

**Универсальный метод оптимизации без использования
производных с квадратичной сходимостью**

С. Н. Моисеев

Россия, Воронеж

2011

Аннотация

Предложен новый универсальный метод оптимизации CDOS (Conjugate Direction with Orthogonal Shift), который использует сопряженные направления с ортогональным сдвигом (СНОС). Метод имеет квадратичную сходимость для квадратичных и близких к ним функций. Метод не требует, чтобы целевая функция имела производные или была непрерывна. Для задач с ограничениями метод учитывает ограничения в виде неравенств напрямую. Метод не требует, чтобы ограничения имели производные, были непрерывны. Более того, метод не использует численные значения ограничений в виде неравенств, он использует только сам факт их нарушения или отсутствия нарушений.

Ключевые слова: нелинейная оптимизация, методы свободные от производных, сопряженные направления, квадратичная сходимость

ВВЕДЕНИЕ.....	4
ФОРМУЛИРОВКА ЗАДАЧИ ОПТИМИЗАЦИИ.....	5
ОПИСАНИЕ МЕТОДА	6
АЛГОРИТМ ОДНОМЕРНОГО ЛИНЕЙНОГО ПОИСКА	12
ПОИСК В КРИВОЛИНЕЙНОМ НАПРАВЛЕНИИ	12
ОПТИМИЗАЦИЯ С ОГРАНИЧЕНИЯМИ.....	13
ПОИСК ГЛОБАЛЬНОГО ОПТИМУМА	17
МНОГОМОДОВАЯ ОПТИМИЗАЦИЯ	18
СМЕШАННАЯ ЦЕЛОЧИСЛЕННАЯ-ДИСКРЕТНАЯ-НЕПРЕРЫВНАЯ ОПТИМИЗАЦИЯ	18
СРАВНЕНИЕ С ДРУГИМИ СВОБОДНЫМИ ОТ ПРОИЗВОДНЫХ МЕТОДАМИ, ИМЕЮЩИМИ КВАДРАТИЧНУЮ СХОДИМОСТЬ.....	19
Дифференцируемые функции.....	20
Недифференцируемые функции	22
Оптимизация с ограничениями	23
ЗАКЛЮЧЕНИЕ	25
ССЫЛКИ	25

Введение

На практике часто встречаются целевые функции, которые не являются дифференцируемыми и бывают даже разрывными. Существуют целевые функции в виде сложных программ, которые можно рассматривать как “черные ящики”. Такие целевые функции могут быть даже дифференцируемыми, но их производные недоступны. Быстрые методы оптимизации, которые используют информацию о производных, не могут быть использованы для оптимизации таких целевых функций. В таких случаях обычно используют методы прямого поиска, не требующие производных целевой функции. Одним из таких методов является симплексный метод Нелдера-Мида. Несмотря на то, что метод Нелдера-Мида достаточно надежен для недифференцируемых функций с небольшим числом переменных, он не имеет квадратичной сходимости для квадратичных функций. Как следствие, этот метод оказывается слишком медленным и ненадежным для оптимизационных задач большой размерности.

Существуют методы, не требующие производных, которые имеют квадратичную сходимость для квадратичных и близких к ним функций, такие как методы линейного поиска Пауэлла и Брента. Однако эти методы очень ненадежны для недифференцируемых и разрывных функций. Они также ненадежны для оптимизационных задач с ограничениями.

Ограничения задачи оптимизации также могут быть недифференцируемыми и разрывными. Иногда встречаются ситуации, когда ограничения не заданы явно. Например, когда целевая функция представляет собой сложную программу симуляции телекоммуникационной сети. Такой симулятор имеет большое число параметров. Ограничения для этих параметров включены в программу программистом и не доступны в явном виде. Мы можем только получить сообщение об ошибке при их нарушении. Быстрые регулярные методы, использующие производные, например, SQP (Sequential Quadratic Programming) метод, не могут быть использованы для решения подобных оптимизационных задач.

В этой статье мы представляем новый универсальный свободный от производных метод оптимизации – метод линейного поиска CDOS (Conjugate Direction with Orthogonal Shift), который использует сопряженные направления с ортогональным сдвигом (CHOC). Этот метод был специально разработан для оптимизационных задач, в которых целевая функция и ограничения в виде неравенств представляют собой “черные ящики” и целевая функция не определена вне границ ограничений в виде неравенств.

Этот метод хорошо подходит для решения всех упомянутых выше оптимизационных задач. Метод CDOS имеет высокую скорость также для дифференцируемых функций, т.к. он имеет квадратичную сходимость для квадратичных и близких к ним функций. Он требует минимально возможное число одномерных линейных поисков $\frac{n(n+1)}{2}$, чтобы точно найти оптимум n -мерной квадратичной функции.

Метод не является полностью “жадным”, т.е. он проводит поиск также по направлениям, которые не содержат текущую точку экстремума. Поэтому он может быть использован как база для построения эффективных методов глобальной оптимизации.

Метод CDOS имеет следующие особенности:

- он имеет квадратичную сходимость для квадратичных и близких к ним функций,
- он более надежен, чем метод Нелдера-Мида, при оптимизации недифференцируемых функций,
- он хорошо подходит для решения оптимизационных задач большой размерности,
- он не требует, чтобы целевая функция была определена за границами ограничений,
- он использует ограничения в виде неравенств напрямую,
- он быстро проводит поиск вдоль границ ограничений,
- он может быть легко преобразован в метод поиска глобального оптимума, т.к. он не является полностью “жадным” методом,
- он может быть легко преобразован в метод многомодовой оптимизации, т.к. он имеет квадратичную сходимость,
- он может быть легко преобразован в метод для решения смешанной целочисленной-дискретной-непрерывной оптимизации, т.к. он не является полностью “жадным” методом и имеет квадратичную сходимость,
- его легко запрограммировать.

Формулировка задачи оптимизации

Задача оптимизации в достаточно общем виде может быть сформулирована следующим образом:

минимизировать (максимизировать) целевую функцию

$$f(\mathbf{x}), \mathbf{x} \in R^n,$$

со следующими ограничениями

$$\begin{aligned} \mathbf{c}_0(\mathbf{x}) &\rightarrow \{\text{true}, \text{false}\}, \\ \mathbf{c}_1(\mathbf{x}) &\geq 0, \\ \mathbf{c}_2(\mathbf{x}) &> 0, \\ \mathbf{c}_3(\mathbf{x}) &\neq 0, \\ \mathbf{h}(\mathbf{x}) &= 0, \end{aligned}$$

где \mathbf{x} - переменные целевой функции, f - целевая функция, \mathbf{c}_i - ограничения в виде неравенств, \mathbf{h} - ограничения в виде равенств. Относительно ограничений \mathbf{c}_0 известно только, нарушены ли они (false), или нет (true). Заметим, что ограничения \mathbf{c}_0 могут быть трансформированы в тип \mathbf{c}_2 ограничений $\mathbf{c}_{02} > 0$ следующим образом: $\mathbf{c}_{02} = 1$ когда $\mathbf{c}_0 = \text{true}$ и $\mathbf{c}_{02} = -1$ когда $\mathbf{c}_0 = \text{false}$. Однако заметим, что значения $\{1, -1\}$ ограничений \mathbf{c}_{02} не несут никакой информации о степени нарушений ограничений \mathbf{c}_0 .

Описание метода

CDOS метод, также как и методы Пауэлла и Брента, использует сопряженные линейные направления поиска. Два направления \mathbf{u}_i и \mathbf{u}_j являются сопряженными, если $\mathbf{u}_i^T \mathbf{H} \mathbf{u}_j = 0$, $i \neq j$; $\mathbf{u}_i^T \mathbf{H} \mathbf{u}_j \geq 0$, $i = j$ где \mathbf{H} - положительно определенная матрица Гессе:

$$\mathbf{H} = \left| \frac{\partial}{\partial x_i} \left(\frac{\partial}{\partial x_j} f(\mathbf{x}) \right) \right|, \quad i, j = 1..n.$$

Если функция $f(x_1, \dots, x_n)$ представляет собой n -мерную

квадратичную функцию ее минимум (максимум) будет достигнут после n одномерных поисков в n различных взаимно-сопряженных направлениях. Поэтому точка экстремума n -мерной квадратичной функции будет достигнута после того как n сопряженных направлений будут построены в первый раз.

В процессе построения сопряженных направлений в CDOS методе используется ортогональный сдвиг от подмножества уже построенных сопряженных направлений. Этот ортогональный сдвиг лишает метод “жадности”, но значительно увеличивает его надежность и придает ему новые уникальные свойства. Эти новые свойства принципиально отличают метод CDOS от других методов, имеющих квадратичную сходимость. Главные из этих свойств – это быстрое прохождение вдоль границ ограничений и резко возросшая надежность при оптимизации недифференцируемых функций.

