**А.В. Еремеев**

# ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И ОПТИМИЗАЦИЯ

2-е издание, исправленное и дополненное

Учебное пособие

# Оглавление

# Введение

Эволюционные алгоритмы (ЭА), к числу которых относятся генетические алгоритмы [32], эволюционные стратегии [37], алгоритмы генетического программирования [33] и др., берут начало в первых работах по эволюционному моделированию Л. Фогеля с соавт. [19] и Дж. Голланда [32], где было предложено моделировать процесс биологической эволюции для синтеза эффективных структур и создания систем искусственного интеллекта. Почти одновременно с возникновением метода эволюционного моделирования, А.Г. Ивахненко и Л.А. Растригиным были предложены методы группового учета аргументов [7] и случайного поиска [15], где также использовались идеи эволюции и случайной мутации.

Основной принцип работы ЭА основан на компьютерном моделировании процесса эволюции с учетом факторов изменчивости, наследования и отбора наиболее приспособленных особей. В ЭА эволюционное моделирование, как правило, используется не для исследования биологических популяций, а для решения задач оптимизации и принятия решений. Базовая идея использования ЭА в оптимизации состоит в построении некоторого множества (популяции) допустимых решений оптимизационной задачи и применении «малых» случайных преобразований с целью получения решений более высокого качества (более приспособленных). При этом чем больше приспособленность особи, тем больше она имеет шансов быть выбранной в качестве родительской для построения новых решений. ЭА имеют многочисленные приложения при решении разнообразных задач в управлении, планировании, проектировании, распознавании образов и в других областях (см., например, [3, 12, 13, 16]).

Одним из типичных представителей ЭА является классический генетический алгоритм (КГА), предложенный Дж. Голландом [32]. В этом алгоритме приспособленность особей к усло-

виям «окружающей среды» выражается некоторой монотонной функцией от значения целевой функции задачи. Чем выше качество допустимого решения, понимаемое как значение целевой функции, тем выше его приспособленность. Популяция развивается за счет отбора родительских особей с помощью оператора пропорциональной селекции и применения к ним случайных операторов, имитирующих мутацию генов и рекомбинацию родительских генотипов (кроссинговер). В различных вариантах генетических алгоритмов (ГА) используются разные операторы отбора, при этом особи с бóльшим значением приспособленности, в среднем, получают большее число потомков на следующем поколении.

При создании ГА находит применение бионический подход, состоящий в заимствовании принципов организации систем из живой природы. В данном случае имеет место использование принципа постепенных адаптивных преобразований в пределах популяции или вида в ходе, так называемой, микроэволюции.

Как показывают исследования акад. Ю.П. Алтухова и других авторов (см., [1], § 6.4), более «масштабная» межвидовая изменчивость (макроэволюция) требует скачкообразных перестроек генотипа и не может быть выведена непосредственно из постепенной внутривидовой изменчивости. В связи с недостаточной изученностью таких скачкообразных перестроек, привлечение известной теории происхождения видов для обоснования работоспособности ГА представляется весьма проблематичным (см. [39], §1.3).

Ограниченный объем настоящего пособия не позволяет дать подробного изложения теоретических результатов, полученных для ГА (см., например, [23, 39]), и описать весь класс эволюционных алгоритмов, включающий в себя также эволюционные стратегии [37], алгоритмы генетического программирования [33], имитации отжига [12] и др. Интересующийся читатель может найти информацию по этим темам в указанных источниках.

Материал настоящего пособия организован следующим образом. Описанию классического варианта ГА и его применению посвящен § 1. Различные модификации ГА обсуждаются в § 2. Далее, в § 3 рассматриваются методы гибридизации ГА со специализированными точными и приближенными алгоритмами.

# 1. Классический генетический алгоритм

## 1.1. Описание алгоритма

Классический генетический алгоритм был предложен Дж. Голландом [32], как алгоритм, имитирующий адаптацию популяции к заданной *функции приспособленности*. При этом, как и в популяционной генетике [1], под приспособленностью генотипа понимается среднее число потомков, порождаемых за время жизни особи данного генотипа в определенных условиях окружающей среды. Впоследствии КГА стал активно использоваться и как метод оптимизации [16, 29]. Пусть задача оптимизации «на максимум» имеет общий вид

$$\max\{f(x) : x \in X\}, \tag{1}$$

где множество $X$ – *пространство решений*, например, евклидово пространство $\mathbb{R}^q$ или пространство $\{0,1\}^n$ булевых строк длины $n$. Процесс работы КГА представляет собой последовательную смену популяций (поколений), состоящих из фиксированного числа *особей*. Здесь и далее под особью понимается элемент пространства решений $x \in X$ или некоторое его представление в алгоритме. Чем больше значение целевой функции особи, тем больше шансов она имеет оставить потомков в следующем поколении и тем больше приспособленность соответствующего генотипа (этот принцип легко адаптируется и применительно к задачам минимизации). При формировании следующего поколения часть потомков полностью идентична родительским особям, а часть изменяется некоторым случайным образом под действием операторов мутации и кроссинговера (скрещивания).

При использовании КГА решения представляются двоичными строками фиксированной длины $l$. Каждой строке из $l$ символов алфавита $\{0,1\}$ сопоставляется элемент пространства $X$, т.е. определяется функция $x : B \to X$, где $B = \{0,1\}^l$. Данная функция называется *схемой представления решений*. Строки $\xi \in B$ принято называть *генотипами* или *хромосомами*, а их образы

$x(\xi) \in X$ – фенотипами. Здесь и далее для обозначения элемента пространства решений используется тот же символ $x$, что и для схемы представления $x(\cdot)$, однако, из контекста смысл символа $x$ всегда будет ясен.

Употребление терминов «фенотип» и «генотип» связано с тем, что при анализе и применении генетических алгоритмов важное значение имеет способ внутреннего представления решений в алгоритме. Элементы строки $\xi \in B$ принято называть *генами* по аналогии с участками молекулы ДНК. Если пространство решений $X$ имеет бесконечную мощность, то в КГА его неявно ограничивают множеством лишь тех решений (фенотипов), которые представимы генотипами.

Популяцией $\Pi = (\xi^1, \xi^2, \ldots, \xi^N)$ численности $N$ называется вектор пространства $B^N$, элементами которого являются генотипы особей данной популяции. Способ нумерации особей в популяции КГА не имеет значения. Популяция поколения $t$, $t \geq 0$ будет обозначаться через $\Pi^t = (\xi^{1t}, \xi^{2t}, \ldots, \xi^{Nt})$. Итерацией КГА является переход от текущей популяции $\Pi^t$ к следующей популяции $\Pi^{t+1}$. Численность популяции $N$ фиксирована от начала работы алгоритма до конца и для удобства предполагается, что $N$ четно.

