Прудников А.М.

Текст лекций по курсу

Методы оптимизации

Версия: 0.1

Москва, 2012

Оглавление

Глава 1

Понятие математической оптимизации

1.1 Исследование операций и место методов оптимизации в данной дисциплине

Исследование операций – математическая дисциплина, занимающаяся построением, разработкой и применением математических моделей принятия оптимальных решений во всех областях человеческой деятельности.

Операция – это всякое мероприятие (система действий), объединенное единым замыслом и направленное к достижению какой-то цели.

Пусть необходимо выполнить какое-либо мероприятие для достижения определенной цели (**операцию**). Обычно существует свобода выбора в том, как это мероприятие организовать (например, выбор техники, распределение ресурсов и т.д.). «**Решение**» – это какой-то выбор из ряда допустимых возможностей.

Краеугольным камнем исследования операций является **математическое моделирование**. Данные, полученные при исследовании математических моделей, являются основой для принятия решений. Но общих способов построения математических моделей и методов их решения не существует. В каждом конкретном случае модель выбирается исходя из вида операции, ее целевой направленности, с учетом задачи исследования.

Наиболее известными методами исследования операций (являющимися, зачастую, самостоятельными математическими дисциплинами), являются:

- Математическое программирование – теория и методы решения задач о нахождении экстремумов функций на множествах векторного пространства, определяемых линейными и нелинейными ограничениями (равенствами и неравенствами). (Присутствие в названии термина «программирование» объясняется тем, что первые исследования оптимизационных задач были в сфере экономики, а в английском языке слово «programming» означает планирование, составление планов или программ).
- Сетевые модели – решение оптимизационных задач с использованием графов.
- Марковские процессы – метод решения стохастических задач, где процесс принятия

решений можно представить конечным числом состояний.

- Теория игр – методы изучения оптимальных стратегий в играх. Теория игр помогает выбрать лучшие стратегии с учётом представлений о других участниках, их ресурсах и их возможных поступках.

- Теория массового обслуживания (теория очередей) – раздел теории вероятностей, целью исследований которого является рациональный выбор структуры системы обслуживания и процесса обслуживания на основе изучения потоков требований на обслуживание, длительности ожидания и длины очередей.

- Имитационное моделирование – метод исследования, при котором изучаемая система заменяется компьютерной моделью, с достаточной точностью описывающей реальную систему, и с ней проводятся эксперименты с целью получения информации об этой системе. Экспериментирование с моделью называют имитацией.

В данном курсе будет изучаться математическое программирование, методы которого и являются так называемыми **методами оптимизации**.

1.2 Понятие оптимизации

Оптимизация в самом широком смысле - это выбор наилучшего варианта из множества возможных. Рассмотрим некоторые классические примеры оптимизационных задач.

1.2.1 Задача коммивояжера

Дано некоторое количество городов и расстояния между ними. Коммивояжер должен посетить каждый город и вернуться к месту отправления. Какой маршрут он должен выбрать? Зададим расстояния между городами (допустим, их пять) в виде следующей таблицы:

	1	2	3	4	5
1	–	1	7	2	8
2	2	–	10	3	1
3	7	10	–	2	6
4	2	3	2	–	4
5	8	1	6	4	–

Проложим несколько маршрутов и посчитаем расстояние для них:

- Маршрут $5 \to 1 \to 2 \to 3 \to 4 \to 5$, расстояние $8 + 1 + 10 + 2 + 4 = 25$

- Маршрут $5 \to 2 \to 1 \to 4 \to 3 \to 5$, расстояние $1 + 1 + 2 + 2 + 6 = 12$

Каким же образом (исключая полный перебор) следует выбрать кратчайший маршрут?

1.2.2 Задача размещения производства

Пусть в некотором регионе имеется ряд потребителей некоторой продукции. Нужно определить, как разместить в этом регионе заводы по производству данной продукции.

Существует два крайних решения:

1. Можно разместить только один завод; в этом случае производственные затраты будут минимальны, но станут максимальными затраты на доставку продукции потребителям (транспортные затраты).

2. Можно разместить большое количество заводов (рядом с каждым потребителем); в этом случае транспортные затраты будут минимальны, но станут максимальными производственные затраты.

Очевидно, что оптимальное решение заключается в минимизации суммарных затрат $C_{\text{общ}} = C_{\text{произв}} + C_{\text{трансп}} \to \min$.

1.2.3 Построение математической модели

Как было сказано выше, для решения любых задач исследования операций (и, соответственно, задач математического программирования) необходимо формализовать решаемую задачу, построив ее математическую модель. В самых общих чертах процесс построения математической модели можно представить следующим образом:

1. Определение неизвестных параметров (элементов решения).

2. Выражение условий задачи через введенные на первом шаге неизвестные.

3. Выбор критерия оптимальности.

Попробуем выполнить описанные выше шаги для построения математической модели задачи размещения производства (см. раздел 1.2.2).

На шаге **определения неизвестных параметров** введем переменные x_{ij} - объем перевозимой продукции с i-го завода j-му потребителю; здесь $i = \overline{1,m}$ - количество заводов, $j = \overline{1,n}$ - количество потребителей.

Теперь **выразим условия задачи** через эти переменные:

1. Очевидно, что количество перевозимой продукции не может быть отрицательно: $x_{ij} \geq 0$.

2. Если обозначить известный нам объем заказов (потребления) продукции j-м потребителем через b_j, то на объем перевозок можно наложить ограничение $\sum\limits_{i=1}^{m} x_{ij} = b_j$ (не следует везти к потребителю продукции больше, чем он заказывает).