Алгоритм минимизации состоит из трех стадий. На первой стадии первое направление поиска определяется как направление противоположное квазиградиенту целевой функции. На второй стадии конструируются первые n сопряженных направлений

с использованием ортогонального сдвига. Третья стадия представляет собой основной итеративный цикл. В этом цикле происходит обновление уже построенных сопряженных направлений с использованием ортогонального сдвига. Ниже мы рассмотрим эти три стадии более детально на примере минимизации n -мерной функции $f(x_1, \dots, x_n)$, $n > 1$.

Стадия I. Начальные n направлений поиска совпадают с координатными осями: $\mathbf{u}_1 = (1, 0, 0, \dots, 0)^T$, $\mathbf{u}_2 = (0, 1, 0, \dots, 0)^T$, ..., $\mathbf{u}_n = (0, 0, 0, \dots, 1)^T$. Начиная с начальной точки, делается по одному шагу в каждом направлении. По умолчанию в качестве начальной точки берется точка $\mathbf{x}_0 = (0.9, \dots, 0.9)$, если пользователем не определена другая точка. Начальное значение шага λ равно 1, если пользователем не определено другое значение начального шага. Вычисляется вектор приращений значений функции в каждом направлении $\mathbf{s} = (\Delta f_1, \dots, \Delta f_n)^T$. В качестве первого сопряженного направления мы выбираем направление антиградиента, которое совпадает с нормализованным вектором $\mathbf{u}_1^* = -\frac{\mathbf{s}}{\|\mathbf{s}\|}$. Первое направление поиска меняется на направление антиградиента $\mathbf{u}_1 = \mathbf{u}_1^*$ и в этом направлении осуществляется одномерный линейный поиск минимума. Обозначим через $\mathbf{x}_{\min}^{(1)}$ полученную точку минимума.

Стадия II. Исходные n направлений поиска на этой стадии: $\mathbf{u}_1 = \mathbf{u}_1^*$, $\mathbf{u}_2 = (0, 1, 0, \dots, 0)^T$, ..., $\mathbf{u}_n = (0, 0, 0, \dots, 1)^T$. Величина шага λ на этой стадии совпадает с начальной величиной шага. Величина ортогонального сдвига рассчитывается как $\lambda_s = 0.62\lambda$ и не меняется на протяжении всей Стадии II. Мы опишем построение оставшихся $n - 1$ сопряженных направлений в виде следующего псевдокода.

for i from 2 to n do # цикл построения 2, 3, ..., n сопряженных направлений

Мы делаем ортогональный сдвиг от уже построенных сопряженных направлений $\mathbf{u}_1, \dots, \mathbf{u}_{i-1}$ от точки текущего минимума $\mathbf{x}_{\min}^{(i-1)}$. Направление ортогонального сдвига может быть получено с помощью процедуры ортогонализации Грамма-Шмидта или с помощью QR декомпозиции матрицы $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_i) = \mathbf{QR}$. Процедура Грамма-Шмидта формирует из исходных линейно независимых векторов $\mathbf{u}_1, \dots, \mathbf{u}_i$ ортонормальное множество векторов $\mathbf{u}_1^*, \dots, \mathbf{u}_i^*$. Вектора $\mathbf{u}_1^*, \dots, \mathbf{u}_{i-1}^*$ не используются. Направление ортогонального сдвига задается вектором \mathbf{u}_i^* . Мы делаем шаг в

этом направлении от текущей точки минимума и получаем точку $\mathbf{y}_{\min}^{(i-1)} = \mathbf{x}_{\min}^{(i-1)} + \lambda_s \mathbf{u}_i^*$.

При использовании QR декомпозиции столбцы матрицы $\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_i)$ являются ортонормальным множеством векторов и направление ортогонального сдвига задается последним i -ым столбцом \mathbf{q}_i матрицы \mathbf{Q} . Мы делаем шаг в этом направлении от текущей точки минимума и получаем точку $\mathbf{y}_{\min}^{(i-1)} = \mathbf{x}_{\min}^{(i-1)} + \lambda_s \mathbf{q}_i$.

QR декомпозиция, которая использует преобразования Хаусхолдера (Householder transformations), быстрее чем процедура Грамма-Шмидта и поэтому более предпочтительна для нахождения направления ортогонального сдвига.

for j from 1 to i-1 do # повторный поиск в $i-1$ сопряженных направлениях

Начиная с точки $\mathbf{y}_{\min}^{(i-1)}$ мы проводим одномерный линейный поиск в направлении \mathbf{u}_j . Если значение функции в полученной точке меньше, чем значение функции в точке $\mathbf{y}_{\min}^{(i-1)}$, мы обновляем точку $\mathbf{y}_{\min}^{(i-1)}$.

end for j.

Направление \mathbf{u}_i замещается на новое направление, которое проходит через точки $\mathbf{x}_{\min}^{(i-1)}$ и $\mathbf{y}_{\min}^{(i-1)}$ следующим образом:

$$\mathbf{u}_i = \begin{cases} \frac{\mathbf{x}_{\min}^{(i-1)} - \mathbf{y}_{\min}^{(i-1)}}{\|\mathbf{x}_{\min}^{(i-1)} - \mathbf{y}_{\min}^{(i-1)}\|}, & f(\mathbf{x}_{\min}^{(i-1)}) < f(\mathbf{y}_{\min}^{(i-1)}), \\ \frac{\mathbf{y}_{\min}^{(i-1)} - \mathbf{x}_{\min}^{(i-1)}}{\|\mathbf{x}_{\min}^{(i-1)} - \mathbf{y}_{\min}^{(i-1)}\|}, & f(\mathbf{x}_{\min}^{(i-1)}) \geq f(\mathbf{y}_{\min}^{(i-1)}). \end{cases}$$

Это направление будет сопряженным ко всем ранее построенным сопряженным направлениям $\mathbf{u}_1, \dots, \mathbf{u}_{i-1}$. Далее мы проводим одномерный линейный поиск минимума в направлении \mathbf{u}_i , начиная с точки $\mathbf{x}_{\min}^{(i-1)}$, если $f(\mathbf{x}_{\min}^{(i-1)}) < f(\mathbf{y}_{\min}^{(i-1)})$, или с точки $\mathbf{y}_{\min}^{(i-1)}$, в противном случае.

Полученная точка минимума становится текущей точкой минимума $\mathbf{x}_{\min}^{(i)}$ для следующего цикла итерации.

end for i.

Если целевая функция представляет собой n -мерную квадратичную функцию, то уже в конце Стадии II достигается ее минимум.

Стадия III. Исходные n направлений поиска на этой стадии совпадают с взаимно сопряженными направлениями, построенными на Стадии II: $\mathbf{u}_1, \dots, \mathbf{u}_n$. Величина начального шага на этой стадии рассчитывается как $\lambda^{(1)} = 0.3 \|\mathbf{x}_{\min}^{(n)} - \mathbf{x}_{\min}^{(n-1)}\| + 0.091 \cdot \lambda$. Если $\lambda^{(1)} = 0$, то $\lambda^{(1)} = \text{tol}$, где tol - это доверительный интервал для точки минимума. Начальная величина ортогонального сдвига рассчитывается как $\lambda_s^{(1)} = 0.62 \lambda^{(1)}$. Если $\lambda_s^{(1)} = 0$, тогда $\lambda_s^{(1)} = \lambda^{(1)}$. Обозначим через $\mathbf{x}_{\min}^{(1)}$ (и $\mathbf{x}_{\min}^{(0)}$) текущую точку минимума. Мы опишем главный итерационный цикл в виде следующего псевдокода.

for i from 1 to infinity do # главный итерационный цикл

Мы делаем ортогональный сдвиг от уже построенных сопряженных направлений $\mathbf{u}_n, \dots, \mathbf{u}_2$ от точки текущего минимума $\mathbf{x}_{\min}^{(i)}$. Направление ортогонального сдвига может быть получено с помощью процедуры ортогонализации Грамма-Шмидта или с помощью QR декомпозиции матрицы $\mathbf{U} = (\mathbf{u}_n, \dots, \mathbf{u}_1) = \mathbf{QR}$. Процедура Грамма-Шмидта формирует из исходных линейно независимых векторов $\mathbf{u}_n, \dots, \mathbf{u}_1$ ортонормальное множество векторов $\mathbf{u}_n^*, \dots, \mathbf{u}_1^*$. Вектора $\mathbf{u}_n^*, \dots, \mathbf{u}_2^*$ не используются. Направление ортогонального сдвига задается вектором \mathbf{u}_1^* . Мы делаем шаг в этом направлении от текущей точки минимума и получаем точку $\mathbf{y}_{\min}^{(i)} = \mathbf{x}_{\min}^{(i)} + \lambda_s^{(i)} \mathbf{u}_1^*$.

