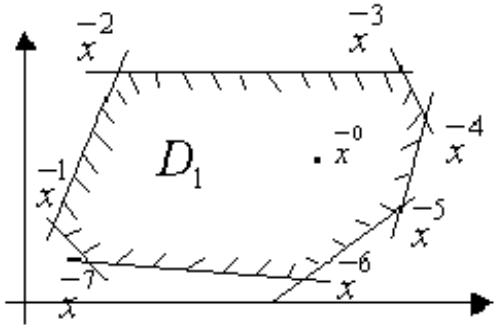
Невырожденный опорный план образуется пересечением n гиперплоскостей из образующих допустимую область. В случае вырожденности в угловой точке многогранника решений пересекается более чем n гиперплоскостей.

Теорема 1 (основная теорема линейного программирования).

1) Линейная форма $Z = \bar{C}^{T}\cdot\bar{X}$ достигает своего минимума в угловой точке многогранника решений.
2) Если она принимает минимальное решение более чем в одной угловой точке, то она достигает того же самого значения в любой точке, являющейся выпуклой комбинацией этих угловых точек.

Доказательство: Доказательство теоремы основано на следующей лемме.

Лемма. Если D - замкнутое, ограниченное, выпуклое множество, имеющее конечное число крайних (угловых) точек, то любая точка $\bar{x} \in D$ может быть представлена в виде выпуклой комбинации крайних точек D.



1) Пусть $\bar{x}^{0}$ - некоторая внутренняя точка. Многогранник ограниченный замкнутый, имеет конечное число угловых точек. D – допустимое множество.

Предположим, что точка $\bar{x}^0$ является оптимальной точкой. То есть, $z(\bar{x}^0) \le z(\bar{x}),\ \forall \bar{x} \in D$. Предположим, что точка $\bar{x}^0$ не является угловой. Тогда на основании леммы точку $\bar{x}^0$ можно выразить через угловые точки многогранника $\bar{x}^i$, т.е.

$$\bar{x}^0 = \sum_{i=1}^{p} \alpha_i \cdot \bar{x}^i,\ \forall \alpha_i \ge 0,\ \sum_{i=1}^{p} \alpha_i = 1.$$

Так как функция $z(\bar{x})$ линейна, то

$$z(\bar{x}^0) = \sum_{i=1}^{p} \alpha_i z(\bar{x}^i). \qquad (*)$$

Выберем среди точек $\bar{x}^i$ ту, в которой линейная форма $z(\bar{x})$ принимает наименьшее значение. Пусть это будет точка $\bar{x}^k$. Обозначим минимальное значение функции в угловой точке через z^*:

Подставим данное значение функции в линейную форму (*) вместо $z(\bar{x}^i)$ и получим:

Так как $\bar{x}^0$ - оптимальная точка, то получили противоречие: $z(\bar{x}^0) \ge z^*$ (!). Следовательно, $z(\bar{x}^0) = z(\bar{x}^k),\ \bar{x}^0 = \bar{x}^k$ – угловая точка.

2) Предположим, что линейная форма $z(\bar{x})$ принимает минимальное значение более чем в одной угловой точке, например, в угловых точках $\bar{x}^1, \bar{x}^2, \ldots \bar{x}^q$

. Тогда если $\bar{x}$ является выпуклой комбинацией этих точек, то есть

$$\bar{x} = \sum_{i=1}^{q} \alpha_i \cdot \bar{x}^i,\ \sum_{i=1}^{q} \alpha_i = 1 \text{ и } \forall i\ \alpha_i \ge 0,$$

то 

То есть, если минимальное значение достигается более чем в одной угловой точке, то того же самого значения линейная форма достигает в любой точке, являющейся выпуклой комбинацией этих угловых точек

Теорема 2. Если известно, что системы векторов условий $\bar{A}_1, \ldots \bar{A}_n$, $(m \le n)$ линейно независима и такова, что

где все $x_j > 0$, то точка $\bar{x} = [x_1 \ldots x_m 0 \ldots 0]$ является угловой точкой многогранника решений.