3. Если обозначить известные нам удельные транспортные затраты на перевозку изделий от i-го завода j-му потребителю через c_{ij}, то можно составить уравнение для транспортных затрат: $C_{\text{трансп}} = \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} c_{ij} x_{ij}$.

4. Если обозначить известную нам стоимость производства на i-ом заводе (которая, вообще говоря, зависит от объемов производства) через $f_i(x_i)$, то можно составить уравнение для производственных затрат: $C_{\text{произв}} = \sum\limits_{i=1}^{m} f_i(\sum\limits_{j=1}^{n} x_{ij})$.

Осталось **выбрать критерий оптимальности**, однако это мы уже сделали в п. 1.2.2: суммарные затраты должны быть минимальны.

Таким образом, в результате формализации задачи мы построили ее математическую оптимизационную модель следующего вида:

$$\min \left\{ \sum_{i=1}^{m} f_i(\sum_{j=1}^{n} x_{ij}) + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \right\} \tag{1.1}$$

при ограничениях

$$\begin{cases} \sum\limits_{i=1}^{m} x_{ij} = b_j \\ x_{ij} \geq 0 \end{cases} \quad i = \overline{1,m}, j = \overline{1,n} \tag{1.2}$$

Опираясь на полученный результат, можно попробовать записать постановку оптимизационной задачи в общем виде.

1.3 Общий вид однокритериальной оптимизационной задачи

В общем виде задача математического программирования ставится следующим образом: найти максимум (минимум) функции

$$f(x_1, x_2, \ldots, x_n) = f(\bar{x}) \tag{1.3}$$

при ограничениях

$$\begin{cases} g_i(x_1, x_2, \ldots, x_n) \leq b_i, i = \overline{1,m} \\ x_j \geq 0, j = \overline{1,n} \end{cases} \tag{1.4}$$

Здесь:

- $f(\bar{x})$ - целевая функция;
- система неравенств и условия неотрицательности переменных (1.4) - система ограничений.

Всякое решение задачи с учетом системы ограничений называется **допустимым решением**. Допустимое решение, максимизирующее (минимизирующее) целевую функцию, называется **оптимальным решением**. Таким образом, задача математического программирования заключается в нахождении оптимального решения, которое по определению обеспечивает максимальное (минимальное) значение целевой функции с учетом заданных ограничений.

1.4 Особенности задачи математического программирования

1. Если требуется найти минимум $f(\bar{x})$, то это эквивалентно поиску максимума $-f(\bar{x})$.

2. В любом случае можно добиться условия неотрицательности переменных, то есть, если задано ограничение $x_j \geq x_{j\_min}$, то можно сделать замену переменных $x_j^{'} = x_j - x_{j\_min} \geq 0$.

3. Если заданы ограничения вида $g_i(\bar{x}) \geq b_i$, то простой заменой знака приходим к первоначальной форме $-g_i(\bar{x}) \leq -b_i$.

4. Функция $f(\bar{x})$ может иметь несколько экстремумов, а именно локальные экстремумы и глобальный экстремум. Функция $f(\bar{x})$, определенная на области D, достигает на ней глобального максимума $\bar{x}^* \in D$, если неравенство $f(\bar{x}) \leq f(\bar{x}^*)$ справедливо для любой точки $\bar{x} \in D$. Функция $f(\bar{x})$, определенная на области D, достигает на ней локального максимума $\bar{x}^* \in D$, если неравенство $f(\bar{x}) \leq f(\bar{x}^*)$ справедливо для точек из некоторой окрестности $\bar{x}^*$.

5. В математическом анализе для нахождения экстремумов функций используются производные (это классические методы оптимизации). Такие методы применяют лишь для сравнительно простых задач из-за следующих недостатков:

   - для использования таких методов нужно, чтобы функции $f(\bar{x})$ и $g_i(\bar{x})$ были непрерывны и имели частные производные по крайней мере до 2-го порядка;
   - с помощью классических методов можно найти экстремум только внутри области; если оптимальная точка находится на границе области, то эти методы бессильны;
   - на переменные x_j не должны быть наложены условия целочисленности.

1.5 Классификация задач математического программирования

В зависимости от вида функций, входящих в критерий оптимальности и систему ограничений, а также допустимой области изменения переменных, задачи математического программирования разделяются на следующие классы:

1. Линейное программирование - целевая функция и ограничения являются линейными. Область допустимых значений - многогранник, а оптимальное решение находится в одной из его вершин.

2. Нелинейное программирование - или целевая функция, или какое-либо ограничение содержит нелинейную зависимость.

3. Дискретное программирование - переменные могут принимать только целочисленные значения.

1.6 Классификация методов решения задач оптимизации

Особенность задач оптимизации состоит в том, что вычисление значений целевой функции и значений ограничивающих функций может требовать больших затрат времени. В связи с этим возникает проблема решения задач оптимизации при наименьшем числе испытаний. **Испытанием** называется операция однократного вычисления функций $f(\bar{x})$ и $g_i(\bar{x})$ (и, в некоторых случаях, их производных) в некоторой точке $\bar{x}$. Далее будем говорить, что задача оптимизации решается с помощью **поискового метода оптимизации**, если используется следующая процедура поиска оптимального решения $\bar{x}^*$:

- по очереди при $r = 0, 1, 2, \ldots, N-1$ производятся испытания в точках

$$\bar{x}^{r+1} = \Psi_{r+1}\left(\bar{x}^0, f(\bar{x}^0), g_i(\bar{x}^0), \ldots, \bar{x}^r, f(\bar{x}^r), g_i(\bar{x}^r)\right) \tag{1.5}$$

- в качестве решения задачи берется точка $\bar{x}^*$, которая находится из условия $f(\bar{x}^*) = \min\limits_{r\in[0;N]} f(\bar{x}^r)$.