При использовании QR декомпозиции матрица $\mathbf{Q} = (\mathbf{q}_1, \dots, \mathbf{q}_n)$ является ортогональной (ее столбцы представляют собой взаимно ортогональные единичные вектора, так что $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$). Направление ортогонального сдвига задается последним столбцом \mathbf{q}_n матрицы \mathbf{Q} . Если $\|\mathbf{q}_n - \mathbf{u}_1\| > \|\mathbf{q}_n + \mathbf{u}_1\|$ мы меняем знак направления ортогонального сдвига \mathbf{q}_n : $\mathbf{q}_n = -\mathbf{q}_n$. Мы делаем шаг в этом направлении от текущей точки минимума и получаем точку $\mathbf{y}_{\min}^{(i-1)} = \mathbf{x}_{\min}^{(i-1)} + \lambda_s \mathbf{q}_n$.

QR декомпозиция, которая использует преобразования Хаусхолдера (Householder transformations), быстрее чем процедура Грамма-Шмидта и поэтому более предпочтительна для нахождения направления ортогонального сдвига.

Мы производим циркулярный сдвиг всего множества сопряженных направлений $\mathbf{u}_1, \dots, \mathbf{u}_n$ в левую сторону. Другими словами мы переобозначаем множество сопряженных направлений следующим образом: вектор \mathbf{u}_2 мы переобозначаем как \mathbf{u}_1 , \mathbf{u}_3 как \mathbf{u}_2 , ..., \mathbf{u}_n как \mathbf{u}_{n-1} и \mathbf{u}_1 как \mathbf{u}_n .

for j from 1 to n-1 do # повторный поиск в $n-1$ сопряженных направлениях

Одномерный поиск минимума внутри этого цикла проводится с шагом величиной $3\lambda^{(i)}$. Величина шага увеличена, поскольку мы сдвинулись от текущей точки минимума и выполняем “не жадный” поиск. Поэтому характерный масштаб изменения функции увеличился.

Начиная с точки $\mathbf{y}_{\min}^{(i)}$ мы проводим одномерный линейный поиск в направлении \mathbf{u}_j . Если значение функции в полученной точке меньше, чем значение функции в точке $\mathbf{y}_{\min}^{(i)}$, мы обновляем точку $\mathbf{y}_{\min}^{(i)}$.

end for j.

Направление \mathbf{u}_n замещается на новое направление, которое проходит через точки $\mathbf{x}_{\min}^{(i)}$ и $\mathbf{y}_{\min}^{(i)}$ следующим образом:

$$\mathbf{u}_n = \begin{cases} \frac{\mathbf{x}_{\min}^{(i)} - \mathbf{y}_{\min}^{(i)}}{\|\mathbf{x}_{\min}^{(i)} - \mathbf{y}_{\min}^{(i)}\|}, & f(\mathbf{x}_{\min}^{(i)}) < f(\mathbf{y}_{\min}^{(i)}), \\ \frac{\mathbf{y}_{\min}^{(i)} - \mathbf{x}_{\min}^{(i)}}{\|\mathbf{x}_{\min}^{(i)} - \mathbf{y}_{\min}^{(i)}\|}, & f(\mathbf{x}_{\min}^{(i)}) \geq f(\mathbf{y}_{\min}^{(i)}). \end{cases}$$

Это направление будет сопряженным ко всем ранее построенным сопряженным направлениям $\mathbf{u}_1, \dots, \mathbf{u}_{n-1}$. Далее мы проводим одномерный линейный поиск минимума в направлении \mathbf{u}_n с шагом $\lambda^{(i)}$, начиная с точки $\mathbf{x}_{\min}^{(i)}$, если $f(\mathbf{x}_{\min}^{(i)}) < f(\mathbf{y}_{\min}^{(i)})$, или с точки $\mathbf{y}_{\min}^{(i)}$, в противном случае. Если значение функции в полученной точке меньше, чем значение функции в точке $\mathbf{x}_{\min}^{(i)}$, мы обновляем точку $\mathbf{x}_{\min}^{(i)}$.

Шаг поиска и величина ортогонального сдвига адаптируются для следующей итерации. Шаг поиска вычисляется как: $\lambda^{(i+1)} = 0.3 \|\mathbf{x}_{\min}^{(i)} - \mathbf{x}_{\min}^{(i-1)}\| + 0.091 \cdot \lambda^{(i)}$. Если $\lambda^{(i+1)} = 0$, то $\lambda^{(i+1)} = \text{tol}$, где tol - это доверительный интервал для точки минимума. Величина ортогонального сдвига вычисляется как $\lambda_s^{(i+1)} = 0.62 \lambda^{(i+1)}$. Если $\lambda_s^{(i+1)} = 0$, то $\lambda_s^{(i+1)} = \lambda^{(i+1)}$.

Теперь мы проверяем следующее условие прекращения поиска. Если следующие неравенства одновременно выполняются N_{exit} раз подряд: $\lambda^{(i+1)} \leq \text{tol}$ и $f(\mathbf{x}_{\min}^{(i-1)}) - f(\mathbf{x}_{\min}^{(i)}) \leq \text{tol}_2$, поиск прекращается. Здесь tol_2 - это доверительный интервал для величины минимума. Для большинства функций достаточно положить

$N_{\text{exit}} = 2$. Для недифференцируемых функций значение $N_{\text{exit}} = 10$ более надежно. В качестве оценки величины минимума мы берем $f(\mathbf{x}_{\min}^{(i)})$, а в качестве точки минимума мы берем $\mathbf{x}_{\min}^{(i)}$. Если условия прекращения поиска не выполняются, мы переходим к следующей итерации. Полученная на предыдущих итерациях точка минимума становится текущей точкой минимума $\mathbf{x}_{\min}^{(i+1)}$ для следующей итерации.

end for i.

На Рис. 1 показана реальная траектория поиска минимума следующей функции

$$f(x, y) = x^2 + y^2 - 1.5xy,$$

Начиная с начальной точки $[x = 5, y = 3]$. Реальная траектория поиска CDOS метода была получена с помощью команды Search пакета DirectSearch, версия 2. Красные кружки обозначают пробные точки поиска.

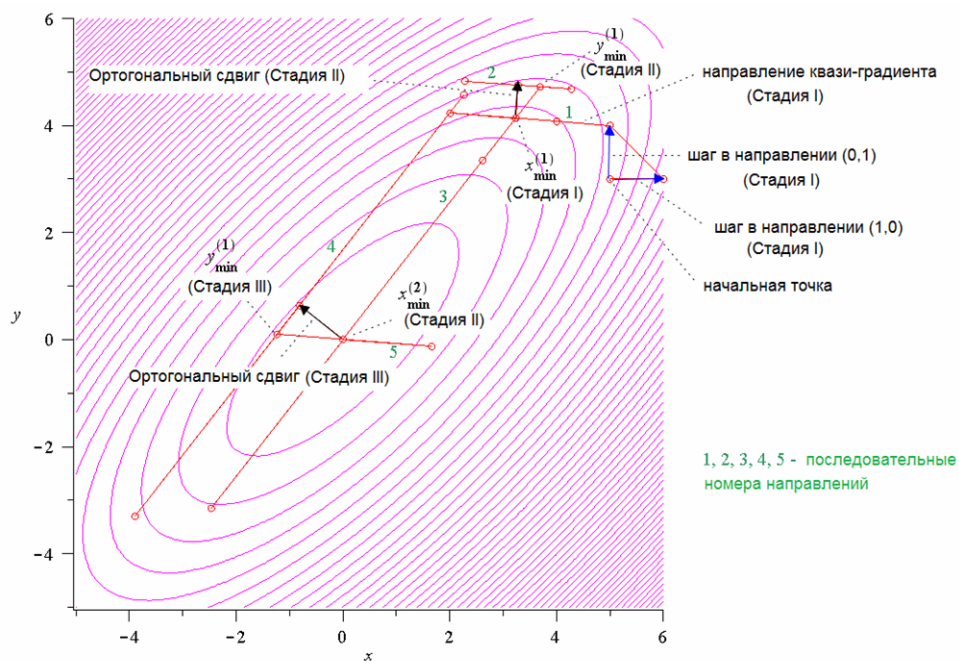


Рис. 1. Траектория поиска

Из рисунка видно, что минимум квадратичной функции достигается в конце Стадии II после прохождения по трем одномерным линейным направлениям. Одна итерация на Стадии III приводит к той же точке минимума, которая получена на Стадии II.