Теорема 3. Если вектор $\bar{x}$ является угловой точкой многогранника решений, то векторы условий, соответствующие положительным компонентам вектора $\bar{x}$, являются линейно независимыми.

Следствия:

1) Угловая точка многогранника решений имеет не более m положительных компонент вектора $\bar{x}$.

2) Каждой угловой точке многогранника решений соответствует m линейно независимых векторов из данной системы: $\bar{A}_1,\ldots,\bar{A}_n$.

Симплекс метод

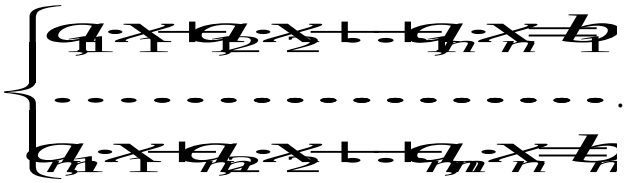
Его называет еще **методом последовательного улучшения плана**. Метод предназначен для решения общей задачи линейного программирования.

Введение в симплекс-метод

Пусть имеем следующую задачу:

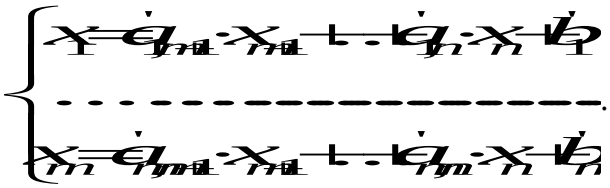
(1)

с системой ограничений следующего вида:



(2)

Разрешим эту систему относительно переменных $x_1,\ldots,x_m$:



(3)

Векторы условий, соответствующие $x_1,\ldots,x_m$, образуют базис. Переменные $x_1,\ldots,x_m$ назовем базисными переменными. Остальные переменные задачи – небазисные.

Целевую функцию можно выразить через небазисные переменные:

Если приравнять небазисные переменные нулю

то соответствующие базисные переменные примут значения

Вектор $\bar{x}$ с такими компонентами представляет собой угловую точку многогранника решений (допустимую) при условии, что $b_i' \geq 0$ (опорный план).

Теперь необходимо перейти к другой угловой точке с меньшим значением целевой функции. Для этого следует выбрать некоторые небазисную и базисную переменные так, чтобы после того, как мы "поменяем их местами", значение целевой функции уменьшится. Такой направленный перебор в конце концов приведет нас к решению задачи.

Пример 1: Пусть

$$Q(x) = x_4 + x_5 \to \min$$

$$\left.\begin{aligned} x_1 + x_4 - 2x_5 &= 1 \\ x_2 - 2x_4 + x_5 &= 2 \\ x_3 + 3x_4 + x_5 &= 3 \end{aligned}\right\}.$$

Выберем в качестве базисных следующие переменные $\{x_1, x_2, x_3\}$ и разрешим систему относительно этих переменных. Система ограничений примет следующий вид:

$$\left.\begin{aligned} x_1 &= 1 - x_4 + 2x_5 \\ x_2 &= 2 + 2x_4 - x_5 \\ x_3 &= 3 - 3x_4 - x_5 \end{aligned}\right\}.$$

Переменные $\{x_4, x_5\}$ являются небазисными. Если взять $x_4 = 0$ и $x_5 = 0$, то получим угловую точку (опорный план)

$$\bar{x}^1 = [1\ 2\ 3\ 0\ 0],$$

которому соответствует $Q(\bar{x}^1) = 0$.

