*Информационные процессы, Том 14, № 3, 2014, стр. 256–274.*


**МАТЕМАТИЧЕСКИЕ МОДЕЛИ, ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ**

# Обзор методов многомерной оптимизации[1]

**Е.М.Захарова, И.К.Минашина**

*Московский физико-технический интститут (ГУ), Москва, Россия*

Поступила в редколлегию 25.06.2014

**Аннотация**—В статье рассмотрены основные методы многомерной оптимизации, проведено сравнение их эффективности, а также дан анализ применимости рассмотренных алгоритмов к различным типам оптимизируемых функций.

**КЛЮЧЕВЫЕ СЛОВА:** многомерная оптимизация, условная оптимизация, локальный экстремум, глобальный экстремум, многоэкстремальная целевая функция.

## 1. ВВЕДЕНИЕ

В процессе проектирования интеллектуальных систем управления часто возникает задача определения наилучших значений параметров или структуры объектов. Такая задача называется оптимизационной. Сегодня оптимизационные задачи и задачи принятия решений моделируются и решаются в самых различных областях техники [1]. К навыкам математического обоснования принятия решений относятся навыки математического моделирования оптимизационных задач, выбора адекватного математического обеспечения (метода, алгоритма, программной системы) с необходимым обоснованием, анализа полученных результатов и их интерпретации в терминах предметной области.

Задача оптимизации в целом сводится к задаче поиска экстремума (минимума или максимума) целевой функции с заданными ограничениями. Её математическая постановка выглядит следующим образом: необходимо определить значения вектора переменных $x = (x_1, x_2, \ldots, x_m)$, которые удовлетворяют ограничениям вида:

$$g_i(x_1, x_2, \ldots, x_m) \leq b_i \tag{1}$$

для всех $i = 1, \ldots, k$ и при которых достигается максимум или минимум целевой функции $f(x_1, x_2, \ldots, x_m)$:

$$f(x_1, x_2, \ldots, x_m) \to (\max, \min).$$

Допустимым решением задачи называется такое решение, которое удовлетворяет ее ограничениям (1). Совокупность допустимых решений задачи называют областью допустимых решений (ОДР)D. Окончательным решением задачи является пара $(x^*, f^*(x^*))$, состоящая из оптимального решения и оптимального значения целевой функции.

Методы математического программирования дают большое разнообразие алгоритмов решения данной задачи. В целом алгоритмы поиска реализуют методы спуска к экстремуму, при которых значение целевой функции последовательно улучшается вплоть до достижения экстремума. В зависимости от возможности нахождения алгоритмом локального либо глобального экстремума, они делятся на алгоритмы локального и глобального поиска.

[1] Работа выполнена при поддержке гранта офи_м_РЖД №13-01-13105.

## 2. ПОИСК ЛОКАЛЬНОГО ЭКСТРЕМУМА

Алгоритмы поиска локального экстремума предназначены для определения одного из локальных экстремумов на множестве допустимых решений, в котором целевая функция принимает максимальное или минимальное значение. При их построении могут использоваться как детерминированный спуск в область экстремума, так и случайный поиск. Среди детерминированных методов различают методы нулевого порядка и градиентные (1-го и 2-го порядка). Первые основаны на вычислениях только значений оптимизируемой функции. Вторые используют частные производные соответствующего порядка. Для поиска экстремума в случаях, когда вид оптимизируемой функции известен не полностью, либо ее структура слишком сложна, применяются методы стохастического программирования или нейронных сетей. Эффективность процедуры поиска оптимума – возможность отыскания решения и сходимость к решению по скорости зависят от вида функции и применяемого для нее метода. Рассмотрим стратегию каждого метода более подробно, исследуя для определенности минимизацию целевой функции.

### *2.1. Методы нулевого порядка (прямые методы)*

Из прямых методов наиболее известны методы:

- координатного спуска – поочередная оптимизация параметров вдоль осей одним из известных одномерных методов;
- спирального координатного спуска;
- вращающихся координат (метод Розенброка);
- поиска по симплексу;
- Хука–Дживса с поиском по образцу и др.

*Метод координатного спуска* заключается в том, что в качестве направлений траектории спуска от предыдущей точки поиска $X^{(k-1)}$ к последующей $X^{(k)}$ принимаются поочередно направления координатных осей $x_i (i = 1, 2, \ldots, n)$. После спуска на один шаг по координате $x_1$ происходит переход к спуску на один шаг по координате $x_2$, а затем движение вдоль координаты $x_3$ и т.д., пока не будет найдена следующая точка поиска $X^{(k)}$ с координатами $x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$. Движение по траектории спуска от предыдущей точки $X^{(k-1)}$ к последующей $X^{(k)}$ продолжается до тех пор, пока не будут достигнуты окрестности точки минимума $X^*$ целевой функции, определяемые точностью вычислений. Для поиска координат точки $X^{(k)}$ на каждом шаге итерации можно использовать любой из методов одномерной минимизации: метод золотого сечения, метод деления отрезка пополам, метод интерполяции-экстраполяции и др.

*Метод спирального координатного спуска* отличается от рассмотренного выше лишь тем, что шаг h меняется каждый раз при переходе от поиска минимума по одной переменной к поиску минимума по другой переменной. В трехмерном пространстве это напоминает спуск во впадину по спирали. Обычно этот метод дает некоторое сокращение времени поиска. Методы координатного спуска недостаточно эффективны для поверхностей с “оврагами”, так как в этом случае получение решения с требуемой точностью не гарантировано. Это вызвано тем, что в случае “оврага”, повернутого относительно осей, попытка продвижения в любом направлении может вызывать “ухудшение” целевой функции. В то же время продвижение вдоль “оврага” может давать “улучшение” целевой функции.

*Метод Розенброка.* Метод Розенброка направлен на устранение одного из недостатков метода покоординатного спуска – высокую чувствительность к выбору системы координат. В процессе поиска методом Розенброка производится поворот координатных осей так, чтобы одна из осей была направлена вдоль направления “оврага”. Рассмотрим алгоритм метода в случае

одномерной минимизации. На каждой итерации процедура осуществляет итеративный поиск вдоль n линейно независимых и ортогональный направлений. Когда получена новая точка в конце итерации, строится новое множество ортогональных векторов.

Построение направлений поиска:

Пусть $d_1, \ldots, d_n$ – линейно независимые векторы, по норме равные единицы. Предложим, что эти векторы взаимно ортогональны, т. е. $(d_i * d_j = 0)$ для $i \neq j$. Начиная из текущей точки $x^{(k)}$, целевая функция последовательно минимизируется вдоль каждого из направлений, в результате чего получается точка $x^{(k+1)}$. В частности, $x^{(k+1)} - x^{(k)} = \Sigma(\lambda_j * d_j)$, где $\lambda_j$ – длина шага по направлению $d_j$. Новый набор направлений $q_1, \ldots, q_n$ строится с помощью процедуры Грамма–Шмидта следующим образом:

$$a_j = \begin{cases} d_j, \text{ если } \lambda_j = 0 \\ \sum_{i=j}^{n} d_i \lambda_i, \text{ если } \lambda_j \neq 0 \end{cases}, \quad b_j = \begin{cases} a_j, \text{ если } j = 1 \\ a_j - \sum_{i=1}^{j-1} (a_j * q_i) * q_i, \text{ если } j \geq 2 \end{cases}, \quad q_j = \frac{b_j}{\|b_j\|}.$$

Новые направления, построенные описанным образом, являются линейно независимыми и ортогональными.

Таким образом, метод Розенброка позволяет избежать проблем, связанных с получением решения заданной точности для поверхностей с “оврагами”. Однако такой подход сравнительно увеличивает время поиска, что является относительным недостатком описанного метода.