Алгоритм одномерного линейного поиска

Одномерный линейный поиск в случае $n > 1$ осуществляется следующим образом. В текущем направлении поиска делается один шаг. Если шаг оказался успешным точка экстремума обновляется, размер шага увеличивается в два раза и поиск продолжается в том же направлении пока не будет получен первый неуспешный шаг. После этого выполняется однократная параболическая аппроксимация по последним трем точкам. В качестве точки минимума мы берем точку, где значение функции наименьшее. Если первый шаг является неуспешным мы меняем направление поиска на противоположное и повторяем весь процесс.

Параболическая аппроксимация выполняется следующим образом. Пусть $\mathbf{x}', \mathbf{x}'', \mathbf{x}'''$ - три отличные точки, лежащие на одном линейном направлении \mathbf{u}_1 . Пусть $d_{13} > d_{12}, d_{13} > d_{23}$, где $d_{13} = \|\mathbf{x}' - \mathbf{x}'''\|, d_{12} = \|\mathbf{x}' - \mathbf{x}''\|, d_{23} = \|\mathbf{x}'' - \mathbf{x}'''\|$. Тогда точка минимума параболической аппроксимации вычисляется как

$$\mathbf{x}_{\min} = \mathbf{x}' + d \cdot \mathbf{u}_1,$$

где

$$d = \frac{1}{2} \frac{x_0^2(f''' - f'') + x_1^2(f' - f''') + x_2^2(f'' - f')}{f'''(x_0 - x_1) + f''(x_2 - x_0) + f'(x_1 - x_2)},$$
$$f' = f(\mathbf{x}'), f'' = f(\mathbf{x}''), f''' = f(\mathbf{x}'''),$$
$$x_0 = 0, x_1 = d_{12}, x_2 = d_{13}.$$

Поиск в криволинейном направлении

Для того чтобы эффективно двигаться вдоль криволинейных оврагов, таких как в функции Розенброка, CDOS использует поиск в криволинейных направлениях. Один из методов построения таких направлений описан в [3]. Метод CDOS использует другой метод построения криволинейных направлений.

Пусть $\mathbf{x}' = (x'_1, \dots, x'_n), \mathbf{x}'' = (x''_1, \dots, x''_n), \mathbf{x}''' = (x'''_1, \dots, x'''_n)$ - три последовательные отличные точки минимумов, которые были получены на i -ой, $(i + n + 1)$ -ой и $(i + 2n + 2)$ -ой итерациях. Ясно, что $f(\mathbf{x}''') \leq f(\mathbf{x}'') \leq f(\mathbf{x}')$. Из всех n координат мы выбираем такую m -ую координату для которой справедливо

$$x'_m < x''_m < x'''_m \text{ или } x'_m > x''_m > x'''_m.$$

Будем рассматривать x'_m, x''_m, x'''_m как значения аргумента параболической функции, а $x'_k, x''_k, x'''_k, k \neq m$ как значения параболической функции, которые соответствуют этим значениям аргумента. Для каждого $k \neq m$ парабола однозначно определяется этими тремя

точками. Таким образом, мы используем $n-1$ параболических аппроксимаций, чтобы двигаться в криволинейном направлении. Например, когда $x'_m < x''_m < x'''_m$ новая точка, лежащая на криволинейном направлении, будет иметь вид

$$\mathbf{x}_c = (x_1, \dots, x_{m-1}, x'''_m + \lambda_c, x_{m+1}, \dots, x_n),$$

где $x_k, k \neq m$ - соответствующее значение параболы для аргумента $x'''_m + \lambda_c$, $\lambda_c > 0$ - размер шага в криволинейном направлении. Когда $x'_m > x''_m > x'''_m$ новая точка, лежащая на криволинейном направлении, будет иметь вид

$$\mathbf{x}_c = (x_1, \dots, x_{m-1}, x'''_m - \lambda_c, x_{m+1}, \dots, x_n).$$

Если значение функции $f(\mathbf{x}_c)$ в полученной точке меньше, чем ее значение в текущей точке минимума $\mathbf{x}_{\min}^{(i)}$, мы обновляем точку минимума $\mathbf{x}_{\min}^{(i)}$.

Оптимизация с ограничениями

Метод CDOS учитывает нелинейные ограничения напрямую, т.е. он не трансформирует целевую функцию и, более того, он не использует числовые значения ограничений – он использует только факт их нарушений. Поэтому CDOS метод никогда не вычисляет значения функции в точке, которая не удовлетворяет нелинейным ограничениям, вместо этого он ищет допустимую точку.

Пусть мы ищем минимум в направлении \mathbf{u}_k , начиная с допустимой точки \mathbf{x} с шагом λ_0 . Допустим, что точка $\mathbf{x} + \lambda_0 \mathbf{u}_k$ является недопустимой. В этом случае мы уменьшаем размер шага $\lambda_1 = \varepsilon_1 \lambda_0$, $0 < \varepsilon_1 < 1$ и проверяем другую точку $\mathbf{x} + \lambda_1 \mathbf{u}_k$. Если эта точка также является недопустимой, мы повторяем процесс поиска допустимой точки. В случае если мы не можем найти допустимую точку после некоторого количества попыток мы меняем направление поиска на противоположное $-\mathbf{u}_k$. Стратегия уменьшения шага выглядит следующим образом: новый размер шага равен $\lambda_k = \varepsilon_k \lambda_{k-1}$, а последовательность множителей для уменьшения шага имеет вид

$$\varepsilon_k = \begin{cases} 1/1.1, & 0 < k \leq 6, \\ 1/1.2, & 6 < k \leq 8, \\ 1/1.5, & 8 < k \leq 10, \\ 1/2, & 10 < k \leq 16, \\ 1/5, & 16 < k \leq 20, \\ 1/10, & 20 < k \leq 40, \\ 1/100, & 40 < k \leq 50. \end{cases}$$

Для направления ортогонального сдвига стратегия уменьшения шага не должна приводить к нулевому сдвигу. Если мы не можем найти допустимую точку для ортогонального сдвига, мы берем в качестве такой точки любую случайную допустимую точку.

Построение сопряженных направлений с помощью ортогонального сдвига позволяет эффективно проводить поиск вдоль границ нелинейных ограничений. Рис. 2 иллюстрирует процесс поиска вдоль границ ограничений.

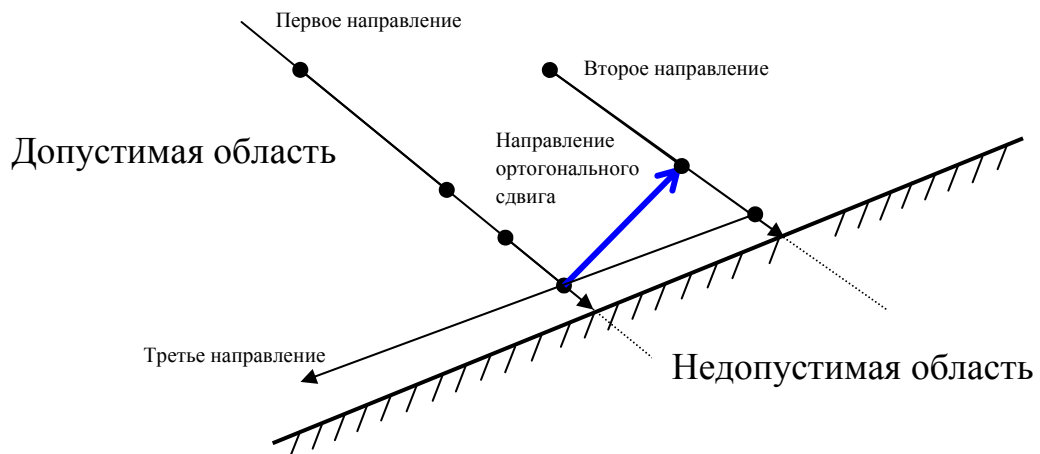


Рис. 2. Поиск вдоль границы ограничений

На Рис. 3 показана реальная траектория поиска минимума следующей функции

$$f(x, y) = x + 10y,$$

с ограничениями

$$y \leq 2x, \quad y \geq \frac{x}{2},$$

начиная с начальной точки $[x = 100, y = 75]$. Реальная траектория поиска CDOS метода была получена с помощью команды Search пакета DirectSearch, версия 2. Недопустимая область показана серым цветом. Красные кружки обозначают пробные допустимые точки поиска.