Здесь:

- r - текущий номер испытания;
- N - число испытаний;
- $\bar{x}^0$ - начальное приближение;
- Ψ_r - алгоритм поисковой оптимизации на r-ом шаге.

В общем случае **алгоритмом поисковой оптимизации** называется способ выбора начального приближения $\bar{x}^0$ и конкретная совокупность функций $\{\Psi_r\}$. Таким образом, понятие алгоритма является более частным по сравнению с понятием метода (одному и тому же методу могут соответствовать разные алгоритмы).

Теперь проведем классификацию методов решения с учетом введенным понятий.

1. **Классификация по наличию или отсутствию системы ограничений**. Если в задаче отсутствует система ограничений, то она решается методами **безусловной** оптимизации; в противном случае - методами **условной** оптимизации.

2. **Классификация по размерности вектора** $\bar{x}$. Если $\bar{x}$ на самом деле скаляр, то применяются **одномерные** методы оптимизации; в противном случае - **многомерные**.

3. **Классификация по характеру искомого решения**. Если метод поиска гарантирует отыскание только локального экстремума, то это метод **локальной** оптимизации. Если делается попытка отыскать глобальный экстремум, то это метод **глобальной** оптимизации. Следует отметить, что удовлетворительных с точки зрения вычислительной эффективности методов глобальной оптимизации не существует.

4. **Классификация по характеру функций** Ψ_r. Если функции Ψ_r являются детерминированными, то метод оптимизации называется **детерминированным**. Если же функции Ψ_r содержат случайные параметры, то метод оптимизации называется **стохастическим**.

5. **Классификация по способу выбора точек** $\bar{x}^r$. Если все точки $\bar{x}^r$ назначаются заранее (до проведения испытаний), то метод оптимизации называется **пассивным**. Если же очередная точка $\bar{x}^{r+1}$ определяется на основе всей или части информации об испытаниях в точках $\bar{x}^0, \ldots, \bar{x}^r$, то метод называется **последовательным**.

6. **Классификация по количеству предыдущих учитываемых шагов**. Если в последовательном методе при определении точки $\bar{x}^{r+1}$ учитывается информация только о предыдущем испытании, то метод называется **одношаговым**. Если же используется информация о $s > 1$ предыдущих испытаниях, то метод называется многошаговым (конкретнее, s-шаговым).

7. **Классификация по виду функций** Ψ_r. Если функция Ψ_r при всех N испытаниях одинакова, то метод называется **итерационным**. Если же функции Ψ_r меняются от испытания к испытанию, то метод является **неитерационным**.

8. **Классификация по порядку используемых производных**. Если при вычислении значений функций Ψ_r производные не используются, то метод называется **прямым** (или **нулевого порядка**). Если же используются производные k-го порядка, то метод называется методом k**-го порядка** (методы 1-го порядка также называются **градиентными**).

1.7 Условия окончания поиска

Выбор условия (критерия) окончания поиска является еще одной важной проблемой при решении оптимизационных задач. Наиболее широко используемыми являются следующие критерии:

- $\|\bar{x}^{r+1} - \bar{x}^r\| \leq \epsilon_x$, где ϵ_x - требуемая точность решения по $\bar{x}$, $\|\cdot\|$ - некоторая векторная норма (например, евклидова);

- $|f(\bar{x}^{r+1}) - f(\bar{x}^r)| \leq \epsilon_f$, где ϵ_f - требуемая точность решения по f.

Глава 2

Нелинейное программирование

2.1 Безусловная оптимизация функций одной переменной

Оптимизация функций одной переменной является, как правило, необходимым элементом методов оптимизации функций многих переменных. На первый взгляд кажется, что эта задача достаточно проста и решается с помощью дифференцирования (классический метод оптимизации). Однако для широкого класса функций это не так, поскольку задача решения уравнения $f'(x) = 0$ может оказаться весьма сложной (или даже невозможной, если $f(x)$ не дифференцируема).

Как уже было упомянуто выше, существование локальных экстремумов функции почти всегда затрудняет поиск глобального экстремума. Поэтому многие методы оптимизации применимы только тогда, когда любой локальный экстремум является одновременно и глобальным; это дает гарантию сходимости метода. Если же таких сведений о функции нет, то методы применять можно, но без гарантии сходимости.

Одним из классов функций, удовлетворяющих указанному условию, является класс **унимодальных (одноэкстремальных)** функций. Дадим определение такой функции для задачи поиска минимума (для задачи поиска максимума определение строится аналогичным образом).

Функция $f(x)$ называется **унимодальной** на отрезке $[a, b]$, если она непрерывна[1] на $[a, b]$ и существуют такие α и β ($a \leq \alpha \leq \beta \leq b$), что:

1. на отрезке $[a, \alpha]$ при $a < \alpha$ $f(x)$ монотонно убывает;
2. на отрезке $[\beta, b]$ при $\beta < b$ $f(x)$ монотонно возрастает;
3. существует минимум $f(x)$ при $x \in [\alpha, \beta]$.

Примеры унимодальных функций приведены на рисунке 2.1.

[1] В общем случае это не так, но мы под унимодальной функцией будем подразумевать непрерывную унимодальную функцию.