При оценке «качества» особи в КГА вместо целевой функции $f(\cdot)$ исходной задачи используется *функция приспособленности* генотипа[1] $\Phi(\xi) = \phi(f(x(\xi)))$, где $\xi \in B$. Здесь $\phi : \mathbf{R} \to \mathbf{R}^+$ – некоторая монотонно возрастающая функция,[2] выбираемая при адаптации КГА к конкретной задаче оптимизации с учетом ее специфики. Простейшим примером функции приспособленности является сама целевая функция $f(x(\xi))$, если она неотрицательна.[3] В биологической интерпретации максимумы функ-

[1] В англоязычной литературе используется термин *fitness function*.

[2] В отдельных случаях эта функция определяется с учетом состава текущей популяции $\Pi^t$. Тогда мы будем ее обозначать через $\phi^{(\Pi^t)}$, а функцию приспособленности – через $\Phi^{(\Pi^t)}(\xi)$.

[3] О задании функции приспособленности в случае, когда $f(x)$ может принимать отрицательные значения см. в конце п. 1.2.

ции $\Phi(\xi)$ соответствуют наиболее приспособленным генотипам для данной «окружающей среды».

Условимся лучший из найденных генотипов к поколению $t$ обозначать через $\tilde{\xi}^t$, т.е.

$$\tilde{\xi}^t = \text{argmax}\{\Phi(\xi^{i,\tau}), i = 1, ..., N, \tau = 0, ..., t\}.$$

Приведем общую схему КГА. Используемые здесь вероятностные операторы $\text{Sel} : B^n \to \{1, \ldots, N\}$, $\text{Cross} : B \times B \to B \times B$ и $\text{Mut} : B \to B$ будут описаны ниже.

**Классический генетический алгоритм**

Положить $t := 0$.
**1.** Для $k$ от 1 до $N$ выполнять:
**1.1.** Построить случайным образом генотип $\xi^{k,0}$.
**2.** Для $k$ от 1 до $N/2$ выполнять шаги 2.1-2.3:
**2.1.** Селекция: выбрать генотипы $\xi := \xi^{Sel(\Pi^t),t}$, $\eta := \xi^{Sel(\Pi^t),t}$.
**2.2.** Скрещивание: построить $(\xi', \eta') := \text{Cross}(\xi, \eta)$.
**2.3.** Мутация: положить $\xi^{2k-1,t+1} := \text{Mut}(\xi')$, $\xi^{2k,t+1} := \text{Mut}(\eta')$.
**3.** Положить $t := t + 1$.
**4.** Если $t \leq t_{max}$, то идти на шаг 2, иначе – на шаг 5.
**5.** Результатом работы КГА является лучшее из найденных решений $x(\tilde{\xi}^{t_{max}})$.

Поясним приведенную схему. На шаге 0 формируется начальная популяция $\Pi^0$, элементы которой генерируются в соответствии с равномерным распределением на множестве генотипов $B$, т.е. $P\{\xi_k^{i,0} = 0\} = P\{\xi_k^{i,0} = 1\} = 1/2$, $i = 1, \ldots, N$, $k = 1, \ldots, l$.

Вероятностный оператор селекции особей на пространстве популяций $\text{Sel}(\Pi)$ имеет то же значение, что и естественный отбор в природе. Действие этого оператора состоит в выборе номера родительской особи для построения очередного потомка. Генотип $\xi^{i,t}$ с номером $i$, $i = 1, \ldots, N$ из популяции $\Pi^t$ оказывается родительской особью при формировании очередного

генотипа $\xi^{k,t+1}$ популяции $\Pi^{t+1}$ с вероятностью

$$P_{\mathtt{s}}(i,\Pi^t)=\frac{\Phi(\xi^{i,t})}{\sum_{j=1}^{N}\Phi(\xi^{j,t})}. \tag{2}$$

Иными словами, любая особь $\xi^{i,t}$ популяции $\Pi^t$ принимает участие в создании следующего поколения в соответствии со схемой Бернулли из $N$ испытаний, где вероятность успеха равна $P_{\mathtt{s}}(i,\Pi^t)$. При этом не исключается выбор $\xi^{i,t}$ одновременно в качестве $\xi$ и $\eta$ на шаге 2.1. Описанный оператор Sel иногда также называют селекцией методом рулетки [16, 29]. Предположим, что колесо рулетки разбито на $N$ секторов, причем сектор $i$ соответствует особи $i$ и имеет радианную меру $2\pi P_{\mathtt{s}}(i,\Pi^t)$. Тогда селекцию особи $\xi^{i,t}$ можно представлять, как выбор $i$-го сектора на колесе рулетки.

**Определение 1.1.** *Селекция называется пропорциональной, если число число случаев, когда особь выступает в качестве родителя, в серднем пропорционально отношению ее приспособленности к сумме приспособленностей всех особей популяции.*

Легко видеть, что описанный выше оператор селекции осуществляет пропорциональную селекцию, т.к. среднее число потомков каждой особи $\xi^{i,t}$ равно $N\Phi(\xi^{i,t})/\sum_{j=1}^{N}\Phi(\xi^{j,t})$. Если задать функцию приспособленности $\Phi^{(\Pi^t)}(\xi^{j,t})$ так, чтобы выполнялось условие

$$\sum_{j=1}^{N}\Phi^{(\Pi^t)}(\xi^{j,t})=N, \tag{3}$$

то приспособленность особи будет равна среднему числу ее потомков. Таким образом, заимствование термина «приспособленность» из биологии вполне оправдано, по крайней мере, в том частном случае, когда выполняется равенство (3). В общем случае значения функции $\Phi(\cdot)$ легко нормировать с помощью такого масштабирования функции $\phi(\cdot)$, которое обеспечивало бы

соотношение (3) на каждой итерации КГА, однако, на практике это не имеет смысла.

Следующее упражнение связано с вычислительной сложностью процедуры селекции.

**Упражнение 1.1.** *Показать, что при известных значениях приспособленности особей текущей популяции* $\Pi^t$ *селекция всех родительских особей для построения новой популяции* $\Pi^{t+1}$ *может быть выполнена в КГА за время* $O(N \log_2(N))$.

Далее, для описания КГА необходимо описать двуместный оператор кроссинговера (скрещивания) $\mathrm{Cross}(\xi, \eta)$ и одноместный оператор мутации $\mathrm{Mut}(\xi)$, действие которых носит случайный характер.