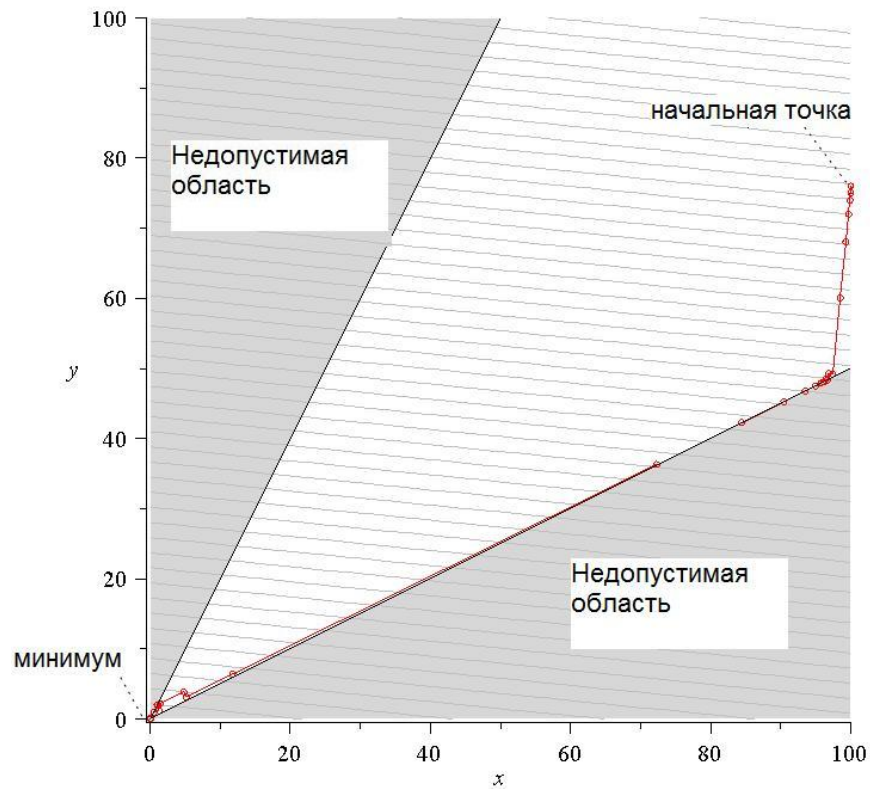


Рис. 3. Траектория поиска

Этот рисунок демонстрирует эффективность CDOS метода для поиска оптимума вдоль линейных границ ограничений. Теперь мы рассмотрим более сложные нелинейные границы ограничений.

На Рис. 4 показана реальная траектория поиска минимума следующей функции

$$f(x, y) = |y - x|^{2.07} + |xy|^{1.07},$$

с ограничениями

$$\{x \leq -1, x \geq -17.001, y \leq -1, y \geq -\frac{1}{3}x - 28, (y + 20)^2 - 3x \geq 51,$$

$$|x + 14.5| + (y + 15)^2 \geq 3, (x + 16)^2 + |y + 8|^{3/2} \geq 20,$$

$$(x + 9.2)^2 + |y + 12| \geq 7, (x + 6)^2 + (y + 15)^2 \geq 29.8, (x + 6)^2 + |y + 1|^{3/2} \geq 15\},$$

начиная с начальной точки $[x = -1.1, y = -27]$. Эта функция имеет один минимум $f_{\min} = 1$ в точке $\mathbf{x}_{\min} = (-1, -1)$. Нелинейные ограничения были выбраны так, чтобы максимально затруднить поиск минимума. Недопустимая область показана серым цветом. Красные кружки обозначают пробные допустимые точки поиска.

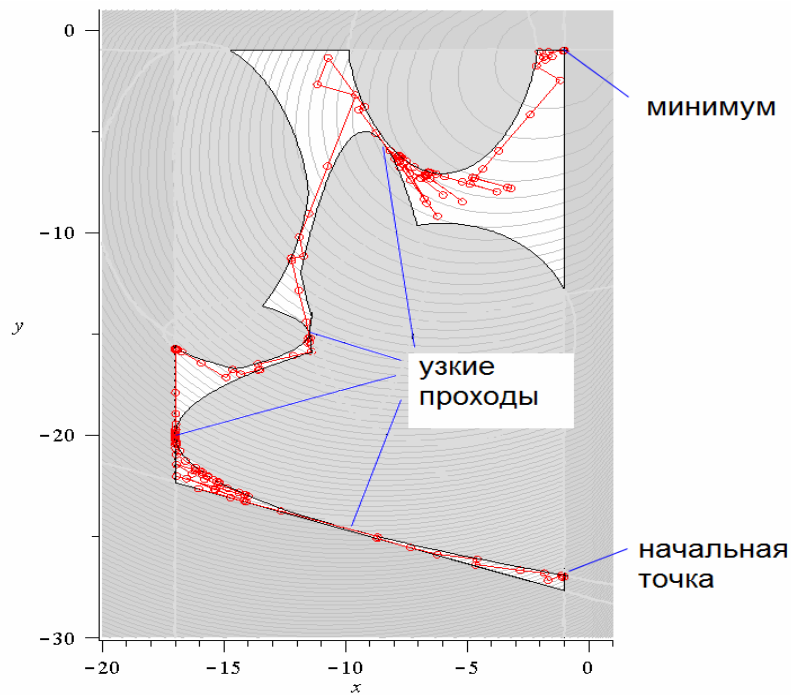


Рис. 4. Траектория поиска

Этот рисунок демонстрирует эффективность CDOS метода для поиска оптимума вдоль нелинейных границ ограничений. Теперь мы рассмотрим более сложную задачу с зигзаго-подобной допустимой областью.

На Fig. 5 показана реальная траектория поиска минимума следующей функции

$$f(x, y) = \frac{|x - 100|}{200} + |y - 101|,$$

с ограничениями

$$\begin{aligned} &\{x \leq 100, x \geq 0, y \leq 101.01, y \geq 0, \\ &|x| + |y - 5|^{7/2} \geq 99.9, |x - 100| + |y - 12|^3 \geq 99.9, \\ &|x| + |y - 19|^{7/2} \geq 99.9, |x - 100| + |y - 26|^3 \geq 99.9, \\ &|x| + |y - 33|^{7/2} \geq 99.9, |x - 100| + |y - 40|^3 \geq 99.9, \\ &|x| + |y - 47|^{7/2} \geq 99.9, |x - 100| + |y - 54|^3 \geq 99.9, \\ &|x| + |y - 61|^{7/2} \geq 99.9, |x - 100| + |y - 68|^3 \geq 99.9, \\ &|x| + |y - 75|^{7/2} \geq 99.9, |x - 100| + |y - 82|^3 \geq 99.9, \\ &|x| + |y - 89|^{7/2} \geq 99.9, |x - 100| + |y - 96|^3 \geq 99.9\}, \end{aligned}$$

начиная с начальной точки $[x = 0, y = 0]$. Эта функция имеет один минимум $f_{\min} = 0$ в точке $x = 100, y = 101$. Нелинейные ограничения были выбраны так, чтобы максимально затруднить поиск минимума. Недопустимая область показана желтоватым цветом. Красные кружки обозначают пробные допустимые точки поиска.

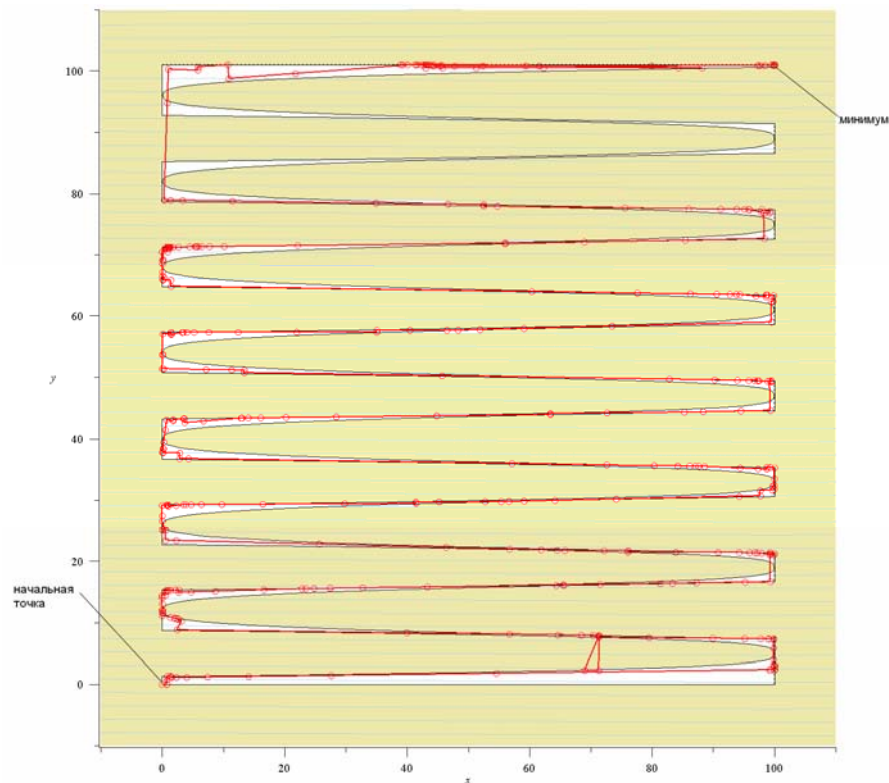


Fig. 5. Трассерия поиска

Этот рисунок демонстрирует эффективность CDOS метода для поиска оптимума вдоль сложных зигзаго-подобных нелинейных границ ограничений.