Значение целевой функции можно уменьшить за счет увеличения x_5. При увеличении x_5 величина x_1 также увеличивается, а x_2 и x_3 – уменьшаются. Причем величина x_2 может стать отрицательной раньше. Поэтому, вводя в базис переменную x_5, одновременно x_2 исключаем из базиса. В результате после очевидных преобразований получим следующие выражения для новой системы базисных переменных и целевой функции:

$$\left.\begin{aligned} x_5 &= 2 - x_2 + 2x_4 \\ x_1 &= 5 - 2x_2 + 3x_4 \\ x_3 &= 1 + x_2 - 5x_4 \end{aligned}\right\}$$

$$Q(x) = 2x_4 + x_2 \to \min.$$

Соответствующий опорный план $\bar{x}^2 = [5\ 0\ 1\ 0\ 2]$ и $Q(\bar{x}^2) = 2$.

Целевую функцию можно уменьшить за счет увеличения x_4. Увеличение x_4 приводит к уменьшению только x_3. Поэтому вводим в базис переменную x_4, а x_3 исключаем из базиса. В результате получим следующие выражения для новой системы базисных переменных и целевой функции:

$$\left.\begin{aligned} x_4 &= \frac{1}{5} + \frac{1}{2}x_2 - \frac{1}{5}x_3 \\ x_1 &= \frac{28}{5} - \frac{7}{5}x_2 - \frac{3}{5}x_3 \\ x_5 &= \frac{12}{5} - \frac{3}{5}x_2 - \frac{2}{3}x_3 \end{aligned}\right\}$$

$$\left.\begin{aligned} x_4 &= \frac{1}{5} + \frac{1}{2}x_2 - \frac{1}{5}x_3 \\ x_1 &= \frac{28}{5} - \frac{7}{5}x_2 - \frac{3}{5}x_3 \\ x_5 &= \frac{12}{5} - \frac{3}{5}x_2 - \frac{2}{3}x_3 \end{aligned}\right\}$$

Соответствующий опорный план $\bar{x}^3 = \left[\frac{28}{5}\ 0\ 0\ \frac{1}{5}\ \frac{12}{5}\right]$ и значение целевой функции $Q(\bar{x}^3) = -\frac{1}{5}$. Так как все коэффициенты при небазисных переменных в целевой функции неотрицательны, то нельзя уменьшить целевую функцию за счет увеличения x_2 или x_3. Следовательно, полученный план $\bar{x}^3$ является оптимальным.

Пример 2: Пусть имеем задачу

$$Q(\bar{x}) = x_1 - x_2 \to \min$$
$$\left.\begin{aligned} x_3 &= 1 + x_1 - x_2 \\ x_4 &= 2 - x_1 + 2x_2 \\ \bar{x} &\geq 0 \end{aligned}\right\}.$$

Переменные $\{x_3, x_4\}$ – базисные, а $\{x_1, x_2\}$ – небазисные переменные. Опорный план $\bar{x}^0 = [0\ 0\ 1\ 2]$, $Q(\bar{x}^0) = 0$.

Теперь вводим в базис переменную x_1, а x_4 исключаем из базиса. В результате получим следующие выражения для базисных переменных и целевой функции:

$$\left.\begin{aligned} x_1 &= 2 + 2x_2 - x_4 \\ x_3 &= 3 + x_2 - x_4 \end{aligned}\right\}$$
$$Q(\bar{x}) = -2 - x_2 + x_4.$$

Опорный план $\bar{x}^1 = [2\ 0\ 3\ 0]$, значение целевой функции $Q(\bar{x}^1) = -2$.

Теперь можно заметить, что при увеличении x_2 значения переменных x_1 и x_3 также возрастают. То есть, при $x_2 \to \infty$ в допустимой области $Q(\bar{x}) \to -\infty$ (задача не имеет решения).

Замечание: В процессе поиска допустимого плана может быть выявлена противоречивость системы ограничений.

Алгоритм симплекс метода

Формализованный алгоритм симплекс метода состоит из двух основных этапов: 1) построение опорного плана; 2) построение оптимального плана.