*Метод поиска по симплексу.* Регулярный симплекс в $N$-мерном пространстве – это многогранник, образованный $N+1$ равноотстоящими точками – вершинами симплекса. Его важным свойством является возможность построения нового симплекса на любой грани исходного путём переноса выбранной вершины на некоторое расстояние вдоль прямой, соединяющей эту вершину с центром тяжести остальных вершин симплекса.

Работа алгоритма начинается с построения регулярного симплекса в пространстве независимых переменных задачи и оценивания значения целевой функции в его вершинах. Затем точка $x^{(j)}$ с наибольшим значением функции отражается через центр тяжести остальных точек:

$$x_c = \frac{1}{N} \sum\nolimits_{i \neq j, i=0}^{N} x^{(i)}.$$

Новая точка используется как вершина нового симплекса. Итерации продолжаются до тех пор, пока либо не будет накрыта точка минимума, либо не начнётся циклическое движение по двум или более симплексам. При этом следует пользоваться тремя правилами:

- Если точка с наибольшим значением функции получена на предыдущей итерации, то вместо неё берётся точка со следующим по величине значением функции.
- Если некоторая вершина симплекса не исключается более чем на $N$ итерациях, то уменьшить размеры симплекса с помощью некоторого коэффициента и построить новый симплекс, используя в качестве базовой точку с наименьшим значением функции.
- Поиск заканчивается, когда размеры симплекса и разности значений функции в вершинах станут достаточно малы.

Все точки прямой, проходящей через $x^{(j)}$ и $x_c$ определяются формулой:

$$x = x^{(j)} + \lambda * (x_c - x^{(j)}).$$

Достоинства метода:

- простота;

– малое количество заранее установленных параметров;
– алгоритм эффективен и при ошибках в определении значений целевой функции.

Недостатки метода:

– возникают трудности связанные с масштабированием задачи (в реальных задачах разные переменные часто не сопоставимы между собой по значениям);
– алгоритм работает медленно (не используется информация предыдущих итераций);
– не существует простого способа изменения размеров симплекса без пересчёта всех значений целевой функции.

*Метод Хука–Дживса.* Алгоритм поиска по симплексу можно усовершенствовать путём введения множества векторов, задающих направления поиска. Эти вектора должны быть линейно-независимы и образовывать базис в пространстве независимых переменных. Таким условиям удовлетворяет система координатных направлений.

Метод Хука–Дживса – это комбинация исследующего поиска по направлениям и поиска по образцу.

Исследующий поиск определяется следующим образом. Задаётся величина шага, которая может быть разной для разных координатных направлений и изменяться в процессе поиска. Если значение целевой функции в пробной точке не превышает значение в исходной, то шаг поиска рассматривается как успешный. В противном случае, необходимо вернуться в предыдущую точку и сделать шаг в противоположном направлении. После перебора всех $N$ координат исследующий поиск заканчивается. Полученная точка называется базовой.

Поиск по образцу заключается в реализации единственного шага из полученной базовой точки вдоль прямой, соединяющей её с предыдущей базовой точкой.

Новая точка строится по формуле:

$$x_p^{(k+1)} = x^{(k)} + \lambda(x^{(k)} - x^{(k-1)}),$$

где $x^{(k)}$ – текущая базовая точка; $x^{(k-1)}$ – предыдущая базовая точка; $x_p^{(k+1)}$ – точка, построенная при движении по образцу; $\lambda$ – параметр алгоритма.

Если движение по образцу не приводит к уменьшению целевой функции, то точка $x_p^{(k+1)}$ фиксируется в качестве временной базовой точки и вновь проводится исследующий поиск из этой точки. Если в результате получается точка со значением функции меньшим, чем в $x^{(k)}$, то она рассматривается как новая базовая точка $x^{(k+1)}$. Если исследующий поиск неудачен, то происходит возврат в $x(k)$ и проводится поиск в противоположном направлении. Если он также не приводит к успеху, то величина шага уменьшается и возобновляется исследующий поиск. Поиск завершается, когда величина шага становится достаточно малой.

Достоинствами данного метода являются простая стратегия поиска и небольшой объём требуемой памяти. Однако алгоритм основан на циклическом движении по координатам, что может привести к вырождению алгоритма в бесконечную последовательность исследующих поисков без поиска по образцу.

### *2.2. Градиентные методы*

Итерационные процессы оптимизации, направление поиска которых на каждом шаге совпадает с антиградиентом функции, называются градиентными методами [2].

*Алгоритм наискорейшего спуска* реализует итерационную процедуру движения к минимуму из произвольно выбранной начальной точки в направлении наиболее сильного уменьшения функции, определенном в окрестности текущего значения аргумента минимизируемой

функции. Такое направление противоположно направлению, задаваемому вектором градиента $\nabla f(x)$ минимизируемой функции $f(x)$. Общая формула для нахождения значения аргумента $x^{(k+1)}$ по значению $x^{(k)}$, найденному на $k$-м шаге работы алгоритма наискорейшего спуска:

$$x^{(k+1)} = x^{(k)} + \lambda^{(k)} * s^{(k)}, \tag{2}$$

где $s^k$ – вектор единичной длины в направлении, противоположном направлению градиента $\nabla f(x^{(k)})$, определенном в точке $x^{(k)}$:

$$s^{(k)} = \frac{-\nabla f(x^{(k)})}{\left\|\nabla f(x^{(k)})\right\|}, \tag{3}$$

где $\left\|\nabla f(x^{(k)})\right\|$ – норма вектора градиента $\nabla f(x^{(k)})$, $\lambda^{(k)}$ – шаг градиентной процедуры.

Алгоритмы наискорейшего спуска различаются по способу определения шага $\lambda^{(k)}$. Если шаг $\lambda^{(k)}$ не зависит от $k$ (является постоянным), то в окрестности экстремума будут наблюдаться незатухающие колебания, амплитуда которых зависит от величины $\lambda$ и от формы минимизируемой функции. Использование постоянного шага:

- позволяет построить наиболее простой вариант алгоритма;
- при больших значениях $\lambda$ обеспечивает быстрое движение к экстремуму, но приводит к заметным колебаниям в его окрестности;
- при малых значениях $\lambda$ приводит к низкой скорости сходимости к экстремуму;
- информация о приемлемой величине шага $\lambda$ может быть получена только в ходе отладки алгоритма, т.к. никакая информация о свойствах минимизируемой функции не используется.

Улучшением метода с постоянным шагом, позволяющим устранить колебания в окрестности экстремума без существенного усложнения алгоритма, является использование шага $\lambda$, величина которого убывает по ходу итерационного процесса и зависит от номера итерации k. Наряду с достоинствами такой подход имеет и недостаток, а именно несвязанность значений $\lambda^{(k)}$ с формой минимизируемой функции. Если вдали от экстремума функция f(x) имеет малый градиент, скорость сходимости может оказаться недопустимо медленной. Эта проблема решается путем модификации алгоритма, которая может быть осуществлена следующими способами:

*Алгоритм наискорейшего спуска с шагом, длина которого зависит от свойств минимизируемой функции (использование производной по направлению)* Пусть найдена точка $x^{(k)}$ и в этой точке определено направление движения к минимуму, задаваемое вектором единичной длины $s^{(k)}$ (3). Тогда из (2) видно, что положение следующей точки $x(k+1)$ и значение функции $f(x^{(k+1)})$ в этой точке зависят только от единственной скалярной переменной $\lambda^{(k)}$. Тогда возникает идея не изменять направления до тех пор, пока функция $f(x)$ в направлении $s^{(k)}$ убывает. Точка $x^{(k+1)}$ соответствует минимуму $f(x)$ по направлению $s^{(k)}$. Тогда в точке $x^{(k+1)}$ должно быть определено новое направление движения к минимуму $s^{(k+1)}$, которое будет ортогональным предыдущему $s^{(k)}$. Значение определяется из условия минимума квадратичной аппроксимации $f(x^{(k+1)})$ по $\lambda^{(k)}$:

$$\lambda^{(k)} = \frac{-[\nabla f(x^k)]^T * s^k}{[s^k]^T \nabla^2 f(x^k) * s^k},$$

где $\nabla^2 f(x^k)$ – матрица вторых производных.