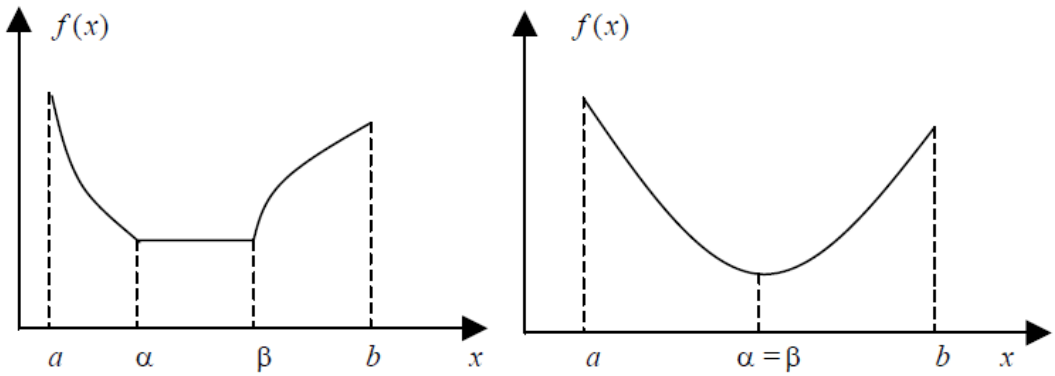

Рис. 2.1. Примеры унимодальных функций

Прежде чем приступить к процедуре оптимизации, следует по возможности установить принадлежность целевой функции классу, для которого гарантирована сходимость процесса.

Заметим, что предположение об унимодальности функции в окрестности точки экстремума весьма естественно. Получение информации о таком промежутке является важным предварительным этапом процедуры оптимизации.

Рассмотрим с общих позиций ряд методов (разделы 2.1.1 и 2.1.2), позволяющих находить экстремумы (далее при рассмотрении алгоритмов будем говорить о задаче поиска минимума) унимодальных функций на отрезке $[a, b]$. Эти методы объединяет идея сокращения **текущего интервала неопределенности (ТИН)**. Она состоит в том, что в процессе поиска исключаются из рассмотрения те подынтервалы, на которых точка оптимума x^* отсутствует в силу унимодальности целевой функции.

2.1.1 Пассивные методы поиска

На практике подобные методы применяются, когда удобно провести независимые эксперименты по измерению значений функции $f(x)$, а последовательное измерение этих значений трудоемко или невозможно по каким-либо причинам.

Метод равномерного поиска

Испытания проводятся в точках, которые определяются путем равномерного деления отрезка $[a, b]$ на N одинаковых интервалов. Из вычисленных в узлах полученной сетки значений целевой функции выбирается наименьшее (пусть это значение достигается в некотором узле x_k. Тогда, в связи с унимодальностью целевой функции, подынтервалы $[a, x_{k-1}]$ и $[x_{k+1}, b]$ можно исключить из рассмотрения, то есть сделать очередным ТИН отрезок $[x_{k-1}, x_{k+1}]$.

Алгоритм метода:

1. Выполняем присваивание $r = 1, a^1 = a, b^1 = b, \text{ТИН}_1 = [a^1, b^1]$.
2. На очередном ТИН строим равномерную сетку с $N + 1$ узлом.
3. Вычисляем значения целевой функции в узлах сетки.
4. Находим минимальное из вычисленных значений $\min(f(x_0^r), f(x_1^r), \ldots, f(x_N^r)) = f(x_k^r)$.

5. Выполняем присваивание $a^{r+1} = x^r_{k-1}, b^{r+1} = x^r_{k+1}, \text{ТИН}_{r+1} = [a^{r+1}, b^{r+1}]$.

6. Если $|\text{ТИН}_{r+1}| \leq \epsilon_x$, то заканчиваем вычисления; иначе выполняем присваивание $r = r + 1$ и переходим к шагу 2.

В качестве приближенного значения точки минимума x^* может быть принята любая точка последнего ТИН.

На рисунке **??** показан один шаг метода равномерного поиска при $N = 13$.

Поскольку после каждой итерации длина ТИН уменьшается в фиксированное ($\frac{N}{2}$) количество раз, можно априорно оценить количество итераций по заданной точности решения. Действительно, после первой итерации $|\text{ТИН}_2| = 2\frac{b-a}{N}$, после второй итерации $|\text{ТИН}_3| = 2((2\frac{b-a}{N})/N) = (b-a)(\frac{2}{N})^2$ и т.д. Тогда после r-ой итерации имеем $|\text{ТИН}_{r+1}| = (b-a)(\frac{2}{N})^r$. Но, так как условие окончания поиска $|\text{ТИН}_{r+1}| \leq \epsilon_x$, получаем $(b-a)(\frac{2}{N})^r \leq \epsilon_x$. Заменив знак неравенства на равенство и выразив r, получим оценку количества итераций.

Метод поразрядного поиска

Можно усовершенствовать метод равномерного поиска, уменьшив количество вычислений значений целевой функции на каждой итерации. Во-первых, если $f(x^r_i) < f(x^r_{i+1})$, то отпадает необходимость вычислять $f(x)$ в точках x^r_{i+2}, x^r_{i+3} и т.д. Во-вторых, разумно было бы сначала грубо определить отрезок, содержащий оптимальную точку; а затем на этом отрезке искать ее с меньшим шагом дискретизации. Иными словами, следует сделать шаг сетки не постоянным, а зависящим от итерации.

В методе поразрядного поиска перебор точек происходит с некоторым шагом h_r до тех пор, пока не выполнится условие $f(x^r_i) < f(x^r_{i+1})$, или пока очередная из точек не совпадет с концом отрезка. После этого шаг уменьшается (обычно в 4 раза) и перебор точек с новым шагом осуществляется в противоположном направлении до тех пор, пока значения $f(x)$ снова не перестанут уменьшаться или не будет достигнут противоположный край отрезка и т.д. Описанный процесс завершается, когда перебор в каком-либо направлении закончен, а длина шага $|h_r| \leq \epsilon_x$.