Результат кроссинговера $(\xi', \eta') = \mathrm{Cross}(\xi, \eta)$ с вероятностью $P_\mathtt{c}$ формируется в виде

$$\xi' = (\xi_1, \xi_2, ..., \xi_\chi, \eta_{\chi+1}, ..., \eta_l),$$

$$\eta' = (\eta_1, \eta_2, ..., \eta_\chi, \xi_{\chi+1}, ..., \xi_l),$$

где случайная координата скрещивания $\chi$ выбрана с равномерным распределением от 1 до $l-1$. С вероятностью $1 - P_\mathtt{c}$ оба генотипа сохраняются без изменений, т.е. $\xi' = \xi,\ \eta' = \eta$. Влияние оператора кроссинговера регулируется параметром $P_\mathtt{c}$. Данный оператор принято называть *одноточечным кроссинговером.*[4]

Оператор мутации в каждой позиции генотипа с заданной вероятностью $P_\mathtt{m}$ изменяет ее содержимое. В противном случае ген остается без изменений. Таким образом, мутация элементов генотипа происходит по схеме Бернулли с вероятностью успеха $P_\mathtt{m}$.

Изменение вероятностей мутации и кроссинговера позволяет регулировать работу КГА и настраивать его на конкретные задачи. Увеличение вероятности мутации до 0.5 превращает КГА

[4] В англоязычной литературе используется термин *one-point crossover.*

в простой случайный перебор, имеющий весьма ограниченное применение (см. [15], § 6.1). Уменьшение же $P_{\mathtt{m}}$ до нуля приводит к малому разнообразию генотипов в популяции и может вызвать «зацикливание» КГА, когда на каждой итерации генерируются лишь ранее встречавшиеся генотипы. Величины $P_{\mathtt{c}}$ и $N$ также могут существенно влиять на скорость сходимости популяции к решениям приемлемого качества (см., например, [6, 16]). Настраиваемые параметры КГА выбирают, как правило, в следующих диапазонах: $0 \leq P_{\mathtt{c}} \leq 1,\ 10^{-3} \leq P_{\mathtt{m}} \leq 0.3,\ 30 \leq N \leq 1000.$

## 1.2. Применение к задачам оптимизации

Рассмотим простейший пример применения КГА к задаче вида (1). Пусть $\mathbb{R}^+$ обозначает множество неотрицательных вещественных чисел.

**Пример 1.** Требуется максимизировать функцию $f : X \to \mathbb{R}^+$, при $X = \{0, 1, ..., 2^n - 1\}$, где $n$ – целое. При адаптации КГА для поиска решений такой задачи естественно положить $l = n$ и воспользоваться двоичной системой счисления для записи решений:

$$x(\xi) = \sum_{j=0}^{l-1} \xi_{l-j} 2^j.$$

В качестве функции приспособленности может быть использована целевая функция: $\Phi(\xi) = f(x(\xi))$.

**Пример 2.** Рассмотрим задачу о максимальном разрезе в графе. Дан граф $G = (V, E), V = \{v_1, ..., v_|V|\}$, каждому ребру приписан вес $w : E \to \mathbb{R}^+$. *Разрезом* в графе $G$ называют пару подмножеств $\{U, U'\}$, таких что $U \cup U' = V,\ U \cap U' = \emptyset$. Для краткости записи в дальнейшем разрез будем обозначать как одно из его подмножеств (в данном случае $U$ или $U'$), т.к. второе

подмножество определяется однозначно. Требуется найти разрез $U$ с максимальным весом

$$W(U) = \sum_{\substack{e=(u,v)\in E:\\ u\in U,\ v\in V\setminus U}} w(e).$$

Если верна известная гипотеза P≠NP (см., например, [5]), то для данной задачи не может быть построено эффективных алгоритмов, с гарантией отыскивающих оптимальное решение.[5] В связи с этим имеет смысл разработка эвристических алгоритмов ее решения, и в частности, генетических алгоритмов. Для применения КГА положим $X = \{U : U \subseteq V\}$, $f(U) = W(U)$. Представление решений можно определить так:

$$x(\xi) = \{v_j \in V : \xi_j = 1, j = 1, ..., |V|\},$$

при этом полагая $l = |V|$ и $\Phi(\xi) = f(x(\xi))$.

Очевидно, при данной кодировке операторы Mut и Cross сохраняют допустимость решений. Необходимо отметить, что при этом возникает, так называемая, проблема вырожденности представления: один и тот же разрез $\{U, U'\}$ может быть представлен двумя способами (либо символ «1» кодирует вершину, лежащую в $U$, либо в $U'$). Такая неоднозначность может привести к снижению эффективности работы алгоритма из-за скрещивания особей, закодированных разными способами.

**Упражнение 1.2.** *Предложить взаимно-однозначное представление решений задачи о максимальном разрезе в графе при* $l = |V| - 1$.

**Задачи условной оптимизации.** Наряду с задачами безусловной оптимизации (1) на практике часто возникают задачи

[5] Здесь и далее термин *эффективный алгоритм* означает алгоритм с трудоемкостью, ограниченной некоторым полиномом от длины записи входных данных задачи. Задача, для которой существует эффективный алгоритм, называется *эффективно разрешимой*.

*условной оптимизации* вида

$$\max\{F(x) : x \in D \subseteq X\}, \tag{4}$$

где $D$ – область допустимых решений. Для поиска решений таких задач также может быть использован КГА. Один из подходов основан на «штрафовании» решений, находящихся за пределами допустимой области. Простейший способ состоит в том, чтобы полагать $f(x) = F(x)$ при $x \in D$, иначе $f(x) = \mu$, где $\mu$ выбирается так, чтобы для любого $x' \in X \backslash D$ выполнялось $f(x') < \max_{x \in D} f(x)$.

**Пример 3.** Пусть требуется максимизировать функцию $f : \mathbf{Z} \to \mathbb{R}^+$ на множестве $D = \{a, a+1, ..., b\}$, где $0 < a < b$ и $\mathbf{Z}$ – множество целых чисел. Как и в примере 1 воспользуемся двоичной системой счисления:

$$x(\xi) = a + \sum_{j=0}^{l-1} \xi_{l-j} 2^j, \tag{5}$$

где $l = \lceil \log_2(b-a) \rceil$, и символ $\lceil y \rceil$ обозначает минимальное целое число не меньшее $y$. При $x(\xi) \in \{a, a+1, ..., b\}$ полагаем $\Phi(\xi) = F(x(\xi))$, иначе $\Phi(\xi) = 0$.

Так как не менее половины генотипов при данной кодировке отображаются в допустимые решения из области $D$, то с большой вероятностью уже в начальной популяции будут найдены допустимые решения.