Для ограничений в виде равенств CDOS использует метод штрафных функций с квадратичной штрафной функцией.

Поиск глобального оптимума

CDOS метод может быть легко преобразован в эффективный метод поиска глобального оптимума, т.к. он не является “жадным” методом поиска. Для этой цели нужно следовать следующим трем правилам.

- 1) Необходимо использовать большой размер шага поиска, например 100. Размер шага должен быть много больше характерного размера области притяжения локальных экстремумов.
- 2) Стратегия уменьшения шага должна быть более медленной по сравнению со стратегией для локальной оптимизации. Для локальной оптимизации стратегия адаптации шага имеет вид $\lambda^{(i+1)} = 0.3 \|\mathbf{x}_{\min}^{(i)} - \mathbf{x}_{\min}^{(i-1)}\| + 0.091 \cdot \lambda^{(i)}$, для глобальной оптимизации стратегия адаптации шага может быть $\lambda^{(i+1)} = \|\mathbf{x}_{\min}^{(i)} - \mathbf{x}_{\min}^{(i-1)}\| + 0.82 \cdot \lambda^{(i)}$. Если мы приблизительно знаем минимальное расстояние d_{\min} между двумя соседними локальными экстремумами, мы

можем использовать более эффективную стратегию для адаптации шага.

Когда текущий размер шага больше чем $\frac{d_{\min}}{2}$ должна использоваться

стратегия адаптации шага для глобальной оптимизации, в противном случае, должна использоваться стратегия адаптации шага для локальной оптимизации.

3) Желательно использовать много начальных точек, например 50 или больше.

Многомодовая оптимизация

Цель многомодовой оптимизации заключается в нахождении всех локальных и глобальных оптимумов (в отличие от нахождения самого лучшего решения).

CDOS метод может быть легко преобразован в эффективный метод многомодовой оптимизации, т.к. он имеет квадратичную сходимость. Для этой цели нужно следовать следующим трем правилам.

- 1) Необходимо использовать маленький размер шага поиска, например 0.005, для того, чтобы не пропустить локальный оптимум. Размер шага должен быть меньше характерного размера области притяжения локальных экстремумов.
- 2) Стратегия уменьшения шага должна быть такой же, как и стратегия для локальной оптимизации. Для локальной оптимизации стратегия адаптации шага имеет вид $\lambda^{(i+1)} = 0.3 \parallel \mathbf{x}_{\min}^{(i)} - \mathbf{x}_{\min}^{(i-1)} \parallel + 0.091 \cdot \lambda^{(i)}$.
- 3) Необходимо использовать много начальных точек, например 100 или больше.

Поскольку метод CDOS имеет квадратичную сходимость, он будет быстрее сходиться к локальным экстремумам по сравнению с методами, не имеющими квадратичной сходимости.

Смешанная целочисленная-дискретная-непрерывная оптимизация

Смешанная целочисленная-дискретная-непрерывная оптимизация является трудной задачей. Отчасти эта задача напоминает глобальную оптимизацию.

Метод CDOS может быть легко преобразован в метод для решения смешанной целочисленной-дискретной-непрерывной оптимизации, который может эффективно решать широкий класс подобных задач. Для этой цели нужно следовать следующим трем правилам.

- 1) Необходимо округлять текущую точку поиска к ближайшей дискретной точке.

- 2) Необходимо использовать большой шаг поиска, например 100. Размер шага должен быть много больше максимального расстояния между двумя соседними дискретными точками.
- 3) Стратегия уменьшения шага должна быть более медленной по сравнению со стратегией для локальной оптимизации. Для локальной оптимизации стратегия адаптации шага имеет вид $\lambda^{(i+1)} = 0.3 \|\mathbf{x}_{\min}^{(i)} - \mathbf{x}_{\min}^{(i-1)}\| + 0.091 \cdot \lambda^{(i)}$, для смешанной оптимизации стратегия адаптации шага может быть $\lambda^{(i+1)} = \|\mathbf{x}_{\min}^{(i)} - \mathbf{x}_{\min}^{(i-1)}\| + 0.51 \cdot \lambda^{(i)}$. Если мы знаем минимальное расстояние d_{\min} между двумя соседними дискретными точками (например, $d_{\min} = 1$ для целочисленных точек), мы можем использовать более эффективную стратегию для адаптации шага. Когда текущий размер шага больше чем $\frac{d_{\min}}{2}$ должна использоваться стратегия адаптации шага для смешанной оптимизации, в противном случае, должна использоваться стратегия адаптации шага для локальной оптимизации.

Сравнение с другими свободными от производных методами, имеющими квадратичную сходимость

Метод CDOS был протестирован на множестве более чем 100 тестовых функций. Для каждой функции использовалось не менее 500 начальных точек. Проверка осуществлялась для разных параметров и опций метода. Результаты проверки показали, что CDOS метод является быстрым и надежным методом оптимизации.

Было проведено сравнение метода CDOS со следующими свободными от производных методами, имеющими квадратичную сходимость:

- Метод сопряженных направлений Пауэлла [1, 2],
- Метод главных осей Брента [3],
- Квадратичный метод (метод последовательной квадратичной аппроксимации) [4].

Методы CDOS, Пауэлла и Брента являются методами линейного поиска. Квадратичный метод использует последовательную квадратичную аппроксимацию целевой функции. Между двумя последовательными точками квадратичной аппроксимации используется одномерный линейный поиск. Метод квадратичной аппроксимации является самым быстрым методом из всех свободных от производных методов для квадратичных функций. Он аналогичен методу Ньютона-Рафсона, но без

использования производных. Квадратичный метод требует минимально возможное число вычислений значений функции $\frac{(n+1)(n+2)}{2} + 1$, чтобы точно найти оптимум n -мерной квадратичной функции.

Программные реализации всех перечисленных методов были взяты из второй версии пакета DirectSearch.

Сравнение различных методов оптимизации является нелегкой задачей из-за возникновения явления детерминированного хаоса в процессе оптимизации. Даже небольшие вариации параметров оптимизации могут проводить к значительным вариациям числа вычислений целевой функции и точности приближений к оптимуму. Поэтому для сравнения необходимо использовать средние характеристики эффективности оптимизации.

Мы будем характеризовать эффективность оптимизации числом вычислений целевой функции, точностью оценки оптимума и надежностью. Надежность рассчитывается как процент успешных оптимизаций.

Для всех методов мы взяли одинаковые ключевые параметры: размер шага равен 1, заданные точности для значения оптимума и точки оптимума равны 10^{-6} . Когда абсолютная разница между оптимумом и его оценкой больше чем “fmin maximum”= 10^{-3} мы считаем, что оптимизация была не успешной.

Дифференцируемые функции

Для квадратичных и близких к ним функций все методы оказались быстрыми и надежными, поскольку все они имеют квадратичную сходимость для таких функций.

Для дифференцируемых, но не квадратичных функций методы линейного поиска в целом оказались более быстрыми по сравнению с квадратичным методом. Для того, чтобы показать типичные характеристики эффективности оптимизации всех рассматриваемых методов для дифференцируемых, но не квадратичных функций, мы выбрали для минимизации функцию Розенброка

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2.$$

Эта функция имеет один минимум $f_{\min} = 0$ в точке $x = 1, y = 1$. Оптимизация проводилась для следующей последовательности из 500 начальных точек:

$$(x_i = -1 + i, y_i = 2 + i), \quad i = 0, 1, 2, \dots, 499.$$

На Рис. 6 показано число вычислений целевой функции для всех четырех методов. На Рис. 7 показана точность оценки минимума целевой функции.