Алгоритм метода:

1. Выполняем присваивание $r = 1, i = 0, x^r_i = a, h_r = \frac{b-a}{4}$.

2. Делаем очередной шаг $x^r_{i+1} = x^r_i + h_r$; проверяем, что $x^r_{i+1} \in [a, b]$: если принадлежит, то переходим к шагу 3, в противном случае - к шагу 4.

3. Вычисляем и сравниваем $f(x^r_i)$ и $f(x^r_{i+1})$. Если $f(x^r_i) \geq f(x^r_{i+1})$, то положить $i = i + 1$ и перейти к шагу 2, в противном случае - к шагу 4.

4. Полагаем $x^* = x^r_i$ и проверяем условие окончания поиска: если $|h_r| \leq \epsilon_x$, то завершаем вычисления; в противном случае переходим к шагу 5.

5. Переходим к следующей итерации: изменяем направление поиска и уменьшаем шаг. Для этого выполняем присваивание $r = r + 1, i = 0, x_i^r = x^*, h_r = \frac{h_{r-1}}{4}$ и переходим к шагу 2.

2.1.2 Последовательные методы поиска

Последовательные методы поиска на практике используются чаще, чем пассивные. Это обусловлено тем, что использование информации о результатах предыдущих измерений для выбора очередной экспериментальной точки x_i как правило приводит к более эффективному поиску.

Метод дихотомии

В методе дихотомии испытания проводятся парами. Точки каждой последующей пары разнесены между собой на величину $\delta_x < \epsilon_x$. Испытания производятся в середине ТИН. По значениям $f(x)$, полученным в этих точках, одна половина ТИН в силу унимодальности целевой функции исключается из дальнейшего рассмотрения.

Алгоритм метода:

1. Выполняем присваивание $r = 1, a^1 = a, b^1 = b, \text{ТИН}_1 = [a^1, b^1]$.

2. Вычисляем величины $x_0^r = \frac{b^r - a^r}{2}$, $x_1^r = x_0^r - \frac{\delta_x}{2}$, $x_2^r = x_0^r + \frac{\delta_x}{2}$.

3. Вычисляем значения $f(x_1^r)$ и $f(x_2^r)$.

4. Если $f(x_1^r) < f(x_2^r)$, то выполняем присваивание $a^{r+1} = a^r$, $b^{r+1} = x_0^r$; в противном случае $a^{r+1} = x_0^r$, $b^{r+1} = b^r$. В обоих случаях $\text{ТИН}_{r+1} = [a^{r+1}, b^{r+1}]$.

5. Если $|\text{ТИН}_{r+1}| \le \epsilon_x$, то заканчиваем вычисления; иначе выполняем присваивание $r = r + 1$ и переходим к шагу 2.

В качестве приближенного значения точки минимума x^* может быть принята любая точка последнего ТИН.

На рисунке **??** показан один шаг метода дихотомии.

Поскольку после каждой итерации длина ТИН уменьшается в 2 раза, можно априорно оценить количество итераций по заданной точности решения. Действительно, после первой итерации $|\text{ТИН}_2| = \frac{b-a}{2}$, после второй итерации $|\text{ТИН}_3| = ((\frac{b-a}{2})/2) = \frac{b-a}{2^2}$ и т.д. Тогда после r-ой итерации имеем $|\text{ТИН}_{r+1}| = \frac{b-a}{2^r}$. Но, так как условие окончания поиска $|\text{ТИН}_{r+1}| \le \epsilon_x$, получаем $\frac{b-a}{2^r} \le \epsilon_x$. Заменив знак неравенства на равенство и выразив r, получим оценку количества итераций.

Замечание 1. Если производная целевой функции считается достаточно просто, можно видоизменить метод дихотомии. На концах рассматриваемого отрезка вычисляют производные целевой функции, отрезок делят пополам и вычисляют производную в средней точке.

Для следующей итерации выбирают тот отрезок из двух получившихся, на концах которого знаки производной различны (по определению унимодальной функции ее производная меняет знак на отрезке поиска только один раз).

Замечание 2. В методе дихотомии на каждой итерации значение функции вычисляется дважды. Однако, как было указано в разделе 1.6, в общем случае стараются уменьшить количество таких вычислений. Поэтому более эффективными являются методы, где новые «экспериментальные» точки на каждой (по возможности) итерации выбираются таким образом, чтобы значение функции приходилось вычислять только один раз.

Метод Фибоначчи

Идея метода Фибоначчи состоит в том, чтобы определять новые «экспериментальные» точки с помощью чисел Фибоначчи, поэтому сначала следует ввести определение и рассмотреть свойства этих чисел. Числа Фибоначчи задаются рекуррентным соотношением $F_i = F_{i-1} + F_{i-2},\ i \geq 2,\ F_0 = F_1 = 1$. Вычислять числа Фибоначчи нерекуррентным образом можно с помощью выражения

$$F_i = \frac{\left(\frac{1}{\tau}\right)^{i+1} - (-\tau)^{i+1}}{\sqrt{5}}, \tag{2.1}$$

где $\tau = \frac{\sqrt{5}-1}{2} \approx 0,618$ - решение квадратного уравнения

$$\tau^2 + \tau - 1 = 0 \tag{2.2}$$

При больших значениях i членом $(-\tau)^{i+1}$ можно пренебречь. Тогда

$$F_i \approx \frac{\left(\frac{1}{\tau}\right)^{i+1}}{\sqrt{5}} \tag{2.3}$$

Можно заметить, что из (2.3) следует $F_{i-1} \approx \frac{\left(\frac{1}{\tau}\right)^{i}}{\sqrt{5}}$ и, следовательно,

$$\frac{F_{i-1}}{F_i} \approx \tau, \tag{2.4}$$

то есть при больших i отношение двух соседних чисел Фибоначчи примерно постоянно.