В общем случае недостатком данного способа учета недопустимости решений является то, что все точки вне области $D$ оказываются «одинаково плохи» и функция приспособленности не дает никакой информации об удаленности недопустимой точки от области $D$. Во многих задачах оптимизации поиск допустимого решения сам по себе представляет достаточно сложную задачу и без информации об удаленности точки от области $D$ алгоритм может не дать допустимых решений.

В примере 4 будет рассмотрен другой способ использования штрафной функции, учитывающий лучше специфику задачи.

Если $D \subset X = \mathbb{R}^q$ и множество $D$ ограничено, то его можно дискретизовать, например, с помощью перехода к дискретной решетке точек с достаточно малым шагом. Таким образом, точное решение задач часто заменяется поиском оптимума на дискретном подмножестве некоторого $q$-мерного параллелепипеда $\Omega \subset X$.

**Пример 4.** Пусть область $D$ погружена в $q$-мерный параллелепипед $\Omega$:

$$D \subseteq \Omega = \{x \in \mathbb{R}^q : a_1 \leq x_1 \leq b_1, ..., a_q \leq x_q \leq b_q\}.$$

Запишем последовательно все координаты вектора фенотипа в двоичной системе счисления:

$$x(\xi)_i = a_i + \frac{d_i}{2^k - 1} \sum_{j=0}^{k-1} \xi_{ki-j} 2^j, \quad i = 1, ..., q, \tag{6}$$

где $d_i = b_i - a_i,\ i = 1, ..., q$. Здесь предполагается, что на кодирование каждой координаты используется $k$ бит, $l = kq$ и последовательность генов $g^i = (\xi_{k(i-1)+1}, ..., \xi_{ki})$ кодирует $i$-ю координату.

Если область $D$ задана системой неравенств

$$D = \{x \in \mathbb{R}^q : f_1(x) \leq 0, ..., f_m(x) \leq 0\},$$

то может быть использована штрафная функция

$$P(x) = \sum_{i=1}^{m} \max\{f_i(x), 0\},$$

при этом полагают $f(x) = F(x) - rP(x)$. Значение $r$ выбирается достаточно большим для того, чтобы какое-либо недопустимое

решение не превзошло оптимум по значению функции $f$. Например, естественно потребовать чтобы выполнялось условие

$$f(x(\xi)) < f(x(\eta)) \text{ для } \xi, \eta, \text{ таких что } x(\xi) \notin D,\ x(\eta) \in D. \quad (7)$$

Подробнее о сводимости задач математического программирования к задачам безусловной оптимизации см., например, в [8].

Наряду с использованием штрафных функций возможны и другие методы учета ограничений. Например, для многих задач дискретной оптимизации при получении недопустимого решения на выходе оператора мутации, такое решение может быть скорректировано с помощью «жадных» алгоритмов (см. п. 3.2). Другой подход состоит в выборе подходящей кодировки, при которой $x(B) \subseteq D$. С этой целью могут быть использованы, в частности, методы, описанные в п. 3.3.

**Пример 5.** Общая схема КГА легко адаптируется для задач целочисленного линейного программирования (ЦЛП). Рассмотрим задачу ЦЛП следующего вида: найти

$$F(x) = (c, x) \to \max \quad (8)$$

при условиях

$$Ax \leq b,\ x \geq 0, \quad (9)$$

$$x \in \mathbf{Z}^q. \quad (10)$$

Здесь $A$ – матрица размерности $(m \times q)$, $c = (c_1, ..., c_q)^T$, $b = (b_1, ..., b_m)^T$, $x = (x_1, ..., x_q)^T$. Далее предполагаем, что множество $\mathcal{M}$, определяемое системой неравенств (9), ограничено. Будем называть вектор целочисленным, если все его компоненты целочисленны.

Многогранник допустимых решений погружается в $q$-мерный параллелепипед

$$\Omega = \{x \in \mathbb{R}^q | d'_j \leq x_j \leq d''_j, j = 1, ..., q\}.$$

Границы параллелепипеда $d'_j, d''_j$ могут быть найдены, например, решением $2q$ задач линейного программирования с системой ограничений (9) и целевыми функциями $x_j \to \max$, $x_j \to \min$, $j = 1, \ldots, q$.

Минимальная длина бинарной строки для кодировки $j$-ой координаты целочисленной точки из $\Omega$ составляет $k_j = \lceil \log_2(d''_j - d'_j) \rceil$ бит при $d'_j < d''_j$. Не теряя общности, далее будем предполагать что $d'_j < d''_j$ для всех $j = 1, \ldots, q$. Допустимым целочисленным точкам в $\Omega$ сопоставляются элементы пространства $B$, состоящие из $q$ последовательно записанных двоичных представлений координат. Таким образом, $B = \{0,1\}^{k_1+\ldots+k_q}$ и

$$x(\xi)_j = d'_j + \sum_{i=0}^{k_j-1} 2^i \xi_{k_1+\ldots+k_j-i}, \quad j = 1, ..., q. \tag{11}$$

Определим функцию $f(x)$ с учетом «штрафа»:

$$f(x) = \begin{cases} F(x) & \text{при} \quad s(x) = 0 \\ F(X) - rs(x) & \text{при} \quad s(x) > 0, \end{cases}$$

где $s(x)$ – сумма нарушений системы ограничений (9) для точки $x$; $r$ – некоторая положительная величина, достаточная чтобы выполнялось условие (7).

Функция приспособленности определяется следующим образом:

$$\Phi(\xi) = \phi^{(\Pi^t)}(f(x(\xi))) = \frac{f(x(\xi)) - f^t_{\min}}{f^t_{\text{avg}} - f^t_{\min}}, \tag{12}$$

где $f^t_{\text{avg}}, f^t_{\min}$ – среднее и минимальное значения функции $f(x(\xi))$ на текущей популяции $\Pi^t$. Легко видеть, что определенная таким образом функция $\Phi(\xi)$ неотрицательна и неубывает с ростом $f(x)$. Данный вариант функции приспособленности позволяет решать с помощью КГА задачи без ограничения на знак целевой функции.

## 1.3. Кодировка Грея

Заметим, что представление в некоторых близких чисел в двоичной системе счисления может существенно различаться (например, представления чисел 127 и 128 различаются в восьми позициях). Данного недостатка лишена кодировка Грея [18].