Итерационный процесс заканчивается при выполнении одного из следующих условий:

– если в задаче интересно значение функции в точке оптимума (а не значение аргумента $x$), то следует прекратить вычисления, если, начиная с $k^*$-той итерации, для которой $\forall k \neq k^*$ абсолютное значение нормированной разности между значениями функции в соседних точках не превышает наперед заданного малого числа $\epsilon > 0$:

$$\frac{\left|f(x^{(k+1)}) - f(x^{(k)})\right|}{f(x^{(k)})} \leq \epsilon, \forall k \geq k^*.$$

Сходимость такого типа называется сходимостью по функционалу;

– если же по постановке задачи необходимо определить именно значение аргумента (а не функции), то следует прекратить вычисления, если, начиная с $k^{**}$ – той итерации, для которой $\forall k \geq k^*$ абсолютное значение нормированной разности между значениями аргумента в "соседних" точках не превышает наперед заданного малого числа $\delta > 0$:

$$\frac{\left|x^{(k+1)} - x^{(k)}\right|}{x^{(k)}} \leq \delta, \forall k \geq k^*.$$

Сходимость такого типа называется сходимостью по параметрам.

При использовании обоих правил остановки необходимы страховочные меры для предотвращения слишком длительных вычислений (например, прерывание вычислений при $k > k^{***}$, где $k^{***}$ – число порядка $10^5$).

*Алгоритм наискорейшего спуска с шагом, длина которого зависит от свойств минимизируемой функции (использование вторых производных). Метод Ньютона* Метод Ньютона основан на квадратичной аппроксимации минимизируемой функции в окрестности точки $x^{(k)}$. Минимум квадратичной функции легко найти, приравнивая ее градиент нулю. Можно сразу же вычислить положение экстремума и выбрать его в качестве следующего приближения к точке минимума. Вычисляя точку нового приближения по формуле: $x^{k+1} = x^k + \Delta x^k$ и разлагая $f(x^{(k+1)})$ в ряд Тейлора, получим формулу квадратичной аппроксимации $f_{\text{кв}}(x^{(k+1)})$:

$$f_{\text{кв}}(x^{(k+1)}) = f(x^{(k)}) + [\bigtriangledown f(x^{(k)})]^T x^k + \frac{1}{2!}[x^k]^T * \bigtriangledown^2 f(x^k) * x^k.$$

Из условия минимума $f_{\text{кв}}(x^{(k+1)})$ по $x^k$ находим длину шага $x^k$:

$$x^k = -[\bigtriangledown^2 f(x^k)]^{-1} * \bigtriangledown f(x^{(k)})$$

Достоинства метода Ньютона:

– если минимизируемая функция является квадратичной, то метод позволит найти минимум за один шаг;
– если минимизируемая функция относится к классу поверхностей вращения (т.е. обладает симметрией), то метод также обеспечивает сходимость за один шаг;
– если функция несимметрична, то метод не обеспечивает сходимость за конечное число шагов. Но для многих функций достигается гораздо более высокая скорость сходимости, чем при использовании других модификаций метода наискорейшего спуска.

Недостатки метода Ньютона связаны с необходимостью вычислений и обращения матриц вторых производных. При этом не только расходуется машинное время, сколько могут появиться значительные вычислительные погрешности, если матрица $\bigtriangledown^2 f(x^k)$ окажется плохо обусловленной.

### *2.3. Метод стохастической оптимизации*

Для учета неопределенности в оптимизационных моделях используют методы стохастического программирования, использующие знание распределений вероятностей для данных или их оценок [3,4]. В широком смысле методом стохастической оптимизации называется последовательный способ улучшения оценки минимизирующей функционал среднего риска, а именно, решающей задачу поиска минимума функции средних потерь:

$$f(x) = E_\omega F(\omega, x) = \int F(\omega, x) P_\omega(d\omega),$$

где $\omega_n$ – последовательность p-мерных случайных векторов из $R_p$, порожденная распределением вероятностей $P_\omega(\cdot)$, $F(\omega, x)$ – функция потерь. *Процедура Кифера–Вольфовица* В случае если вид оптимизируемой функции известен не полностью, либо если на вычисление ее значений необходимо чрезмерное количество усилий, можно воспользоваться только зашумленной информацией о значениях функции $F(\omega, x)$ в выбираемых точках $X$ с неконтролируемыми при этом значениями случайной величины $\omega$ [3,5].

Дж. Кифер с Дж. Вольфовицем в одномерном случае ($r = 1$) и Дж. Блюм в многомерном случае для построения оценок $\hat{x^n}$ предложили использовать процедуру следующего вида:

$$\hat{x}^n = \hat{x}^{n-1} - \alpha_n \frac{Y_+^n - Y_-^n}{2\beta_n},$$

где обозначено:

$$Y_\pm^n = \begin{pmatrix} F(\omega^{\frac{2r(n-1)+(3\pm1)}{2}}, \hat{x^{n-1}} \pm \beta_n e^1) \\ F(\omega^{\frac{2r(n-1)+(7\pm1)}{2}}, \hat{x^{n-1}} \pm \beta_n e^2) \\ \vdots \\ F(\omega^{\frac{2rn-(1\pm1)}{2}}, \hat{x^{n-1}} \pm \beta_n e^r) \end{pmatrix}.$$

Они обосновали, что последовательность оценок $\hat{x}^n$ сходится к точке минимума функции потерь при определенных условиях на свойства числовых последовательностей $\alpha_n$ и $\beta_n$, функции $F(\cdot, \cdot)$ и распределения соответствующих случайных величин [5,6]. Из накладываемых условий обычно следует, что в среднем по всевозможным реализациям $\omega$ значение $\dfrac{Y_+^n - Y_-^n}{2\beta_n}$ совпадает со значением градиента функции $f(\cdot)$ в точке $\hat{x}^{n-1}$.

### *2.4. Сеть Хопфилда*

В случаях поиска экстремума целевой функции часто возникают ситуации, когда становятся важны следующие две особенности: во-первых, решение должно быть адаптивным, то есть учитывать текущее состояние сети связи и наличие сбойных участков, а во-вторых, найти оптимальное решение (в нашем случае экстремум целевой функции) нужно очень быстро, в реальном времени. Для решения подобного рода задач прекрасно приспособлены сети Хопфилда [7,8].

Нейронная сеть Хопфилда (Рисунок 1) представляет собой слой адаптивных сумматоров с обратными связями, выходные сигналы которых, подвергаясь нелинейной обработке по заданному закону, поступают с некоторой временной задержкой на входы нейронов, в результате чего выходной сигнал нейронной сети формируется лишь после того, как сеть достигнет динамического равновесия.