Проиллюстрируем алгоритм на рассмотренном ранее примере:

$$Q(\bar{x}) = x_4 + x_5 \to \min$$
$$\left.\begin{aligned} x_1 + x_4 - 2x_5 &= 1 \\ x_2 - 2x_4 + x_5 &= 2 \\ x_3 + 3x_4 + x_5 &= 3 \end{aligned}\right\}.$$
$$\bar{x} \geq \bar{0}.$$

В случае базисных переменных $\{x_1, x_2, x_3\}$ начальная симплексная таблица для данного примера будет выглядеть следующим образом:

	$-x_4$	$-x_5$	1
$x_1 =$	1	–2	1
$x_2 =$	–2	1	2
$x_3 =$	3	1	3
$Q(\bar{x}) =$	–1	1	0

Она уже соответствует опорному плану $\bar{x}^1 = [1\ 2\ 3\ 0\ 0]$ (столбец свободных членов).

Построение оптимального плана. Для того чтобы опорный план был оптимален при минимизации целевой функции необходимо, чтобы коэффициенты в строке целевой функции были неположительными (в случае максимизации – неотрицательными). То есть, при поиске минимума мы должны освободиться от положительных коэффициентов в строке $Q(\bar{x})$.

Выбор разрешающего элемента. Если при поиске минимума в строке целевой функции есть коэффициенты больше нуля, то выбираем столбец с положительным коэффициентом в строке целевой функции в качестве разрешающего. Пусть это столбец с номером l.

Для выбора разрешающей строки (разрешающего элемента) среди положительных коэффициентов разрешающего столбца выбираем тот (ту строку), для которого отношение коэффициента в столбце свободных членов к коэффициенту в разрешающем столбце минимально:

$$\frac{b_r}{a_{rl}} = \min_i \left\{ \frac{b_i}{a_{il}} \middle| a_{il} > 0 \right\}.$$

a_{rl} – разрешающий (направляющий) элемент, строка r – разрешающая.

Для перехода к следующей симплексной таблице (следующему опорному плану с меньшим значением целевой функции) делается шаг модифицированного жорданова исключения с разрешающим элементом a_{rl}.

Если в разрешающем столбце нет положительных коэффициентов, то целевая функция неограничена снизу (при максимизации – неограничена сверху).

Шаг модифицированного жорданова исключения над симплексной таблицей:

1) На месте разрешающего элемента ставится 1 и делится на разрешающий элемент.
2) Остальные элементы разрешающего столбца меняют знак на противоположный и делятся на разрешающий элемент.

3) Остальные элементы разрешающей строки делятся на разрешающий элемент.
4) Все остальные элементы симплексной таблицы вычисляются по следующей формуле:

	$-x_4$	$-x_5$	1
x_1	1	–2	1
~~x_2~~	~~–2~~	~~1~~	2
x_3	3	1	3
$Q(\bar{x})$	-1	1	0

Разрешающий элемент, соответствующий замене базисной переменной x_2 на небазисную переменную x_5.

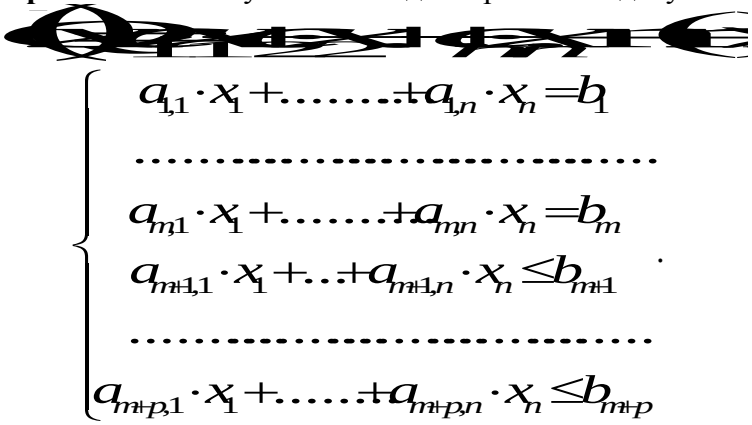
	$-x_4$	$-x_2$	1
x_1	–3	2	5
x_5	–2	1	2
~~x_3~~	~~5~~	~~–1~~	1
$Q(\bar{x})$	1	–1	–2

Разрешающий элемент, соответствующий замене базисной переменной x_3 на небазисную переменную x_4.