Рассмотрим для простоты случай $q = 1$. При этом отображение $x(\xi)$ в формуле (6) заменяется композицией $x(\gamma(\xi))$, где преобразование $\gamma$ переводит число из кодировки Грея $\xi$ в его запись $\tilde{\xi}$ в двоичной системе счисления по формуле:

$$\tilde{\xi}_k = \bigoplus_{j=1}^{k} \xi_j, \ k = 1, ..., l,$$

где символ $\bigoplus$ обозначает сложение по модулю два. Данное отображение взаимно-однозначно и обратный оператор $G = \gamma^{-1}$ действует следующим образом: $G(\tilde{\xi}) = (\tilde{\xi}_1, \tilde{\xi}_1 \oplus \tilde{\xi}_2, ..., \tilde{\xi}_{l-1} \oplus \tilde{\xi}_l)$ (нетрудно проверить что $\gamma(G(\tilde{\xi})) = \tilde{\xi}$ для любого $\tilde{\xi}$.)

Пусть $\beta_l(i)$ – запись целого числа $i$ в двоичной системе счисления с использованием $l$ битов. Обозначим через $\rho(\xi, \xi') = \sum_{j=1}^{l} |\xi_j - \xi'_j|$ расстояние в метрике Хэмминга между строками $\xi$ и $\xi'$. Из таблицы 1 видно, что расстояние Хэмминга между кодировками чисел, различающихся на 1, равно 1.

**Упражнение 1.3.** *Доказать, что для всякого* $i = 1, 2, ..., 2^l - 1$ *выполняется равенство*

$$\rho(G(\beta_l(i-1)), G(\beta_l(i))) = 1.$$

Особенно полезна данная кодировка в задачах управления, где параметры непрерывно изменяются с течением времени и популяция должна постоянно следовать за экстремумом целевой функции.

Таблица 1. Двоичная система счисления и кодировка Грея

| $i$ | $\beta_3(i)$ | $G(\beta_3(i))$ |
|---|---|---|
| 0 | 000 | 000 |
| 1 | 001 | 001 |
| 2 | 010 | 011 |
| 3 | 011 | 010 |
| 4 | 100 | 110 |
| 5 | 101 | 111 |
| 6 | 110 | 101 |
| 7 | 111 | 100 |

## 2. Модификации классического генетического алгоритма

Для многих задач оптимизации применение КГА оказывается малоэффективным, т.к. в данном алгоритме слабо учитывается специфика задачи. В связи с этим разработано большое число модификаций КГА. Далее все такие модификации будем называть *генетическими алгоритмами* (ГА).

Рассмотрим, к примеру, ряд модификаций КГА для задачи ЦЛП из п. 1.2. Классическая схема одноточечного кроссинговера для этой задачи может быть адаптирована так, чтобы $\chi$ выбиралось с равномерным распределением из множества $\{k_1, k_1 + k_2, ..., k_1 + ... + k_{q-1}\}$. Данная модификация позволяет «разрывать» генотипы только в точках соединения двоичных представлений координат. Как отмечено Р.Т. Файзуллиным [28], действие такого оператора кроссинговера соответствует некоторому повороту родительских фенотипов вокруг середины соединяющего их отрезка. Ввиду отсутствия доказательства этого утверждения в других источниках, приведем его с доказательством.

**Утверждение 2.1.** *Пусть* $\chi \in \{k_1, k_1 + k_2, ..., k_1 + ... + k_{q-1}\}$. *Тогда существует такое преобразование поворота* $R_{\chi,\xi,\eta}$ *в* $\mathbb{R}^q$, *при котором* $x(\xi') = R_{\chi,\xi,\eta}(x(\xi)), \quad x(\eta') = R_{\chi,\xi,\eta}(x(\eta))$. *При этом середина отрезка, соединяющего родительские фенотипы,* $x^0 = \frac{x(\xi)+x(\eta)}{2}$, *остается неподвижной, т.е.* $R_{\chi,\xi,\eta}(x^0) = x^0$.

**Доказательство.** Обозначим через $r$ номер координаты, кодировка которой оканчивается на бите $\chi$, т.е. $\chi = k_1 + ... + k_r$. Будем искать оператор $R_{\chi,\xi,\eta}(x)$ в виде следующего аффинного преобразования:

$$R_{\chi,\xi,\eta}(x) = A(x - x^0) + x^0. \tag{13}$$

Случай 1. При $q-r$ четном предположим, что $A$ - диагональная матрица с $r$ единицами в начале диагонали, далее заполненная четным числом символов $-1$. С помощью непосредственной проверки легко убедиться, что отображением $A(x - x^0) + x^0$ с матрицей указанного вида дает тот же результат, что и при действии оператора кроссинговера. Действительно, для всех координат $j \leq r$ имеем

$$(A(x(\xi) - x^0) + x^0)_j = \frac{x(\xi)_j - x(\eta)_j}{2} + \frac{x(\xi)_j + x(\eta)_j}{2} = x(\xi)_j,$$

$$(A(x(\eta) - x^0) + x^0)_j = \frac{-x(\xi)_j + x(\eta)_j}{2} + \frac{x(\xi)_j + x(\eta)_j}{2} = x(\eta)_j,$$

а для всех $j$, таких что $r < j \leq q$, имеем

$$(A(x(\xi) - x^0) + x^0)_j = -\frac{x(\xi)_j - x(\eta)_j}{2} + \frac{x(\xi)_j + x(\eta)_j}{2} = x(\eta)_j,$$

$$(A(x(\eta) - x^0) + x^0)_j = -\frac{-x(\xi)_j + x(\eta)_j}{2} + \frac{x(\xi)_j + x(\eta)_j}{2} = x(\xi)_j.$$

То есть, представление (13) корректно.

Рассмотрим 3-мерное подпространство, образованное координатами $x_1, x_j, x_{j+1}$, при любом $j = r + 1, r + 3, ..., q - 1$.

В этом подпространстве действие кроссинговера описывается диагональной подматрицей матрицы $A$ с диагональю (1,-1,-1), задающей поворот вокруг оси $x_1$ на угол $\pi$. Преобразование матрицы $A$ есть композиция таких поворотов, следовательно, $A(x-x^0)+x^0$ является поворотом в $R^q$.

Случай 2. При $q-r$ нечетном рассмотрим матрицу $A$ вида

$$A = \begin{vmatrix} E' & 0 & 0 & 0\ldots 0 \\ 0\ldots 0 & \cos(-2\alpha) & -\sin(-2\alpha) & 0\ldots 0 \\ 0\ldots 0 & \sin(-2\alpha) & \cos(-2\alpha) & 0\ldots 0 \\ 0\ldots 0 & 0 & 0 & -E'' \end{vmatrix},$$

где $\alpha = \mathrm{arctg}\frac{x(\eta)_{r+1}-x(\xi)_{r+1}}{x(\eta)_r-x(\xi)_r}$, а единичные матрицы $E'$ и $E''$ имеют размерости $r-1$ и $q-r-1$, соответственно. Непосредственная проверка, подобная сделанной в случае 1, показывает что отображение $A(x-x^0)+x^0$ с матрицей указанного вида дает тот же результат, что и оператор кроссинговера.