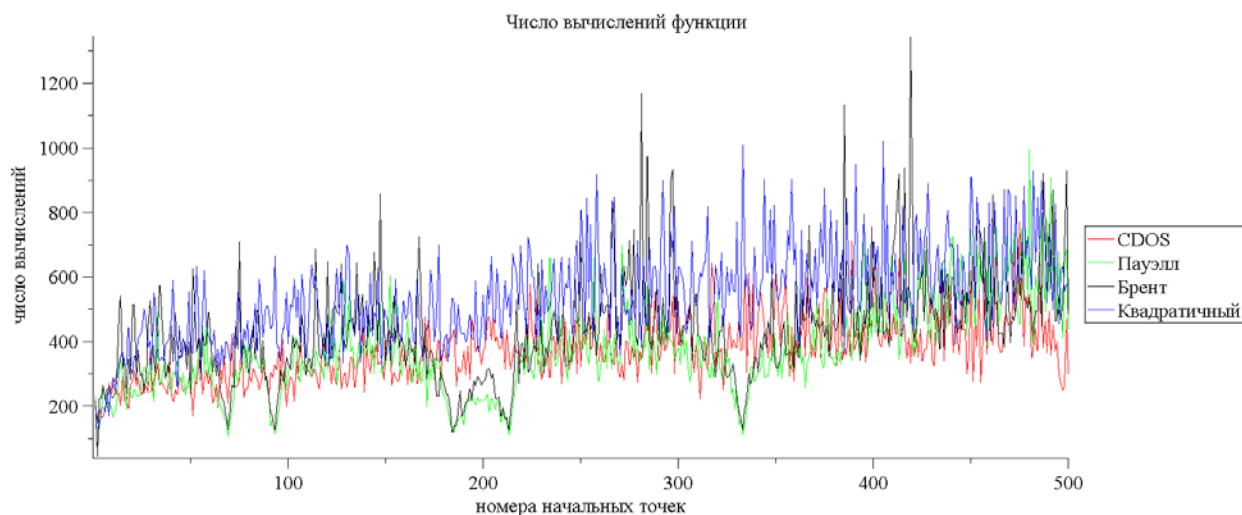


Рис. 6. Число вычислений функции

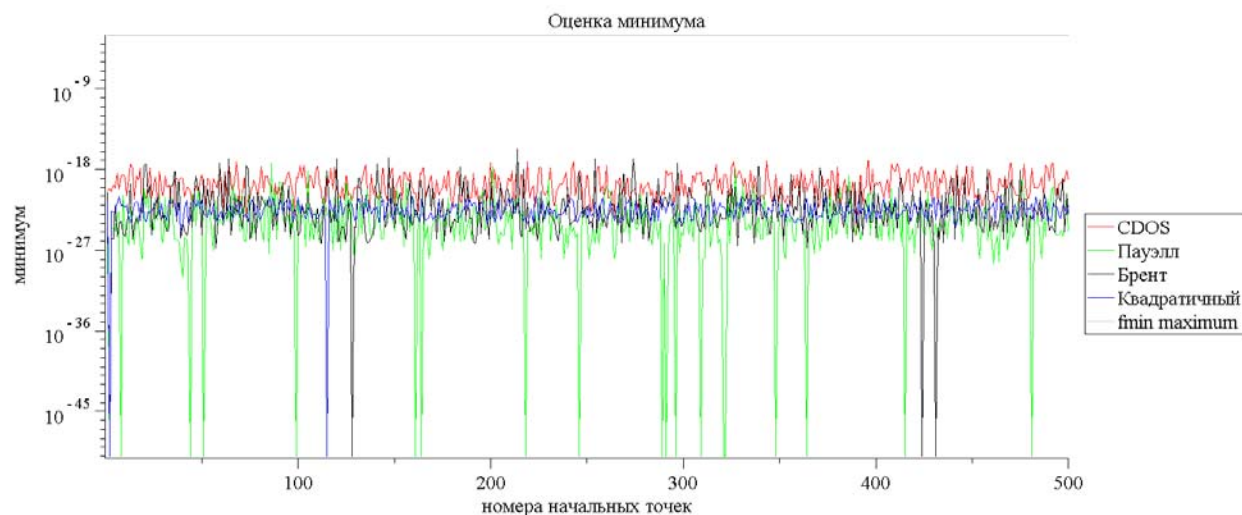


Рис. 7. Достигнутый минимум функции

В Ошибка! Источник ссылки не найден. приведены суммарные показатели эффективности оптимизации для всех методов применительно к тестовой функции Розенброка.

Таб. 1. Суммарная эффективность методов оптимизации

Метод	CDOS	Пауэлл	Брент	Квадратичный
Среднее число вычислений функции	372	402	403	527
Медиана оценки минимума	10^{-20}	10^{-24}	10^{-23}	10^{-23}
Надежность	100%	100%	100%	100%

Из Рис. 7 и Таб. 1 видно, что благодаря квадратичной сходимости всех четырех методов точность оценки минимума целевой функции значительно превышает заданную точность 10^{-6} .

Эффективность оптимизации для методов линейного поиска примерно одинакова. Возможно, методы Пауэлла и Брента быстрее для некоторых плохо масштабированных функций, а метод CDOS более надежен и быстрее когда начальная точка находится далеко от точки оптимума.

Недифференцируемые функции

CDOS является надежным методом для оптимизации недифференцируемых функций, тогда как методы Пауэлла, Брента и Квадратичный метод оказались очень ненадежными при оптимизации таких функций. Следующий пример демонстрирует этот факт.

Для минимизации мы выбрали недифференцируемый аналог функции Розенброка

$$f(x, y) = 100 |y - x^2| + |1 - x|.$$

Эта функция имеет один минимум $f_{\min} = 0$ в точке $x = 1, y = 1$. Оптимизация проводилась для следующей последовательности из 500 начальных точек:

$$(x_i = -1 + i, y_i = 2 + i), \quad i = 0, 1, 2, \dots, 499.$$

На Рис. 8 показано число вычислений целевой функции для всех четырех методов. На Рис. 9 показана точность оценки минимума целевой функции.

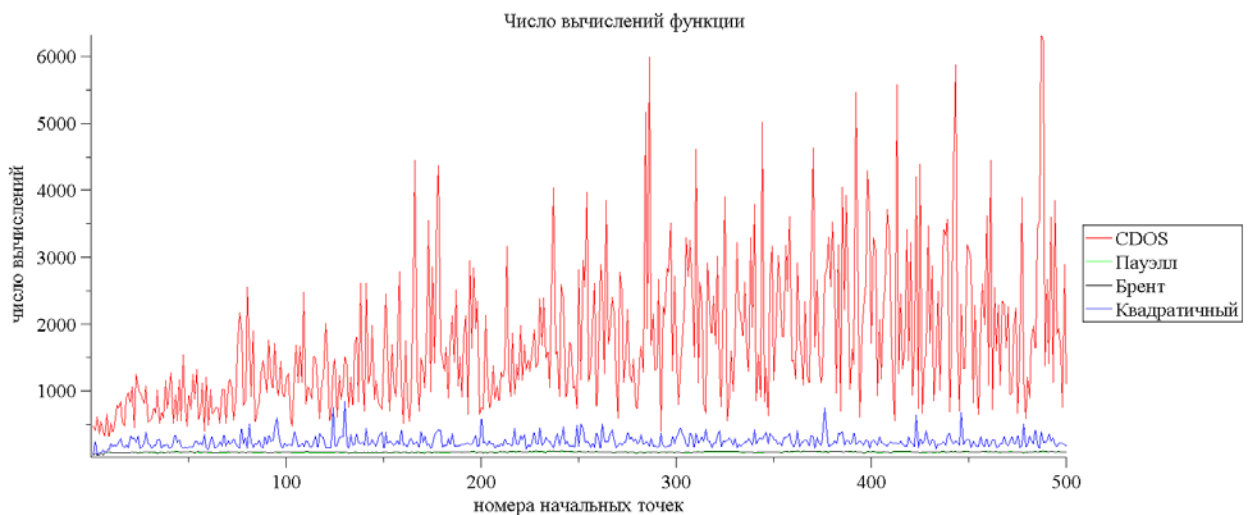


Рис. 8. Число вычислений функции

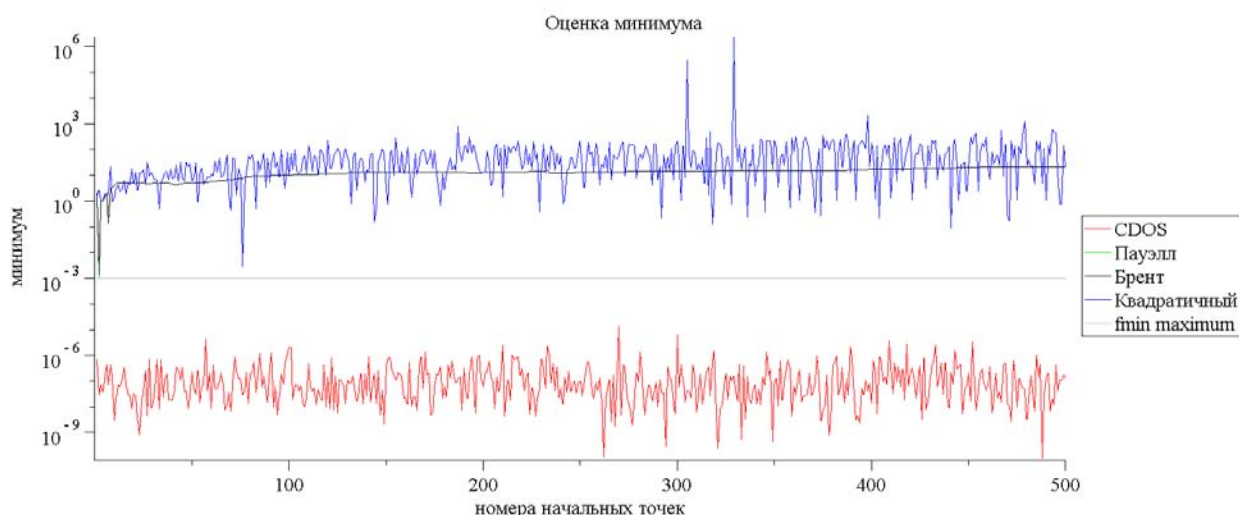


Рис. 9. Достигнутый минимум функции

В Таб. 2 приведены суммарные показатели эффективности оптимизации для всех методов применительно к недифференцируемому аналогу функции Розенброка.