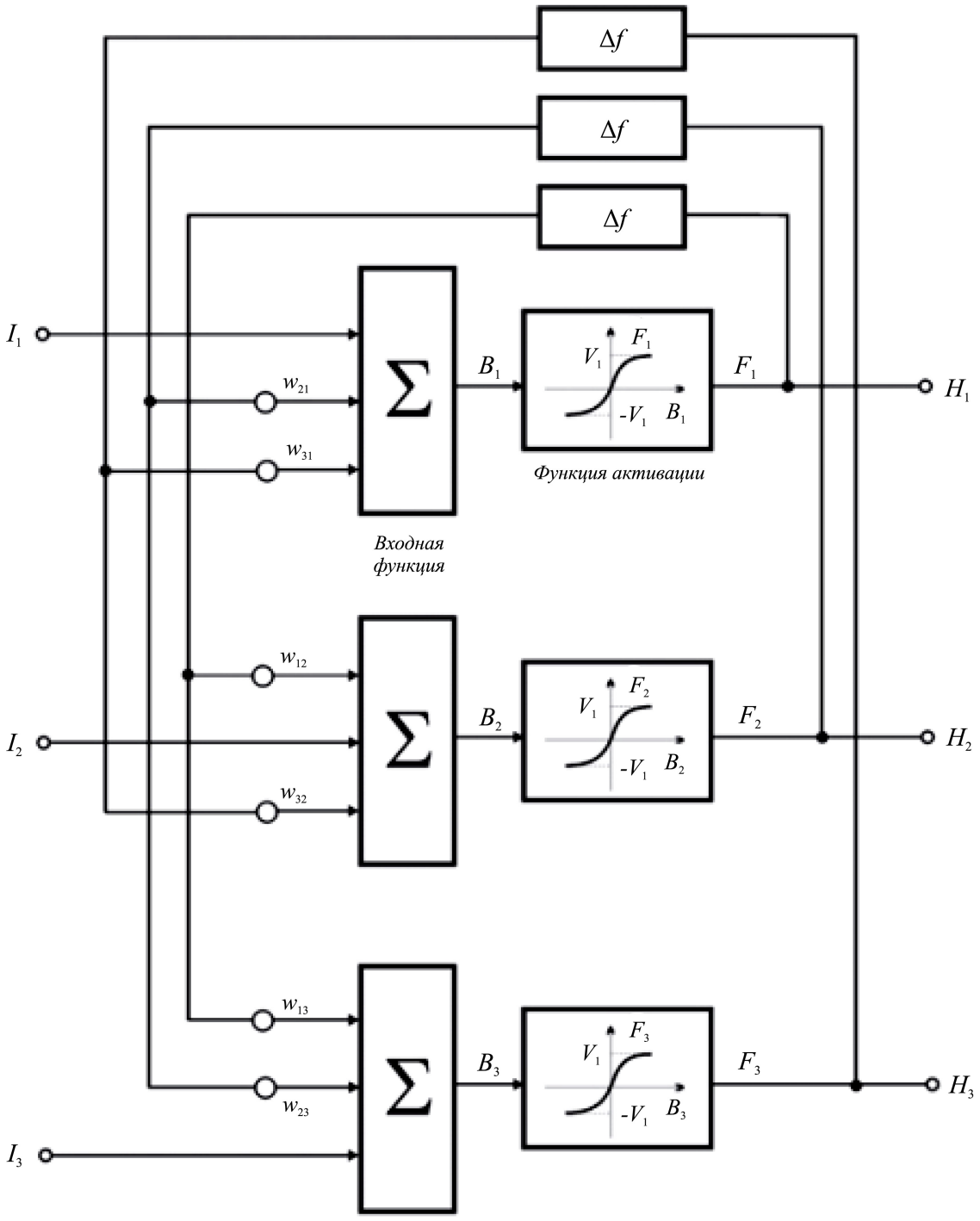


**Рис. 1.** Общая схема генетического алгоритма

Поведение нейронной сети моделирует, таким образом, некоторый стохастический процесс, конечное состояние которого определяется входным вектором нейросети, являющимся, по сути, вектором внешних смещений.

Пусть состояние каждого $i$-го нейрона определяется его выходным сигналом $F_i$. В архитектуре нейросети, реализующей бинарные операции, функция $F_i$ может принимать значения $F_i^0 = 0$ или $F_i^1 = 1$. Как видно из рисунка 1, выходной сигнал каждого нейрона $H_i$ представляет собой суперпозицию двух сигналов: внешнего сигнала $I_i$ и сигнала обратной связи, в виде суммы выходных сигналов других нейронов. Тогда:

$$H_i = \sum_{j=1}^{J} \omega_{ij} F_j + I_j, \tag{4}$$

где $\omega_{ij}$ – вес синаптической связи, соединяющей $j$-й нейрон с $i$-м нейроном, и $\omega_{ij} = 0$, если $i = j$.

Каждый нейрон изменяет свое состояние в зависимости от заданного уровня активации $S_i$, так что

$$F_i = \begin{cases} \downarrow F_i^0, H_i < S_i \\ \uparrow F_i^1, H_i > S_i \end{cases} \tag{5}$$

Если предположить, что весовые коэффициенты синаптических связей $\omega_{ij}$ являются фиксированными для всех $i$ и $j$, то система уравнений (4)–(5) определяет стохастический процесс, который достигает устойчивых положений равновесия в зависимости от внешних значений $I_i$. Доказано, что сходимость гарантирована, если ее матрица весовых коэффициентов $W$ является симметричной и все диагональные элементы равны нулю. Доказательство сходимости может быть получено из анализа "энергетической" функции нейросистемы, а именно, функции Ляпунова , которая для рассматриваемой нейронной сети с обратными связями имеет вид

$$E = -\frac{1}{2} \sum_{i=1}^{I} \sum_{j=1}^{J} \omega_{ij} F_i F_j - \sum_{i=1}^{I} I_i F_i + \sum_{i=1}^{I} S_i F_i,$$

и представляет собой квадратичный функционал состояния нейронной сети. Изменение функции E вследствие изменения состояния i-го нейрона на $\triangle F_i$ представляется в виде

$$\triangle E = -[\sum_{j=1}^{J} \omega_{ij} F_j + I_i + S_i] \triangle F_i. \tag{6}$$

Из уравнения (6)) следует, что величина $\triangle F_i$ принимает положительные значения только в том случае, когда $\sum_{j=1}^{J} \omega_{ij} F_j + I_i + S_i > 0$ и наоборот, принимает отрицательные значения, если $\sum_{j=1}^{J} \omega_{ij} F_j + I_i + S_i < 0$.

Следовательно, произвольное изменение состояния нейрона в архитектуре нейросети Хопфилда приводит к уменьшению энергетической функции всей системы.

Поведение нейронной сети Хопфилда можно анализировать, используя графовую модель либо решая задачу о собственных векторах и собственных значениях рассматриваемой системы. С точки зрения графовой модели, нейронная сеть представляет собой направленный граф, вершины которого образуют нейроны с приложенными к ним внешними смещениями, а ребра – синаптическими связями с весовыми коэффициентами $\omega_{ij}$. Доказательство сходимости алгоритма в этом случае сводится к задаче нахождения минимального разреза графа.

Сеть Хопфилда, нейроны которой обладают непрерывной монотонной активацией, т. е. $F_i^0 \le F_i \le F_i^1$ также обладает свойствами процессора, производящего минимизацию целевой функции.

Динамическая процедура, описанная выше, на каждом шаге понижает значение энергии нейронной сети. Это позволяет решать комбинаторные задачи оптимизации, если они могут быть сформулированы как задачи минимизации энергии.