Рассмотрим 2-мерное подпространство, образованное координатами $x_r, x_{r+1}$: здесь матрица $A$ задает поворот на угол $2\alpha$ вокруг начала координат. Во всех 3-мерных подпространствах, образованных координатами $x_1, x_j, x_{j+1}$, при $j = r+2, r+4, ..., q-1$ преобразование $A$ является поворотом вокруг оси $x_1$ на угол $\pi$, как и в случае 1. Следовательно, $A(x-x^0)+x^0$ является поворотом в $R^q$. Что требовалось.

Зачастую оптимальное решение задачи ЦЛП (8)-(10) оказывается расположенным сравнительно близко к оптимуму $\tilde{x}$ задачи линейного программирования (8)-(9), называемой *ЛП-релаксацией* исходной задачи. На практике решение $\tilde{x}$ может быть достаточно быстро найдено методами линейного программирования и дать «подсказку» о той области, откуда имеет смысл начинать поиск решения задачи ЦЛП при использовании ГА. Особи начальной популяции могут порождаться, например, с помощью $q$-мерного нормального распределения с математическим ожиданием в точке $\tilde{x}$. В таком случае в качестве генотипа используется двоичная запись округленных значений нормаль-

но распределенных координат.

Экспериментальные исследования показали [25], что если вероятность мутации достаточно велика, то дисперсия нормального распределения может быть установлена равной нулю.

## 2.1. Операторы селекции

Свойства оператора селекции оказывают существенное влияние на среднее время получения решений заданного качества и на разброс результатов при независимых реализациях эволюционного процесса в ГА. В ряде случаев оператор пропорциональной селекции из КГА может быть с успехом заменен другими операторами селекции. При этом выбор каждой родительской особи не обязательно осуществляется независимо от выбора других особей на данной итерации $t$.

Для сравнения различных операторов селекции необходимо выбрать некоторые характеристики, имеющие одинаковый смысл для всех из них. Возьмем в качестве таких характеристик математическое ожидание и дисперсию числа событий, состоящих в выборе особи $i$ в качестве родительской в процессе построения очередного поколения. Число таких случайных событий будет обозначаться через $Z(i, \Pi^t)$ и формально определяется следующим образом. Пусть $K_1, \ldots, K_N$ – номера родительских особей, выбранных из $\Pi^t$ при построении очередной популяции в ГА. Тогда $Z(i, \Pi^t)$ – число таких $j$, что $K_j = i$.

**Стохастическая универсальная селекция Бэкера.** При построении очередной популяции оператор селекции КГА выбирает особь $i$ в среднем $P_{\mathsf{s}}(i, \Pi^t)N$ раз, однако, случайная величина $Z(i, \Pi^t)$ может существенно отклоняться от этого среднего значения. Большие отклонения от среднего могут приводить к неустойчивой работе ГА в целом. Для снижения таких колебаний Дж. Бэкер предложил следующую модификацию оператора селекции, называемую *стохастической универ-*

*сальной селекцией*[6] [16, 29]: каждая особь $i$ гарантированно выбирается $\lfloor P_{\mathtt{s}}(i,\Pi^t)N \rfloor$ раз в качестве родителя, а дробная часть $\{P_{\mathtt{s}}(i,\Pi^t)N\}$ используется, как вероятность отбора этой особи еще один раз. Очевидно, такой оператор, как и оператор селекции КГА, осуществляет пропорциональную селекцию.

Действие оператора стохастической универсальной селекции можно представлять себе с помощью колеса рулетки, разделенного на $N$ секторов как при селекции в КГА. Пусть в центре колеса через равные угловые промежутки, закреплено $N$ указателей. Поворот колеса рулетки на случайно выбранный угол (с равномерным распределением от 0 до $2\pi$) относительно указателей дает назначение родительских особей $K_1, \ldots, K_N$. В завершение, элементы последовательности $K_1, \ldots, K_N$ переупорядочиваются случайным образом, чтобы снизить вероятность совпадения родительских генотипов на входе следующего за селекцией оператора кроссинговера.

Обозначим число событий, состоящих в выборе особи $i$ при селекции КГА через $Z_{\mathrm{CGA}}(i,\Pi^t)$, а при стохастической универсальной селекции Бэкера – через $Z_{\mathrm{B}}(i,\Pi^t)$. Пусть $D[\cdot]$ обозначает дисперсию случайной величины.

**Упражнение 2.1.** *Показать, что* $D[Z_{\mathrm{B}}(i,\Pi^t)] \leq 1/4$, *в то время как* $D[Z_{\mathrm{CGA}}(i,\Pi^t)]$ *может расти неограниченно с ростом* $N$ *при подходящем выборе популяции* $\Pi^t$.

Как видно из упражнения, оператор стохастической универсальной селекции имеет преимущество по сравнению с селекцией из КГА, обеспечивая меньшие колебания численности потомства каждой особи и, тем самым, более устойчивую работу алгоритма.

**Ранговая селекция.**[7] Оба из рассмотренных выше операторов селекции имеют общее проблемное свойство: как прави-

[6] В англоязычной литературе: Baker's stochastic universal selection.

[7] В англоязычной литературе используется термин *ranking selection*.

ло, в процессе развития популяции ГА происходит приближение приспособленности особей к максимальному значению, и вместе с этим распределение вероятностей $P_{\mathtt{s}}(i, \Pi^t)$ приближается к равномерному. Таким образом, снижается «чувствительность» операторов пропорциональной селекции к различиям в приспособленности особей текущей популяции, что представляется нецелесообразным при решении задачи оптимизации. Один из способов решения этой проблемы был предложен Д. Голдбергом [29] и состоит в преобразовании значений функции приспособленности (см. выражение (12) в примере 5). Альтернативный подход состоит в использовании *ранжирования* особей.