	$-x_3$	$-x_2$	1
x_1	3/5	7/5	28/5
x_5	2/5	3/5	12/5
x_4	1/5	–1/5	1/5
$Q(\bar{x})$	–1/5	–4/5	–11/5

Все коэффициенты в строке целевой функции отрицательны, т.е. мы нашли оптимальное решение.

Построение опорного плана. Пусть необходимо решить задачу:



Введем дополнительные переменные, чтобы преобразовать ограничения-неравенства к равенствам. В ограничениях-равенствах дополнительные переменные должны быть нулевыми. Тогда система ограничений принимает вид:

$$\begin{cases} 0 = b_1 - a_{1,1} \cdot x_1 - \ldots\ldots - a_{1,n} \cdot x_n \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0 = b_m - a_{m,1} \cdot x_1 - \ldots\ldots - a_{m,n} \cdot x_n \\ x_{n+1} = b_{m+1} - a_{m+1,1} \cdot x_1 - \ldots - a_{m+1,n} \cdot x_n \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_{n+p} = b_{m+p} - a_{m+p,1} \cdot x_1 - \ldots - a_{m+p,n} \cdot x_n \end{cases},$$

где $x_{n+i} \geq 0 \quad \forall i = \overline{1, p}$.

В качестве базисных переменных будем брать систему дополнительно введенных переменных. Тогда симплексная таблица для преобразованной задачи будет иметь следующий вид:

	$-x_1$	$-x_2$	….	$-x_S$	….	$-x_n$	1
0	$a_{1,1}$	$a_{1,2}$	….	$a_{1,S}$	….	$a_{1,n}$	b_1
….	….	….	….	….	….	….	….
0	$a_{m,1}$	$a_{m,2}$	….	$a_{m,S}$	….	$a_{m,n}$	b_m
x_{m+1}	$a_{m+1,1}$	$a_{m+1,2}$	….	$a_{m+1,s}$	….	$a_{m+1,n}$	b_{m+1}
….	….	….	….	….	….	….	…
x_{m+p}	$a_{m+p,1}$	$a_{m+p,2}$	….	$a_{m+p,S}$	….	$a_{m+p,n}$	b_{m+p}
$Q(\bar{x})$	$-c_1$	$-c_2$	….	$-c_S$	….	$-c_n$	0

Правила выбора разрешающего элемента при поиске опорного плана:

1. При условии отсутствия “0-строк” (ограничений-равенств) и “свободных” переменных (т.е. переменных, на которые не наложено требование неотрицательности).
   - Если в столбце свободных членов симплексной таблицы нет отрицательных элементов, то опорный план найден.
   - Есть отрицательные элементы в столбце свободных членов, например $b_i < 0$. В такой строке ищем отрицательный коэффициент a_{il}, и этим самым определяем разрешающий столбец l. Если не найдем отрицательный a_{il}, *то система ограничений несовместна* (противоречива).
   - В качестве разрешающей выбираем строку, которой соответствует минимальное отношение:

$$\frac{b_r}{a_{rl}} = \min_i \left\{ \frac{b_i}{a_{il}} \middle| \frac{b_i}{a_{il}} > 0 \right\},$$