## 3. ПОИСК ГЛОБАЛЬНОГО ЭКСТРЕМУМА

Задача поиска глобального экстремума функции на допустимом множестве $X$ состоит в поиске точки $x_* \in X$, при которой выполняется $f(x_*) \le f(x), f(x_*) \ge f(x)$, для всех $x \in X$. Ограничения, связанные с вычислительной погрешностью и другими факторами, часто не позволяют найти точное решение задачи. В этом случае имеет место поиск приближенного решения, т.е. точки из множества $\epsilon$-оптимальных решений $X_*^\epsilon = \{x \in X : f(x) \le f(x_*)+\epsilon\}$. Поиск точного решения можно рассматривать как частный случай поиска приближенного решения с $\epsilon = 0$. Алгоритмы поиска глобального экстремума делятся на детерминированные, статистические и комбинированные. Часто задача глобальной оптимизации сводится к задаче поиска локальных экстремумов и нахождению среди них глобального оптимума, используя, таким образом, методы поиска локального экстремума. Рассмотрим подробнее каждую из стратегий поиска глобального экстремума на основе конкретных алгоритмов.

### *3.1. Детерминированные методы*

Детерминированные методы находят глобальное решение посредством поиска на всем допустимом множестве.

#### Алгоритм Гомори или метод отсекающих плоскостей

Методы отсечений, или отсекающих плоскостей являются альтернативной основой для построения как точных, так и приближенных алгоритмов решения задач целочисленного программирования [9]. В настоящее время доказана их высокая эффективность в сочетании с методами ветвей и границ. Такие гибридные вычислительные схемы носят общее название метода ветвей и отсечений. Все эти методы реализуют общую вычислительную стратегию, заключающуюся в решении последовательности ослабленных (релаксированных) подзадач линейного программирования. В методе отсечений релаксированные подзадачи постепенно улучшают аппроксимацию данной целочисленной задачи, уменьшая окрестность оптимального решения. В тяжелых случаях оптимальность может не быть получена, или доказаны за приемлемое число шагов, однако методы отсечений даже в этом случае позволяют найти приближенное целочисленное решение с заданной погрешностью (в заданной окрестности оптимального решения). Алгоритмы метода отсекающих плоскостей могут быть эффективно использованы при решении как частных задач целочисленного программирования, включая задачу коммивояжера, задачу о максимальном разрезе, задачу о ранце, так и общей задачи целочисленного линейного программирования. По сравнению с методами ветвей и границ, методы отсекающих плоскостей являются более технологичными при программировании, т.к. не требуют дополнительного объема оперативной памяти неопределенного размера для хранения дерева решений, но в тоже время – менее универсальными, т.к. не умеют работать с релаксированными подзадачами, являющимися задачами выпуклого программирования. Первый алгоритм отсекающих плоскостей был предложен Р.Е. Гомори в 1958 году. В настоящее время эти методы эффективно используются для решения различных задач, включая задачу коммивояжера. Недавние

исследования также показали, что отсечения, предложенные Гомори вполне актуальны в настоящее время. Кроме этого сейчас используются и другие типы отсекающих плоскостей для решения общей задачи целочисленного программирования. Рассмотрим следующую задачу целочисленного линейного программирования:

$$f(x) = (c, x) \to \max \tag{7}$$

$$Ax = b \tag{8}$$

$$x \geq 0 \tag{9}$$

$$x \equiv (mod1), \tag{10}$$

где $c, b$ – целочисленные коэффициенты, $A$ – целочисленная матрица. Предположим, что задача имеет решение и $x_R^* \neq x_Z^*$. Систему (8) представим в виде:

$$x_B = B^{(-1)}b - B^{(-1)}Nx_N, \tag{11}$$

где $B$ – оптимальный базис задачи линейного программирования (7)–(10); $x_B, x_N$ – векторы базисных и небазисных переменных, $N = A \setminus B$. Введем следующие обозначения: $\overline{b} = B^{(-1)}b$; $A = B^{(-1)}N$; $J_B, J_N$ – множества номеров базисных и небазисных переменных. Запишем систему (11) в координатной форме:

$$x_i = \overline{b}_i - \sum_{j \in J_N} \overline{\alpha_{ij}} x_j, i \in J_B \tag{12}$$

чтобы все $x_i, i \in J_B$. В были целыми числами, должно выполняться условие: $\overline{b}_i - \sum_( j \in J_N)\overline{\alpha}_{ij}x_j$, которое эквивалентное условию:

$$\{\overline{b}_i\} - \sum_{j \in J_N} \{\overline{\alpha}_{ij}\}x_j \equiv 0(mod1), \quad i \in J_B, \tag{13}$$

где $\{\ \}$ – дробная часть числа. Соотношение (13) является необходимым и достаточным условием целочисленности вектора $x_B$, но ему трудно удовлетворить. Заменим (13) более простым, но уже только необходимым условием:

$$\sum_{j \in J_N} \{\overline{\alpha}_{ij}\}x_j \leq -\{\overline{b}_i\}, \quad i \in J_B. \tag{14}$$

Если $\{\overline{b}_i\} > 0$, то оптимальное непрерывное решение $x_N^* = 0, x_B^* = \overline{b}$ не удовлетворяет соответствующему неравенству (14), т.е. неравенство является отсечением. Это правильное (по построению) отсечение называется отсечением Гомори. Самое сильное отсечение, чаще всего, соответствует $\max_{i \in J_N}\{\overline{b}_i\}$. Если гарантирована целочисленность целевой функции (например, когда $c$ – целочисленный вектор), то аналогичное (14) неравенство можно получить, выразив целевую функцию через небазисные переменные. Достоинством данного метода является то, что любая линейность в исходной постановке задачи сохраняется. Данный алгоритм достаточно эффективен для решения определённого класса задач – геометрического программирования. Его основной недостаток – требование выпуклости допустимой области и рост размерности задачи линейного программирования от итерации к итерации.

**Интервальный метод ветвей и границ**

Основным интервальным методом поиска глобального экстремума многомерной функции является метод ветвей и границ [10]. Метод начинает работу с определения нижней и верхней границ для исходной задачи. Если верхняя и нижняя границы совпадают, то полученный результат является оптимальным значением, и метод прекращает работу. Иначе, множество переменных разбивается на несколько собственных подмножеств, объединение которых совпадает с исходным множеством. Эти подзадачи становятся потомками исходной. Далее алгоритм применяется рекурсивно к каждой из подзадач, создавая дерево подзадач. Если оптимальное решение найдено для некоторой подзадачи, то оно является достижимым для исходной задачи (не обязательно оптимальным), но так как оно достижимо, его можно использовать для обрезания ветвей у исходного дерева. Процесс поиска продолжается до тех пор, пока каждая из подзадач не будет решена или выкинута или до тех пор, пока не будет достигнут заданный порог между лучшим из найденных решений и нижней границей $f(x)$ для всех нерешенных задач. Описание алгоритма:

1. Нулевой шаг – вычисление нижней границы $\xi(G)$ множества решений $G : \xi(G) = \xi(G(0))$.
2. Если при этом удается найти такой план $x_0$, что $f(x_0) = \xi(G(0))$, то $x_0$ – оптимальный план. Если оптимальный план $x_0$ не найден, то некоторым способом разбивают множество G(0) на конечное число непересекающихся подмножеств $G_1^{(}(1)), G_2^{(}(1)), \ldots, G_{(r_1)}^{(}(1))$, таких что $G^0 = G_1^{(}(1)) \bigcup G_2^{(}(1)) \bigcup \ldots \bigcup G_{(r_1)}^{(}(1))$ и переходят к первой итерации.
3. Первая итерация – вычисляют оценки $\xi(G_1^{(}(1)))$при $i = 1, 2, \ldots, r_1$. Если удается найти такой план $x_0$, что $x_0 \in G_\rho^{(}(1))(1 \le \rho \le r_1)$ и $f(x_0) = \xi(G_{rho}^{(}(1))) \le \xi(G_i^{(}(1)))$ при $i = 1, 2, \ldots, r_1$, то $x_0$ – оптимальный план. В противном случае для дальнейшего разбиения выбирают наиболее перспективное множество $G_\gamma^{(}(1))$ по следующему правилу:

$$\xi(G_\gamma^{(}(1))) = \min_i \xi(G_i^{(}(1))).$$

4. Разбивают множество $\xi(G_\gamma^{(}(1)))$ на несколько подмножеств $\xi(G_\gamma^{(}(1))) = \bigcup_i^{r_1} c\xi(G_{\gamma i}^{(}(1)))$. Еще не подвергавшиеся разбиению множества переобозначают $G_1^{(}(2)), G_2^{(}(2)), \ldots, G_{(r_2)}^{(}(2))$ и переходят ко второй итерации.
5. $k$-ая итерация – вычисляют оценки $\xi(G_i^{(}(k)))$ при $i = 1, 2, \ldots, r_1$. Если удается найти такой план $x_0$, для которого $f(x_0) = \xi(G_\rho^{(}(k))) \le \xi(G_i^{(}(k)))$ для всех $i = 1, 2, \ldots, r_k$, то $x_0$ – оптимальный план. Если же оптимальный план не найден, то снова выбирают наиболее перспективное множество $G_\gamma^{(}(k))$ такое, что $\xi(G_\gamma^{(}(k))) = \min_i \xi(G_i^{(}(k)))$.

Далее разбивают множество $G_\gamma^{(}(k))$ на несколько подмножеств $\xi(G_\gamma^{(}(k))) = \min_i \xi(G_i^{(}(k)))$ и переходят к $(k+1)$-й итерации. Суть работы метода Хансена заключается в последовательном удалении из начальной области подобластей, в которых не содержится глобальный минимум. Удаление происходит одним из ниже перечисленных способов:

– Удаляются подобласти, в которых градиент $g$ функции $f$ отличен от нуля.
– Удаляются подобласти, в которых $f > \overline{f}$, где $\overline{f}$ – верхняя оценка глобального минимума.
– Удаляются подобласти, в которых функция $f$ невыпукла.

Главный недостаток метода ветвей и границ заключается в необходимости полностью решать задачи линейного программирования, ассоциированные со всей областью допустимых решений. Для задач большой размерности это требует значительных и неоправданных затрат времени. Однако данный метод являются наиболее надежным средством решения целочисленных задач.

### *3.2. Стохастические методы*

Большинство детерминированных алгоритмов теряют эффективность с возрастанием размерности задачи. Стохастические методы позволяют уйти от проблем детерминированных алгоритмов. Стохастический подход присутствует не только в разработке алгоритма, но и, например, при определении условия остановки. К числу стохастических методов глобальной оптимизации относят алгоритм имитации отжига, генетические алгоритмы, эволюционные и поведенческие стратегии, метод виртуальных частиц, алгоритм контролируемого случайного поиска.

#### Эволюционные или генетические алгоритмы

Генетические алгоритмы – класс методов оптимизации, основанных на имитации процессов протекающих в природе, в частности естественного отбора – концепции, озвученной в эволюционной теории Чарльза Дарвина [11]. В соответствие с теорией Дарвина в естественной среде преимущество к выживанию и размножению имеют особи, более приспособленные к условиям конкретной среды обитания. Основным материалом для естественного отбора служат естественные мутации генов и их комбинации, получаемые при размножении. В ходе естественного отбора выживают особи с наибольшей функцией приспособленности – численной характеристикой, определяющейся в соответствии с конкретной задачей. Наиболее приспособленные особи получают возможность скрещиваться (кроссовер) и давать потомство. После этого на получившуюся популяцию оказывают влияния случайные мутации. Общая схема "классического"генетического алгоритма приведена на рис. 2

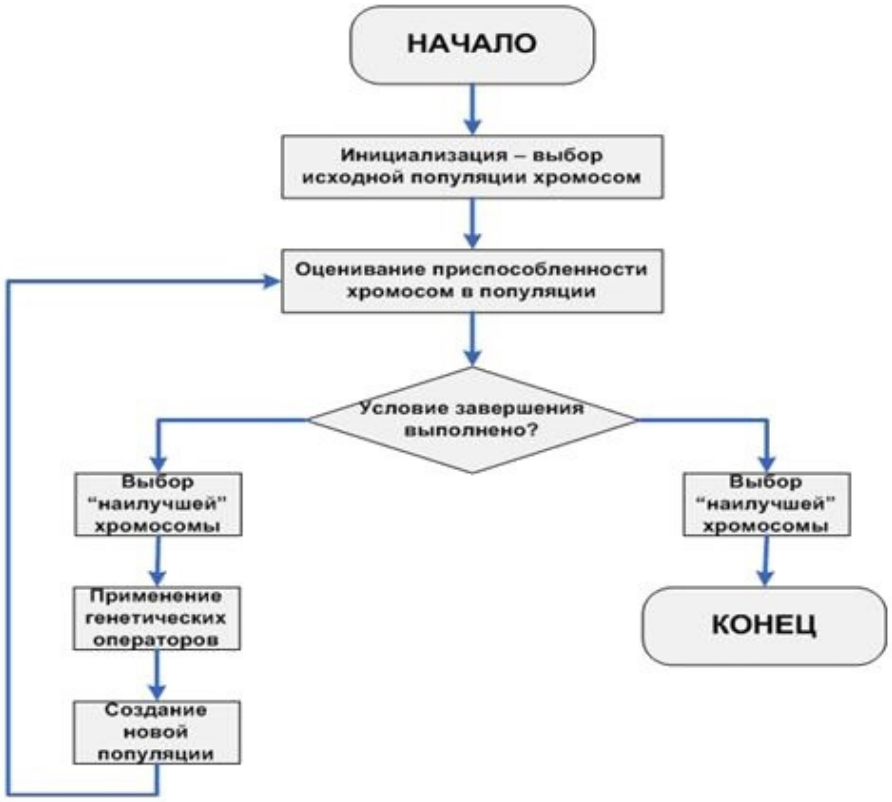


**Рис. 2.** Общая схема генетического алгоритма

Переформулируем задачу оптимизации как задачу нахождения максимума некоторой функции $f(x_1, x_2, \ldots, x_n)$, называемой функцией приспособленности. Необходимо, чтобы $f(x_1, x_2, \ldots, x_n) \geq 0$ на ограниченной области определения, при этом совершенно не требуются непрерывность и дифференцируемость. Каждый параметр функции приспособленности кодируется строкой битов. Особью будет называться строка, являющаяся конкатенацией строк упорядоченного набора параметров:

$$\begin{array}{c} 1010\ 10110\ 101\ \ldots\ 10101 \\ |\ x_1\ |\ \ x_2\ |\ \ x_3\ |\ \ldots\ |\ \ x_n\ | \end{array}$$

Универсальность ГА заключается в том, что от конкретной задачи зависят только такие параметры, как функция приспособленности и кодирование решений. Остальные шаги для всех задач производятся одинаково. С помощью функции приспособленности среди всех особей популяции выделяют:

- наиболее приспособленные (более подходящие решения), которые получают возможность скрещиваться и давать потомство
- наихудшие (плохие решения), которые удаляются из популяции

Таким образом, приспособленность нового поколения в среднем выше предыдущего. В классическом ГА:

- начальная популяция формируется случайным образом
- размер популяции (количество особей $N$) фиксируется и не изменяется в течение работы всего алгоритма
- каждая особь генерируется как случайная $L$-битная строка, где $L$ – длина кодировки особи
- длина кодировки для всех особей одинакова.