**Определение 2.1.** *Биекция* $r_\Pi : \{1, ..., N\} \to \{1, ..., N\}$ *называется ранжированием для популяции* $\Pi$*, если* $r_\Pi(i) > r_\Pi(j)$ *для всех* $i, j \in \{1, 2, ..., N\}$*, таких что* $\Phi(\xi^i) > \Phi(\xi^j)$*. Значение* $r_\Pi(i)$ *называется рангом особи* $\xi^i$*.*

**Ранговая селекция.**[8] Предложена в работе [30]. Пусть функция $\alpha : \{1, \ldots, N\} \to \mathbb{R}_+$, такая что $\sum_{r=1}^{N} \alpha(r) = 1$. Тогда $\alpha$ называется ранжирующей функцией. При заданной ранжирующей функции оператор селекции с распределением вероятностей

$$P\{\text{выбрать особь с рангом } r\} = \alpha(r), \quad r = 1, \ldots, N,$$

называется ранговой селекцией. Такая селекция не теряет «чувствительности» при сколь угодно малых различиях приспособленности особей.

Частный случай, где

$$\alpha(r) = \frac{\eta - 1}{N} \left( \frac{2(r - N)}{N - 1} + \frac{\eta}{\eta - 1} \right),$$

при $\eta \in (1, 2]$ называется линейным ранжированием. Легко видеть, что условия ранжирующей функции выполняются.

[8] В англоязычной литературе используется термин *ranking selection*.

Особь с рангом $N$ имеет вероятность селекции, равную $\eta/N$, а особь с рангом 1 – вероятность $(2-\eta)/N$. Если положить $\eta = 2$, то особь с рангом 1 имеет нулевую вероятность селекции. Если же $\eta \to 1$, то распределение вероятностей селекции стремится к равномерному.

Ранговая селекция при $\eta = 2 - 2/(N+1)$ эквивалентна применению оператора селекции КГА с подстановкой рангов особей $r_\Pi(i)$ вместо значений их приспособленности.

Следующий из рассматриваемых операторов также предполагает ранжирование особей.

**Турнирная селекция.**[9] В операторе турнирной селекции с размером турнира $s$ (иное название: оператор $s$-турнирной селекции) для отбора очередного родительского решения из текущей популяции независимо извлекаются $s$ особей с равномерным распределением и из них выбирается особь с наибольшим рангом.

Следующее упражнение показывает, что оператор 2-турнирной селекции близок по своим свойствам к ранговой селекции. Обозначим число событий, состоящих в выборе особи $i$ при использовании оператора 2-турнирной селекции через $Z_{\mathrm{T}}(i, \Pi^t)$, а при использовании ранговой селекции – через $Z_{\mathrm{R}}(i, \Pi^t)$. Пусть $E[\cdot]$ обозначает математическое ожидание случайной величины.

**Упражнение 2.2.** *Показать, что* $E[Z_{\mathrm{T}}(i,\Pi^t)] = \frac{2r-1}{N}$, $E[Z_{\mathrm{R}}(i,\Pi^t)] = \frac{2r}{N+1}$, *и* $\frac{D[Z_{\mathrm{R}}(i,\Pi^t)]}{D[Z_{\mathrm{T}}(i,\Pi^t)]} \to \frac{2r}{2r-1}$ *при* $N \to \infty$.

## 2.2. Стратегии управления популяцией

Обновление всей популяции на каждой итерации КГА соответствует подходу, применяемому при имитационном моделировании в популяционной генетике (см., например, [1]), однако,

[9] В англоязычной литературе используется термин *tournament selection*.

для ускорения поиска генотипов с высокой приспособленностью общая схема ГА зачастую модифицируется. Основная мотивация при этом состоит в том, что в КГА даже генотип, существенно превышающий по пригодности все прочие особи популяции, с большой вероятностью будет исключен из рассмотрения уже на следующей итерации после его появления. Однако в успешных приложениях ГА приспособленность потомков, как правило, имеет положительную корреляцию с приспособленностью родительских генотипов. В таких случаях целесообразно сохранять наиболее пригодные особи в течение ряда итераций ГА и генерировать с помощью кроссинговера и мутации оставшуюся часть популяции. Рассмотрим некоторые известные схемы управления популяцией, реализующие этот принцип.

**Элитарная стратегия.** На кажой итерации ГА, во-первых, строится очередная популяция $\Pi^{t+1}$ по правилам КГА. Во-вторых, если по приспособленности все генотипы новой популяции уступают максимально приспособленной (*элитной*) особи $\xi_{\mathtt{e}}^t$ из предыдущей популяции, то один из наименее приспособленных генотипов в $\Pi^{t+1}$ заменяется на $\xi_{\mathtt{e}}^t$.

Таким образом, если ГА применяется для решения задачи безусловной оптимизации (1), то последовательность значений целевой функции элитных фенотипов $f(x(\xi_{\mathtt{e}}^1)), f(x(\xi_{\mathtt{e}}^2)), \ldots$ будет неубывающей. Более того, как показал Г. Рудольф [40], при $x(B) = X$ и $0 < P_{\mathtt{m}} < 1$, дополнение КГА элитарной стратегией обеспечивает сходимость последовательности $f(x(\xi_{\mathtt{e}}^1)), f(x(\xi_{\mathtt{e}}^2)), \ldots$ к оптимальному значению целевой функции задачи (1) почти наверное.

Вместо одной элитной особи в ГА может сохраняться некоторое подмножество генотипов текущей популяции, имеющих высокую приспособленность (см., например, [25]). Такие стратегии называются *частичной заменой популяции*. Следующая стратегия может рассматриваться как предельный случай расширения множества элитных особей.

**Стационарная стратегия управления популяцией.**

При этой стратегии на каждой итерации ГА в популяцию добавляются два генотипа, полученных применением операторов кроссинговера и мутации. Каждая новая особь замещает некоторый «неперспективный» генотип. При этом в качестве «неперспективного» может быть взят генотип с наименьшей приспособленностью, или генотип, выбранный с равномерным распределением среди имеющих приспособленность ниже средней в текущей популяции. В некоторых вариантах ГА на выходе кроссинговера имеется только один генотип – тогда изменяется только одна особь популяции.

Особенностью стационарной стратегии управления популяцией является значительно более быстрое «сужение» области поиска, по сравнению с КГА. В связи с этим, во многих реализациях стационарной стратегии управления популяцией при совпадении новой особи с одной из имеющихся в популяции, новая особь в популяцию не добавляется.

## 2.3. Операторы кроссинговера и мутации

Наряду с оператором одноточечного кроссинговера КГА, в генетических алгоритмах используются и другие операторы рекомбинации родительских генотипов. Общей чертой для всех из них является, так называемое, свойство *передачи генов*: значение для каждого гена потомка выбирается из значений соответствующих генов одного или другого родителей.[10]

В некоторых вариантах кроссинговера результатом является один генотип (см., например, [6]), однако, наиболее распространены операторы с двумя выходными генотипами. Во втором случае, с целью сохранения разнообразия популяции, стремятся построить как можно более удаленные один от другого генотипы потомков.