Таб. 2. Суммарная эффективность методов оптимизации

Метод	CDOS	Пауэлл	Брент	Квадратичный
Среднее число вычислений функции	1751	99	102	274
Медиана оценки минимума	$7 \cdot 10^{-8}$	13.5	13.5	30.5
Надежность	100%	0%	0%	0%

Из Рис. 9 и Таб. 2 видно, что надежность методов Пауэлла, Брента и Квадратичного метода равна нулю для этой функции.

Известно, что симплексный метод Нелдера-Мида достаточно хорошо подходит для оптимизации недифференцируемых функций. Этот метод справился с оптимизацией недифференцируемого аналога функции Розенброка, но потребовал примерно в 15 раз больше вычислений целевой функции по сравнению с методом CDOS.

Оптимизация с ограничениями

CDOS метод является надежным для оптимизационных задач с ограничениями, когда ограничения в виде неравенств учитываются напрямую, тогда как методы Пауэлла, Брента и Квадратичный метод оказались ненадежными для подобных задач. Следующий пример демонстрирует этот факт.

Для минимизации мы выбрали следующую простую функцию

$$f(x, y) = x + 10y,$$

с ограничениями

$$y \leq 2x, \quad y \geq \frac{x}{2}.$$

Эта функция имеет один минимум $f_{\min} = 0$ в точке $x = 0, y = 0$. Оптимизация проводилась для следующей последовательности из 500 начальных точек:

$$(x_i = i, y_i = i), \quad i = 1, 2, \dots, 500.$$

На Рис. 10 показано число вычислений целевой функции для всех четырех методов. На Рис. 11 показана точность оценки минимума целевой функции.

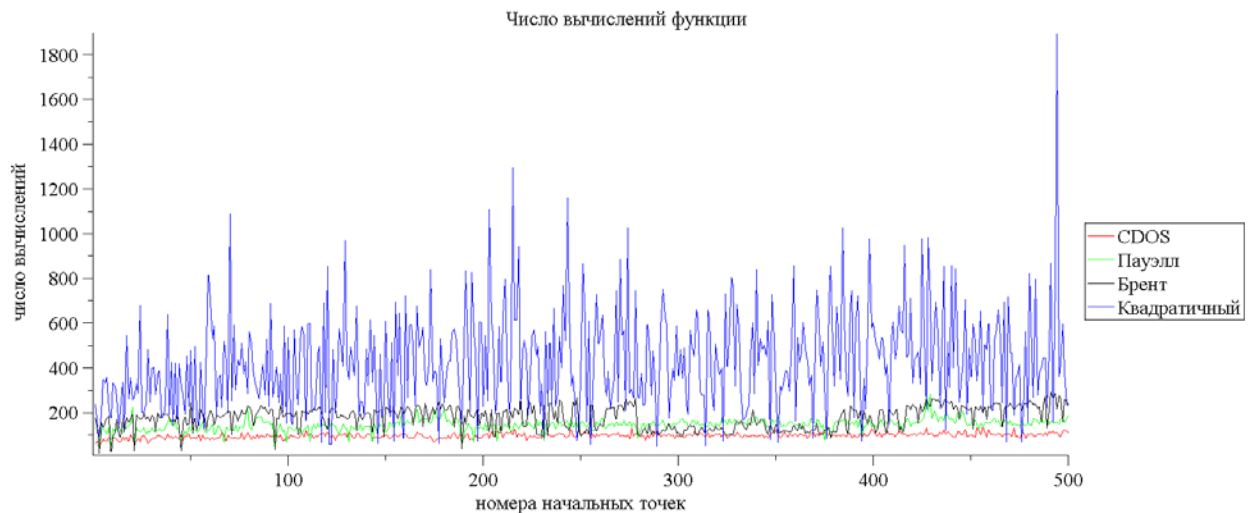


Рис. 10. Число вычислений функции

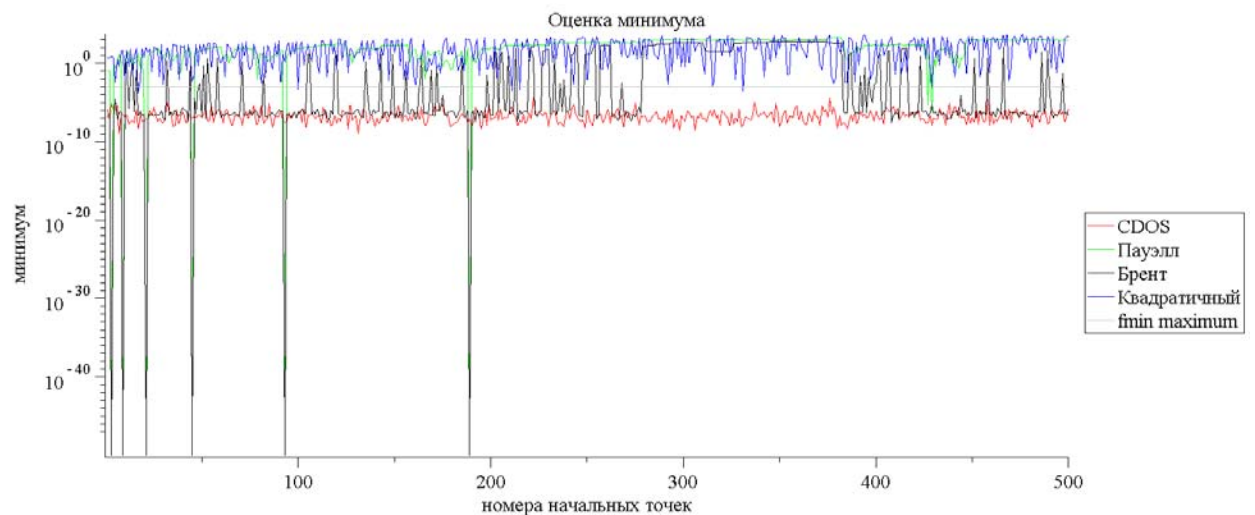


Рис. 11. Достигнутый минимум функции

В Таб. 3 приведены суммарные показатели эффективности оптимизации для всех методов применительно к задаче оптимизации с ограничениями.

Таб. 3. Суммарная эффективность методов оптимизации

Метод	CDOS	Пауэлл	Брент	Квадратичный
Среднее число вычислений функции	108	160	191	493
Медиана оценки минимума	$6 \cdot 10^{-9}$	202	10^{-7}	48.8
Надежность	100%	1.8%	62.6%	1.2%

Из Рис. 11 и Таб. 3 видно, что надежность Квадратичного метода и метода Пауэлла очень низкая для рассматриваемой функции. Надежность метода Брента выше, но явно недостаточна для такой простой задачи оптимизации. Метод CDOS показывает высокую надежность для задач оптимизации с ограничениями.

Заключение

Известно, что универсальных методов оптимизации, которые могли бы эффективно решать все классы оптимизационных задач, не существует. Мы назвали метод CDOS универсальным, т.к. он оказался быстрым и надежным для широкого класса оптимизационных задач. Он является быстрым и надежным не только для дифференцируемых функций, но также для недифференцируемых функций и для оптимизационных задач с ограничениями, когда целевая функция может быть не определена вне границ ограничений и, как целевая функция, так и ограничения, могут быть “черными ящиками”.

Ссылки

1. Powell, M. J. D., 1964, An efficient method for finding the minimum of a function of several variables without calculating derivatives, *Computer J.*, 7,155-162.
2. Химмельблау Д. Прикладное нелинейное программирование. “Мир”, Москва, 1975, 534 с.
3. Richard P. Brent, Algorithms for Minimization without Derivatives, Prentice-Hall, Englewood Cliffs, NJ, 1973, 195 p.
4. Численные методы условной оптимизации. Глава 7. / Редакторы Ф. Гилл и У. Мюррэй, Изд. Мир, Москва, 1977, 290 с.