Алгоритм метода состоит из двух этапов.

Первый этап состоит из $(N-1)$-ой итерации для $r = 1, 2, \ldots, N-1$. Рассмотрим схему r-ой итерации, когда $\text{ТИН}_r = [a^r, b^r]$:

1. Вычисляем величины $x_1^r = a^r + |\text{ТИН}_r| \frac{F_{N-r-1}}{F_{N-r+1}}$, $x_2^r = a^r + |\text{ТИН}_r| \frac{F_{N-r}}{F_{N-r+1}}$.
2. Вычисляем значения $f(x_1^r)$ и $f(x_2^r)$.
3. Если $f(x_1^r) < f(x_2^r)$, то выполняем присваивание $a^{r+1} = a^r$, $b^{r+1} = x_2^r$; в противном случае $a^{r+1} = x_1^r$, $b^{r+1} = b^r$. В обоих случаях $\text{ТИН}_{r+1} = [a^{r+1}, b^{r+1}]$.

Данный этап алгоритма обладает тем свойством, что после выполнения $(N-1)$-ой итерации имеет место следующая ситуация: $x_1^{N-1} = x_2^{N-1} = x^{N-1}$. Таким образом, на $(N-1)$-ой итерации сужения ТИН не происходит: $\text{ТИН}_N = \left[a^{N-1}, b^{N-1}\right] = \text{ТИН}_{N-1}$, причем точка x^{N-1} оказывается в середине ТИН_{N-1}.

Второй этап позволяет определить, с какой стороны от точки x^{N-1} лежит точка минимума. Для этого:

1. Находим точку $x^N = x^{N-1} + \delta_x$, где $\delta_x \ll |\text{ТИН}_{N-1}|$ - свободный параметр алгоритма.
2. Вычисляем значение $f(x^N)$.
3. Если $f(x^N) > f(x^{N-1})$, то выполняем присваивание $\text{ТИН}_{N+1} = \left[a^{N-1}, x^{N-1}\right]$; в противном случае $\text{ТИН}_{N+1} = \left[x^{N-1}, b^{N-1}\right]$.

В качестве приближенного значения точки минимума x^* может быть принята любая точка последнего ТИН.

Некоторые **свойства метода Фибоначчи**.

Утверждение 1. Для любого $r = 1, 2, \ldots, N-2$ метод Фибоначчи обладает следующим свойством: одна из точек x_1^{r+1}, x_2^{r+1} совпадает с одной из точек x_1^r, x_2^r.

◄ Пусть на r-ой итерации выполняется $f(x_1^r) < f(x_2^r)$, тогда $\text{ТИН}_{r+1} = [a^r, x_2^r]$ и $x_1^r \in \text{ТИН}_{r+1}$. Рассмотрим точку $x_2^{r+1} = a^r + |\text{ТИН}_{r+1}| \frac{F_{N-(r+1)}}{F_{N-(r+1)+1}} = a^r + (x_2^r - a^r) \frac{F_{N-r-1}}{F_{N-r}}$. Так как $x_2^r = a^r + |\text{ТИН}_r| \frac{F_{N-r}}{F_{N-r+1}}$, то $x_2^{r+1} = a^r + \left(a^r + |\text{ТИН}_r| \frac{F_{N-r}}{F_{N-r+1}} - a^r\right) \frac{F_{N-r-1}}{F_{N-r}} = a^r + |\text{ТИН}_r| \frac{F_{N-r-1}}{F_{N-r+1}} = x_1^r$ ►

Доказательство для случая $f(x_1^r) > f(x_2^r)$ проводится аналогично. Указанное свойство позволяет на каждой итерации (кроме первой) производить испытания только в одной точке.

Утверждение 2. Точки x_1^r, x_2^r расположены симметрично относительно концов ТИН.

◄ В соответствии с алгоритмом имеем: $x_1^r - a^r = a^r + (b^r - a^r) \frac{F_{N-r-1}}{F_{N-r+1}} - a^r = (b^r - a^r) \frac{F_{N-r-1}}{F_{N-r+1}}$; $b^r - x_2^r = b^r - a^r - (b^r - a^r) \frac{F_{N-r}}{F_{N-r+1}} = (b^r - a^r) \left(1 - \frac{F_{N-r}}{F_{N-r+1}}\right) = (b^r - a^r) \left(\frac{F_{N-r+1} - F_{N-r}}{F_{N-r+1}}\right)$. Так как из определения чисел Фибоначчи следует, что $F_{N-r} = F_{N-r+1} - F_{N-r-1}$, второе выражение можно записать как $b^r - x_2^r = (b^r - a^r) \frac{F_{N-r-1}}{F_{N-r+1}}$. Получили, что $x_1^r - a^r = b^r - x_2^r$ ►

Утверждение 3. В результате любой итерации $r = 1, 2, \ldots, N-2$ длина ТИН уменьшается в $\frac{F_{N-r}}{F_{N-r+1}}$ раз.

◄ Из утверждения 2 следует, что $x_2^r - a^r = b^r - x_1^r$, поэтому достаточно рассмотреть только один из отрезков $[a^r, x_2^r]$, $[x_1^r, b^r]$. Рассмотрим первый отрезок: $x_2^r - a^r = a^r + (b^r - a^r) \frac{F_{N-r}}{F_{N-r+1}} - a^r = (b^r - a^r) \frac{F_{N-r}}{F_{N-r+1}}$ ►

Утверждение 4. Из утверждения 3 и (2.4) следует, что при достаточно больших N в результате одной итерации длина ТИН уменьшается в τ раз.