Шаг алгоритма состоит из трех стадий:

1. генерация промежуточной популяции путем отбора текущего поколения. Промежуточная популяция – это набор особей, получивших право размножаться. Наиболее приспособленные особи могут быть записаны туда несколько раз, наименее приспособленные с большой вероятностью туда вообще не попадут. В классическом ГА вероятность каждой особи попасть в промежуточную популяцию пропорциональна ее приспособленности, т.е. работает пропорциональный отбор.
2. скрещивание особей промежуточной популяции путем кроссовера, что приводит к формированию нового поколения. Особи промежуточной популяции случайным образом разбиваются на пары, потом с некоторой вероятностью скрещиваются, в результате чего получаются два потомка, которые записываются в новое поколение, или не скрещиваются, тогда в новое поколение записывается сама пара. В классическом ГА применяется одноточечный оператор кроссовера: для родительских строк случайным образом выбирается точка раздела, потомки получаются путём обмена отсечёнными частями.
3. мутация нового поколения. Оператор мутации, необходим для “выбивания” популяции из локального экстремума и способствующий защите от преждевременной сходимости. Каждый бит каждой особи популяции с некоторой вероятностью инвертируется. Эта вероятность обычно очень мала, менее 1%. Можно выбирать некоторое количество точек в хромосоме для инверсии, причем их число также может быть случайным. Также можно инвертировать сразу некоторую группу подряд идущих точек.

Такой процесс эволюции, вообще говоря, может продолжаться до бесконечности. Критерием останова может служить заданное количество поколений или схождение популяции. Схождением называется состояние популяции, когда все строки популяции находятся в области некоторого экстремума и почти одинаковы. То есть кроссовер практически никак не изменяет популяции, а мутирующие особи склонны вымирать, так как менее приспособлены. Таким образом, схождение популяции означает, что достигнуто решение, близкое к оптимальному. Итоговым решением задачи служит наиболее приспособленная особь последнего поколения с наибольшей функцией приспособленности. Одним из недостатков генетического алгоритма является отсутствие гарантий обнаружения глобального решения за приемлемое время. Также генетический алгоритм не гарантирует, что найденное решение будет оптимальным решением. Тем не менее, они применимы для поиска "достаточно хорошего" решения задачи за "достаточно короткое время". Однако данный алгоритм имеет преимущества перед другими алгоритмами при очень больших размерностях задач и отсутствия упорядоченности в исходных данных, когда альтернативой им является метод полного перебора вариантов. Главным преимуществом генетического алгоритма является то, что он может применяться для решения сложных неформализованных задач, для которых не разработано специальных методов, т.е. обеспечиваются решение нестандартных проблем.

### Метод отжига

Метод отжига – это метод оптимизации, использующая упорядоченный случайный поиск на основе аналогии с процессом образования в веществе кристаллической структуры с минимальной энергией при охлаждении. Преимуществом метода отжига является возможность избежать локальные минимумы оптимизируемой функции за счет принятия изменений, временно ухудшающих результат, что отражает суть процесса нагрева расплава для предотвращения его быстрого остывания. Еще одним преимущество – даже в условиях нехватки вычислительных ресурсов для нахождения глобального минимума метод отжига выдает один из локальных минимумов. Основной задачей имитации отжига является поиск глобального экстремума целевой функции, которая характеризует какую-либо систему. Этот метод является важным инструментом решения невыпуклых задач оптимизации. Метод отжига отличается от итеративных алгоритмов адаптивностью. Основные свойства конечного состояния системы уже проявляются при высоких температурах, при низких же происходит уточнение их состояния [12]. Метод отжига является одним из алгоритмов поиска глобального экстремума целевой функции $f(x)$, заданной для x из некоторого пространства $S$, дискретного или непрерывного. Элементы множества $S$ представляют собой состояния воображаемой физической системы ("энергетические уровни"). А значение функции $f$ в этих точках используется как энергия системы $E = f(x)$, в каждый момент предполагается заданной температура системы. Находясь в состоянии при температуре $T$, следующее состояние системы выбирается в соответствие с заданным порождающим семейством вероятностных распределений $Q(x, T)$, которое и задает новый случайный элемент $x^l = G(x, T)$. После генерации $x^l$ система с вероятностью $h(\triangle E; T)$ переходит к следующему шагу в состояние $x^l$. Если переход не произошел, процесс генерации $x^l$ повторяется. Здесь $\triangle E$ обозначает приращение функции энергии $f(x^l) - f(x)$. Величина $h(\triangle E; T)$ называется вероятностью принятия нового состояния. То есть на каждом шаге алгоритма от текущей температуры $T$ зависит как новый случайный элемент, так и вероятность его принятия как текущего. Итак, конкретная схема метода отжига задается следующими параметрами:

1. выбором закона изменения температуры $T(k)$, где $k$ – номер шага;
2. выбором вероятностного распределения $Q(x; T)$;
3. выбором функции вероятности принятия $h(\triangle E; T)$

В общем виде алгоритм можно представить следующим способом:

1. Случайным образом выбирается начальная точка $x = x_0; x_0 \in S$. Текущее значение энергии $E$ устанавливается в значение $f(x_0)$.
2. $k$-я итерация основного цикла состоит из следующих шагов:
   (a) Сравнить энергию системы $E$ в состоянии $x$ с найденным на текущий момент глобальным минимумом. Если $E = f(x)$ меньше, то изменить значение глобального минимума.
   (b) Сгенерировать новую точку $x^l = G(x; T(k))$.
   (c) Вычислить значение функции в ней $E^l = f(x^l)$.
   (d) Сгенерировать случайное число $\alpha$ из интервала $[0; 1]$.
   (e) Если $\alpha < h(E^l - E; T(k))$, то установить $x] \mapsto x^l; E \mapsto E^l$ и перейти к следующей итерации.

Иначе повторить шаг (b), пока не будет найдена подходящая точка $x^l$. Метод имитации отжига широко применяется при решении NP-полных задач, а также как алгоритм обучения нейронных сетей размерностью до нескольких сотен синаптических весов. В случаях, когда в качестве критерия останова выступает достижение решения, удовлетворяющего определенным ограничениям, возможно получение результата за малое количество итераций, но не соответствующего заданной точности.

### Машина Больцмана

Машина Больцмана была исследована и предложена в 1985 году Дж.Е. Хинтоном и Р. Земелом. Машина Больцмана похожа по функции и действию на сеть Хопфилда и включает понятие “моделированного отжига” для поиска в пространстве состояний [13]. Подобно сети Хопфилда, машина Больцмана имеет пространство состояний, которое базируется на весах соединений в слое образов. Состояния активности являются биполярными, то есть характеризуются значениями $\pm 1$. Энергия сети Хопфилда определяется следующей формулой:

$$E = -\frac{1}{2} \sum_j \sum_i s_j s_i w_{ji},$$

где $s$ обозначает состояние элемента сети. Если элемент $j$ изменяет состояние $\triangle s_j$, то изменение энергии будет равно:

$$\triangle E = - \triangle s_j \sum_i s_i w_{ij}. \tag{15}$$

Перепишем формулу изменения энергии 15 в следующем виде:

$$\triangle E = -2 \triangle s_j \sum_i s_i w_{ij}.$$

Докажем, что такое допущение применимо в данном случае. Если состояние $s_j$ в данный момент равно $-1$ и оно изменяется на $+1$, то такое изменение будет равно $+2$ или $-2s_j$. Если состояние $s_j$ в данный момент равно $+1$ и оно стремится изменится к $-1$, то изменение соответственно будет равно $-2$ или $-2s_j$. Для вычисления вероятности принятия какого-либо изменения состояния используем следующую функцию-сигмоид:

$$p = \frac{1}{1 + \exp(-\frac{\triangle E}{T})}.$$

Вероятность перехода элемента в новое состояние при этом:

$$p = \frac{1}{1 + \exp(2s_j \frac{\sum\limits_i s_i w_{ji}}{T})}. \tag{16}$$

Машина Больцмана используется для решения ряда проблем, связанных с оптимизацией, включая проблему коммивояжера. Машина Больцмана может иметь и скрытые элементы. Видимые элементы, используемые для взаимодействия с внешней средой, делятся на входные и выходные элементы (Рис. 3).