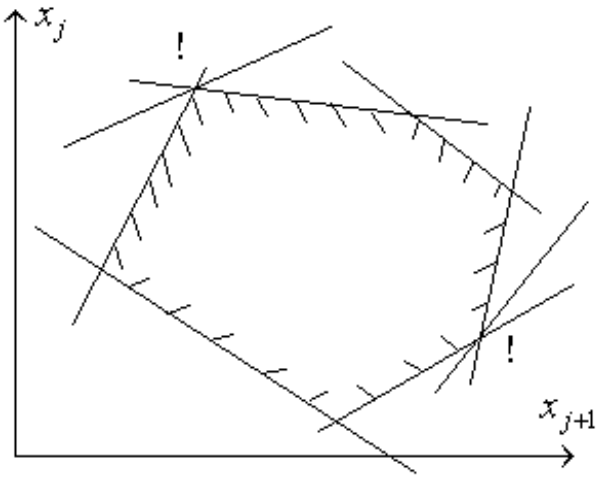
   где r - номер разрешающей строки. Таким образом, a_{rl} - разрешающий элемент.
   - После того, как разрешающий элемент найден, делаем шаг модифицированного жорданова исключения с направляющим элементом a_{rl} и переходим к следующей симплексной таблице.

2. В случае присутствия ограничений-равенств и “свободных” переменных поступают следующим образом.
   (1) Выбирают разрешающий элемент в “0-строке” и делают шаг модифицированного жорданова исключения, после чего вычеркивают этот разрешающий столбец. Данную последовательность действий продолжают до тех пор, пока в симплексной таблице остается хотя бы одна “0-строка” (при этом таблица сокращается).
   (2) Если же присутствуют и свободные переменные, то необходимо данные переменные сделать базисными. И после того, как свободная переменная станет базисной, далее в процессе определения разрешающего элемента при поиске опорного и оптимального планов данная строка не учитывается (но преобразуется).

Вырожденность в задачах линейного программирования

Рассматривая симплекс-метод, мы предполагали, что задача линейного программирования является невырожденной, т.е. каждый опорный план содержит ровно m положительных компонент, где m – число ограничений в задаче. В вырожденном опорном плане число положительных компонент оказывается меньше числа ограничений: некоторые базисные переменные, соответствующие данному опорному плану, принимают нулевые значения.

Используя геометрическую интерпретацию для простейшего случая, когда $n-m=2$ (число небазисных переменных равно 2), легко отличить вырожденную задачу от невырожденной. В вырожденной задаче в одной вершине многогранника условий пересекается более двух прямых, описываемых уравнениями вида $x_i = 0$. Это значит, что одна или несколько сторон многоугольника условий стягиваются в точку.



Аналогично при $n-m=3$ в вырожденной задаче в одной вершине пересекается более 3-х плоскостей $x_i = 0$.

В предположении о невырожденности задачи находилось только одно значение, соответствующее минимуму $\Theta = \min_i \left\{ \frac{b_i}{a_{il}} \middle| \frac{b_i}{a_{il}} > 0 \right\}$, по которому определялся индекс выводимого из базиса вектора условий (выводимой из числа базисных переменной).

В вырожденной задаче $\min_i \left\{ \frac{b_i}{a_{il}} \middle| \frac{b_i}{a_{il}} > 0 \right\}$ может достигаться на нескольких индексах сразу (для нескольких строк). В этом случае в находимом опорном плане несколько базисных переменных будут нулевыми.

Если задача линейного программирования оказывается вырожденной, то при плохом выборе вектора условий, выводимого из базиса, может возникнуть бесконечное движение по базисам одного и того же опорного плана. Так называемое явление зацик-

ливания. Хотя в практических задачах линейного программирования зацикливание явление крайне редкое, возможность его не исключена.

Один из приемов борьбы с вырожденностью состоит в преобразовании задачи путем “незначительного” изменения вектора правых частей системы ограничений на величины ε_i, таким образом, чтобы задача стала невырожденной и, в то же время, чтобы это изменение не повлияло реально на оптимальный план задачи.

Чаще реализуемые алгоритмы включают в себя некоторые простые правила, снижающие вероятность возникновения зацикливания или его преодоления.