Утверждение 5. В результате N итераций длина ТИН становится равной $\frac{b-a}{F_N}$.

◄ Действительно, после первой итерации $|\text{ТИН}_2| = (b-a) \frac{F_{N-1}}{F_N}$, после второй итерации $|\text{ТИН}_3| = (b-a) \frac{F_{N-1}}{F_N} \frac{F_{N-2}}{F_{N-1}} = (b-a) \frac{F_{N-2}}{F_N}$ и т.д.; после $(N-2)$-ой итерации имеем $|\text{ТИН}_{N-1}| = (b-a) \frac{F_2}{F_N} = (b-a) \frac{2}{F_N}$, после $(N-1)$-ой итерации длина ТИН не меняется, после N-ой итерации длина ТИН уменьшается ровно в два раза (так как из утвер-

ждения 2 и выражения $x_1^{N-1} = x_2^{N-1} = x^{N-1}$ следует, что x^{N-1} делит ТИН пополам): $|\text{ТИН}_{N+1}| = (b-a)\,\frac{2}{F_N}/2 = \frac{b-a}{F_N}$ ►

Метод Фибоначчи является достаточно эффективным (см. раздел 2.1.3) методом одномерной оптимизации, однако в нем доставляет неудобство тот факт, что после выбора значения параметра N следует обязательно сделать $N-1$ итераций, без возможности остановки даже в том случае, когда длина очередного ТИН стала меньше заданной точности решения. От этого недостатка можно избавиться следующим образом: из (2.4) следует, что отношение соседних чисел Фибоначчи примерно постоянно, следовательно, на каждой итерации можно выбирать экспериментальные точки так, чтобы они делили ТИН в одном и том же отношении τ, которое не зависит от N. Эти рассуждения ложатся в основу метода золотого сечения.

Метод золотого сечения

Метод золотого сечения является почти таким же эффективным, как метод Фибоначчи, но позволяет остановить вычисления на любой итерации. Определение золотого сечения дается следующим образом: говорят, что точка c выполняет золотое сечение отрезка $[a,b]$, если $\frac{c-a}{b-a} = \tau$.

Из определения золотого сечения следует, что $\frac{b-c}{b-a} = \frac{(b-a)-(c-a)}{b-a} = 1 - \frac{c-a}{b-a} = 1-\tau$.

Алгоритм метода:

1. Выполняем присваивание $r = 1, a^1 = a, b^1 = b, \text{ТИН}_1 = [a^1, b^1]$.
2. Вычисляем величины $x_1^r = b^r - (b^r - a^r)\,\tau$, $x_2^r = a^r + (b^r - a^r)\,\tau$.
3. Вычисляем значения $f(x_1^r)$ и $f(x_2^r)$.
4. Если $f(x_1^r) < f(x_2^r)$, то выполняем присваивание $a^{r+1} = a^r$, $b^{r+1} = x_2^r$; в противном случае $a^{r+1} = x_1^r$, $b^{r+1} = b^r$. В обоих случаях $\text{ТИН}_{r+1} = [a^{r+1}, b^{r+1}]$.
5. Если $|\text{ТИН}_{r+1}| \le \epsilon_x$, то заканчиваем вычисления; иначе выполняем присваивание $r = r+1$ и переходим к шагу 2.

В качестве приближенного значения точки минимума x^* может быть принята любая точка последнего ТИН.

Некоторые **свойства метода золотого сечения**.

Утверждение 1. Для любого $r \ge 1$ метод золотого сечения обладает следующим свойством: одна из точек x_1^{r+1}, x_2^{r+1} совпадает с одной из точек x_1^r, x_2^r.

◄ Пусть на r-ой итерации выполняется $f(x_1^r) < f(x_2^r)$, тогда $\text{ТИН}_{r+1} = [a^r, x_2^r]$ и $x_1^r \in \text{ТИН}_{r+1}$. Чтобы доказать справедливость утверждения, нужно показать, что верно соотношение $\frac{x_1^r - a^r}{x_2^r - a^r} = \tau$. Из 2-го шага алгоритма следует, что $b^r - x_1^r = (b^r - a^r)\,\tau \Rightarrow b^r - x_1^r - a^r + a^r = (b^r - a^r)\,\tau \;\Rightarrow\; (b^r - a^r) - (x_1^r - a^r) = (b^r - a^r)\,\tau \;\Rightarrow\; x_1^r - a^r = (b^r - a^r)\,(1-\tau)$. Аналогично имеем $x_2^r - a^r = (b^r - a^r)\,\tau$. Разделив первый из полученных результатов на второй, получаем $\frac{x_1^r - a^r}{x_2^r - a^r} = \frac{1-\tau}{\tau}$. Но из (2.2) следует, что $1 - \tau = \tau^2$, в результате чего имеем $\frac{x_1^r - a^r}{x_2^r - a^r} = \frac{1-\tau}{\tau} = \frac{\tau^2}{\tau} = \tau$ ►

Доказательство для случая $f(x_1^r) > f(x_2^r)$ проводится аналогично. Указанное свойство позволяет на каждой итерации (кроме первой) производить испытания только в одной точке.

Утверждение 2. Точки x_1^r, x_2^r расположены симметрично относительно концов ТИН.