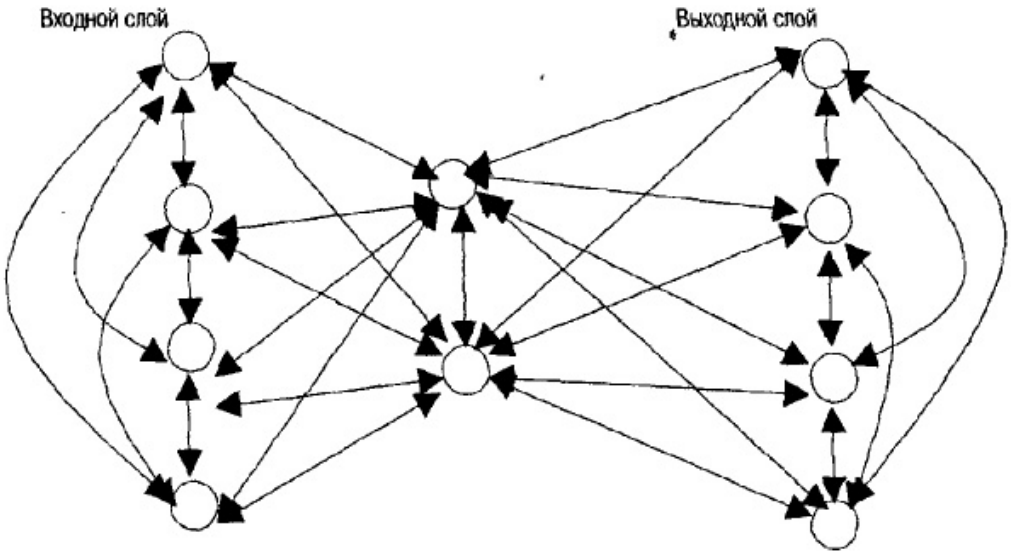


**Рис. 3.** Общая схема машины Больцмана с одинаковым количеством входных и выходных слоев

Все элементы имеют двунаправленные связи со всеми элементами, кроме элементов входного слоя, которые в свою очередь не связаны с элементами выходного слоя непосредственно. Сеть с такой архитектурой легко поддается обучению, в ходе которого входные и выходные элементы закрепляются(задаются взаимно соответствующими входным и выходным векторами). При завершении обучения закрепляются только входные элементы, и сеть выполняет некоторое число итераций до тех пор, пока в результате не установятся значения выходных элементов. Процесс обучения машины Больцмана разделяется на два этапа: этап фиксации, в котором входные и выходные элементы закрепляются входными и целевыми образцами, и этап свободного выполнения, при котором фиксируются только входные элементы. Сетевая статистика собирается в течение обоих этапов и используется для обновления весовых значений. Вес связи между $i$ и $j$ изменяется в соответствии с формулой:

$$\triangle w_{ij} = \eta(p_{ij}^{+} - p_{ij}^{-}),$$

где ($p_{ij}^{+}$ – корреляция между элементами $i$ и $j$ при этапе фиксации, ($p_{ij}^{-}$ – корреляция при этапе свободного выполнения; в данном случае корреляция подразумевает под собой вероятность одновременного нахождения двух элементов во “включенном состоянии”. Здесь для каждой пары входных и выходных образцов закрепляют видимые элементы и выполняется “модельная закалка”. Для каждого уровня температуры T происходит обновления состояний скрытых

элементы сети в соответствии с (16). Для конечной температуры $T_{end}$ собирается статистика, позволяющая оценить корреляции $p_{ij}^{+}$. Процедура повторяется для этапа свободного выполнения .по завершении которой обновляются весовые значения. Весь алгоритм останавливается тогда, когда не происходят изменения весовых значений (схождение сети). Механизм работы машины Больцмана дает возможность выбирается из локальных экстремумов адаптивного рельефа пространства состояний, но при этом имеет недостаток, связанный с медленным алгоритмом обучения.

## 4. ВЫВОДЫ

При создании и построении систем принятия решений возникает необходимость в решении самых разнообразных задач оптимизации. В статье были рассмотрены методы многомерной оптимизации как методы решения общей задачи поиска глобального и локального экстремума, а также проведено сравнение их эффективности. Результатом исследования является анализ применимости рассмотренных алгоритмов к тому или иному типу оптимизируемой функции и, соответственно, к той или иной задаче оптимизации.

## СПИСОК ЛИТЕРАТУРЫ


1. Захарова Е.М., Кузнецов Н.А., Минашина И.К., Пащенко Ф.Ф., Рябых Н.Г. Моделирование алгоритмов оптимизации мультиагентной системы управления перевозочным процессом. *Вестник Международной Академии Системных Исследований. Информатика, Экология, Экономика.* 2014. Т. 16. № -1. С. 9-15.
2. Барабашова О. В., Крушель Е. Г. *Алгоритмы поиска экстремума функции многих переменных.* Методические указания, Волгоград. гос. техн. ун-т, Волгоград, 2000. - 30с.
3. Граничин О.Н., Поляк Б.Т. *Рандомизированные алгоритмы оценивания и оптимизации при почти произвольных помехах.* М.: Наука, 2003. - 291 с.
4. Пащенко Ф.Ф. *Введение в состоятельные методы моделирования систем; Учеб. пособие: В 2-х ч. Ч. 1. Математические основы моделирования систем. Ч. 2. Идентификация нелинейных систем.* М.: Финансы и статистика, 2007.
5. Вазан М. *Стохастическая аппроксимация.* М.: Наука, 2003. - 291 с.
6. Dvoretzku A. On Stochastic Approximation, *Third Berkeley Symp. on Math. Stat. and Probability* University of California - 1956.- vol.1. pp.39-56.
7. Горбань А. Н., Дунин, Барковский В. Л., Кардин А. Н., Новиков Е. А. *9. Нейроинформатика.* Новосибирск: Наука, 1998.- 295 с.
8. Bruck J. On the convergence properties of the Hopfield model, *Proceedings of the IEEE.* - 1990. - V. 78. - P. 1579-1585.
9. Колков Д.А. Анализ интервальных методов поиска глобального экстремума, *Фундаментальные исследования*, 2006, том № 2,стр. 22-23
10. Hansen E., *Global Optimization Using Interval Analysis*, New York: Dekker, 1992.
11. Барский А.Б. *Параллельные процессы в вычислительных системах: планирование и организация*, М.: Радио и связь, 1990.
12. Simon Haykin. *Neural Networks: A Comprehensive Foundation (2nd ed.)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998
13. Rongxiang Liu, Tom W. Blackwell, and David U. States. A structure based similarity measure for nucleic acid sequence comparison. In Proceedings of the second annual international conference on Computational molecular biology. *RECOMB '98*, Sorin Istrail, Pavel Pevzner, and Michael Waterman (Eds.). ACM, New York, NY, USA, 173-181.1998

*Статью представил к публикации член редколлегии Н.А.Кузнецов*

# Review of multidimensional optimization techniques

**I.K.Minashina, E.M.Zakharova**

The article considers basic multidimensional optimization techniques and provides the comparison of their efficiency. An analysis of the applicability of considered algorithms to different types of objective functions is also covered.

**KEYWORDS:** multidimensional optimization, conventional optimization, local extremum, global extremum, multiextremal function.