Пусть переменную x_j необходимо сделать базисной. Рассмотрим множество индексов E_0, состоящее из тех i, для которых достигается $\Theta = \min_i \left\{ \frac{b_i}{q_i} \middle| \frac{b_i}{q_i} > 0 \right\}$. Множество индексов i, для которых выполняется данное условие, обозначим через E_0. Если E_0 состоит из одного элемента, то из базиса исключается вектор условий A_i (переменная x_i делается небазисной).

Если E_0 состоит более чем из одного элемента, то составляется множество E_1, которое состоит из $i \in E_0$, на которых достигается $\Theta_1 = \min_{i \in E_0} \left\{ \frac{q_1}{q_l} \right\}$. Если E_1 состоит из одного индекса k, то из базиса выводится переменная x_k. В противном случае составляется множество E_2 и т.д.

Практически правилом надо пользоваться, если зацикливание уже обнаружено.

Литература

1) Карманов В.Г. Математическое программирование. –М.: Наука, 1975. –272 с.
2) Химмельблау Д. Прикладное нелинейное программирование. – М.: Мир, 1975. – 534 с.
3) Фиакко А., Мак-Кормик Г. Нелинейное программирование: Методы последовательной безусловной оптимизации. – М.: Мир, 1972. – 240 с.
4) Кюнци Г.П., Крелле В. Нелинейное программирование. – М.: Сов. радио, 1965. – 303 с.
5) Васильев Ф.П. Численные методы решения экстремальных задач. – М.: Наука, 1980. – 520 с.
6) Пшеничный Б.Н., Данилин Ю.М. Численные методы в экстремальных задачах. – М: Наука, 1975. – 319 с.
7) Сеа Ж. Оптимизация: Теория и алгоритмы. – М.: Мир, 1973. – 244 с.
8) Полак Э. Численные методы оптимизации: Единый подход. – М.: Мир, 1974. – 376 с.
9) Аоки М. Введение в методы оптимизации. – М: Наука, 1977. – 344 с.
10) Зангвилл У.И. Нелинейное программирование. – М.: Советское радио, 1973. – 312 с.
11) Коробкин А.Д., Лемешко Б.Ю., Цой Е.Б. Математические методы оптимизации. Учебное пособие. - Новосибирск, 1977.
12) Лемешко Б.Ю. Исследование операций. Методические указания. - Новосибирск, 1995.
13) Зуховицкий С.И., Авдеева Л.И. Линейное и выпуклое программирование. - М.: Наука, 1967.
14) Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. - М.: Наука, 1969. – 368 с.
15) Гольштейн Е.Г., Юдин Д.Б. Задачи линейного программирования транспортного типа. - М.: Наука, 1969. – 382 с.
16) Юдин Д.Б., Гольштейн Е.Г. Линейное программирование (теория и конечные методы). - М.: Наука, 1963.
17) Хедли Дж. Нелинейное и динамическое программирование. - М.: Мир, 1968.
18) Беллман Р. Динамическое программирование. - М.: Мир, 1963.
19) Вагнер Г. Основы исследования операций.-М.: Мир, Т.1, 1972.; т. 2, 1973.; т. 3, 1973.
20) Акоф Р., Сасиени М. Основы исследования операций. -М.: Мир, 1971. -533 с.
21) Вентцель Е.С. Исследование операций. – М.: Советское радио, 1972. – 552 с.

22) Саати Т.Л. Математические методы исследования операций. - М.: Мир, 1973.
23) Зайченко Ю.П. Исследование операций. – Киев: Вища школа, 1975. – 320 с.
24) Давыдов Э.Г. Исследование операций. -М.: Высшая школа, 1990.
25) Подиновский В.В., Ногин В.Д. Парето-оптимальные решения многокритериальных задач. - М.: Наука, 1982.
26) Эльсгольц Л.Э. Дифференциальные уравнения и вариационное исчисление. М.: Наука, 1965. – 424 с.
27) Лемешко Б.Ю. Курс лекций по методам оптимизации (электронный вариант http: //www.ami.nstu.ru/~headrd/).