◄ Из 2-го шага алгоритма следует, что точка x_1^r отстоит от точки b^r на величину $(b^r - a^r)\,\tau$; точка x_2^r отстоит от точки a^r на ту же величину. ►

Утверждение 3. Поскольку после каждой итерации длина ТИН уменьшается в τ раз, можно априорно оценить количество итераций по заданной точности решения. Действительно, после первой итерации $|\text{ТИН}_2| = (b-a)\,\tau$, после второй итерации $|\text{ТИН}_3| = (b-a)\,\tau\tau = (b-a)\,\tau^2$ и т.д. Тогда после r-ой итерации имеем $|\text{ТИН}_{r+1}| = (b-a)\,\tau^r$. Но, так как условие окончания поиска $|\text{ТИН}_{r+1}| \le \epsilon_x$, получаем $(b-a)\,\tau^r \le \epsilon_x$. Заменив знак неравенства на равенство и выразив r, получим оценку количества итераций.

2.1.3 Сравнение эффективности методов, сокращающих ТИН

Обозначим через Ф класс непрерывных унимодальных одномерных функций. Пусть множество рассматриваемых методов поиска есть $\{A_1, A_2, A_3, A_4\}$, где

- A_1 - метод равномерного поиска;
- A_2 - метод дихотомии;
- A_3 - метод Фибоначчи;
- A_4 - метод золотого сечения.

В качестве критерия оптимальности указанных методов на классе функций Ф будем использовать максимальную длину ТИН после m **испытаний**:

$$W(A) = \max_{f \in \Phi} |\text{ТИН}_m(f, A)| \tag{2.5}$$

Использование подобного критерия обусловлено тем, что при решении оптимизационной задачи зачастую требуется минимизировать число испытаний (это уже обсуждалось в разделе 1.6). Но, поскольку в предыдущем разделе длина ТИН связывалась с количеством итераций (а, как было показано, почти во всех методах число итераций не совпадает с числом испытаний), необходимо в выведенных формулах сделать переход от итераций к испытаниям.

При рассмотрении метода равномерного поиска было показано, что после каждой итерации длина ТИН уменьшается в $\frac{N}{2}$ раз, а количество испытаний на каждой итерации $m = N + 1$. Следовательно, $W(A_1) = \frac{2}{N}\,(b-a) = \frac{2}{m-1}\,(b-a)$.

При рассмотрении метода дихотомии было показано, что после каждой итерации длина ТИН уменьшается в 2 раза, количество испытаний на каждой итерации равно 2, поэтому $m = 2N$. Следовательно, $W(A_2) = \frac{b-a}{2^N} = \frac{b-a}{2^{m/2}}$.

При рассмотрении метода Фибоначчи было показано, что в результате N итераций длина ТИН становится равной $\frac{b-a}{F_N}$, а количество испытаний $m = N + 1$ (на первой итерации два

испытания, дальше по одному испытанию на каждой итерации). Следовательно, $W(A_3) = \frac{b-a}{F_{m-1}}$.

При рассмотрении метода золотого сечения было показано, что в результате N итераций длина ТИН становится равной $(b-a)\,\tau^N$, а количество испытаний $m = N+1$ (на первой итерации два испытания, дальше по одному испытанию на каждой итерации). Следовательно, $W(A_4) = (b-a)\,\tau^{m-1}$.

Пользуясь полученными результатами, сравним метод дихотомии и метод Фибоначчи при числе испытаний $m = 14$: $\frac{W(A_2)}{W(A_3)} = \frac{F_{m-1}}{2^{m/2}} = \frac{F_{13}}{2^7} = \frac{233}{128} \approx 1,82$. Видно, что метод Фибоначчи почти в два раза эффективнее метода дихотомии.

2.1.4 Методы полиномиальной аппроксимации

Если при решении оптимизационной задачи была получена информация о том, что целевая функция является достаточно гладкой в некоторой окрестности экстремума, то в этой окрестности ее можно достаточно точно заменить (аппроксимировать) полиномом некоторого порядка, после чего использовать для нахождения экстремума этот полином.

Качество аппроксимации может быть повышено двумя способами:

- Увеличение степени аппроксимирующего полинома;
- Уменьшение сетки аппроксимации.

Второй способ является более предпочтительным, так как построение полинома порядка выше трех - достаточно сложная задача.

Метод квадратичной аппроксимации

В данном методе целевая функция приближенно заменяется параболой, проходящей через три известные точки (x_i, y_i), где $y_i = f(x_i),\ i = 1,2,3$.

Построим полином следующего вида: $q(x) = a_0 + a_(1)(x - x_1 + a_2(x - x_1)(x - x_2)$. Необходимо выбрать $a_i,\ i = 1,2,3$ таким образом, чтобы $q(x_i) = y_{i,\ i=1,2,3}$.

Подставив в полином x_1: $q(x_1) = a_0$, следовательно, $a_0 = y_1$. Подставим в полином x_2: $q(x_2) = a_0 + a_1(x_2 - x_1) = y_2$, следовательно, $a_1 = \frac{y_2 - y_1}{x_2 - x_1}$. Подставим в полином x_3: $q(x_3) = a_0 + a_1(x_3 - x_1) + a_2(x_3 - x_1)(x_3 - x_2) = y_3$, следовательно, $a_2 = \frac{1}{x_3 - x_2}\left(\frac{y_3 - y_1}{x_3 - x_1} - \frac{y_2 - y_1}{x_2 - x_1}\right)$.

Найдем экстремум классическим методом, взяв производную: $q'(x) = a_1 + a_2(x - x_1) + a_2(x - x_2) = 0$, откуда $x^* = \frac{x_1 + x_2}{2} - \frac{a_1}{2a_2}$.

Глава 3

Линейное программирование

Глава 4

Комбинаторная оптимизация

Глава 5

Стохастическая оптимизация

Литература