Пусть даны родительские генотипы $\xi$ и $\eta$, для которых по-

[10] Данное свойство в работах N. Radcliffe назавно *gene transmission* и является частным случаем *allele transmission* (см. [38]).

рождается пара генотипов потомков $\xi'$, $\eta'$. Если в задаче отсутствуют ограничения, или схема представления решений такова, что $x(B) \subseteq D$, тогда уместно использовать, так называемую, *маску кроссинговера*. Под этим термином понимают вспомогательную последовательность $\mathbf{m} = (m_1, \ldots, m_l) \in B$, по которой строятся генотипы потомков:

$$\xi'_i = \begin{cases} \xi_i, & \text{если } m_i = 1 \\ \eta_i, & \text{иначе,} \end{cases} \quad ; \quad \eta'_i = \begin{cases} \eta_i, & \text{если } m_i = 1 \\ \xi_i, & \text{иначе,} \end{cases}$$

для $i = 1, \ldots, l$. Рассмотрим два примера использования маски кроссинговера.

**Равномерный кроссинговер.** При действии этого оператора $i$-ый ген, $i = 1, \ldots, l$, копируется в генотип потомка из $i$-той позиции генотипа одного или другого родителя с равными вероятностями, независимо от выбора других генов. Формально данный оператор определяется выбором маски кроссинговера с равномерным распределением на множестве $B$.

**$k$-точечный кроссинговер.** Данный оператор представляет собой обобщение одноточечного кроссинговера. В строке генотипа выбирается $k$ различных координат скрещивания $0 < \chi_1 < \chi_2 < \ldots < \chi_k < l$ с равномерным распределением среди всевозможных таких наборов. Обозначим $\chi_0 = 0$, тогда маска кроссинговера определяется следующим образом:

$$m_i = \begin{cases} 1, & \text{если } \max\{j : \chi_j < i\} \text{ – четное число} \\ 0, & \text{иначе} \end{cases}$$

для $i = 1, \ldots, l$. Данный оператор имеет то свойство, что при задании четного числа точек скрещивания $k$, первая и последняя координаты одного родителя всегда переходят одному потомку. Наоборот, при нечетном $k$ эти координаты копируются в каждый из генотипов потомков от разных родителей. Если эти свойства крайних координат представляются нежелательными,

то может быть использован, так называемый, *круговой кроссинговер*, при котором строка генотипа рассматривается как кольцо и используется четное число $k$ координат скрещивания.

Каждый из описанных операторов кроссинговера может быть реализован и в варианте с одним генотипом потомка (для этого достаточно отбросить второй генотип). Далее будет рассмотрен оператор кроссинговера, при котором естественным образом строится только один генотип потомка.

**Задача оптимальной рекомбинации.** Пусть решается задача условной максимизации в пространстве двоичных строк длины $n$. Рассмотрим вычислительную сложность задачи отыскания наилучшего по приспособленности генотипа, как результата кроссинговера для заданной пары родительских генотипов при условии выполнения свойства передачи генов. Для простоты постановки этой задачи будем предполагать что операторы построения начальной популяции, кроссинговера и мутации генерируют только генотипы, представляющие допустимые решения.

С учетом свойства передачи генов, сформулируем *задачу оптимальной рекомбинации*: для произвольных заданных родительских генотипов $p^1, p^2$, представляющих допустимые решения, требуется найти представляющий допустимое решение генотип $\xi$, такой что:

1) для каждого $j = 1, \ldots, n$ выполняется $\xi_j = p^1_j$ или $\xi_j = p^2_j$;

2) $\xi$ имеет максимальное значение функции приспособленности среди всех генотипов, удовлетворяющих условию 1).

Далее множество номеров координат, в которых родительские генотипы различны, будем обозначать через $D(p^1, p^2)$.

В качестве примера эффективно разрешимой задачи оптимальной рекомбинации рассмотрим следующую известную задачу из теории графов. Пусть имеется граф $G = (V, E)$ с множеством вершин $V = \{v_1, \ldots, v_|V|\}$ и множеством ребер $E$. Задача о наибольшем независимом множестве состоит в отыскании та-

кого подмножества $S \subseteq V$, что ни одно ребро $e \in E$ не инцидентно сразу двум вершинам из $S$ (т.е. $S$ – независимое множество) и мощность этого множества максимальна.

Естественным будет представление решений с помощью вектора-индикатора из $\{0,1\}^n$, где $n = |V|$, и $\xi_j = 1$ тогда и только тогда, когда вершина $v_j$ принадлежит искомому подмножеству. Пусть $\Phi(\xi) = |x(\xi)|$ для любого допустимого решения $x(\xi)$. Как замечено в работе Э.Балаша и В.Нихауса [20], при использовании данного представления решений задача оптимальной рекомбинации разрешима за полиномиальное время.

Для того чтобы в этом убедиться, рассмотрим произвольные родительские независимые множества $S_1$ и $S_2$ и соответствующие им генотипы $p_1$ и $p_2$. Исходя из свойства передачи генов, решение-потомок $S$ должно содержать все множество вершин $L = S_1 \cap S_2$, кроме того, в $S$ не должно быть элементов множества $V \setminus (S_1 \cup S_2)$, а вершины с номерами из множества $D(p^1, p^2)$ необходимо выбрать оптимальным образом. Последнее требование формулируется, как задача о наибольшем независимом множестве в подграфе, порожденном множеством вершин с номерами из $D(p^1, p^2)$. Легко видеть, что данный подграф является двудольным.

Для отыскания наибольшего независимого множества в двудольном графе $H = (V', E')$ можно воспользоваться тем фактом, что наибольшее независимое множество всегда является дополнением наименьшего вершинного покрытия $C'$, то есть такого наименьшего по мощности множества вершин, что каждое ребро инцидентно хотя бы одной из них.

Задача о наименьшем вершинном покрытии двудольного графа $H = (V', E')$ эффективно разрешима с помощью алгоритма построения минимального разреза во вспомогательном графе, состоящем из графа $H$ и дополнительных вершины-источника $v_0$ и вершины-стока $v_{n+1}$. Источник $v_0$ соединяется со всеми вершинами одной доли, а сток $v_{n+1}$ – со всеми вершинами другой доли. Ребрам из множества $E'$ приписываются

бесконечные пропускные способности, а ребрам, инцидентным дополнительным вершинам – единичные пропускные способности. Наименьшее вершинное покрытие $C'$ формируется из вершин, инцидентных ребрам минимального разреза.

Генотип $\xi$, являющийся вектором-индикатором множества $L \cup (V' \setminus C')$ представляет собой решение задачи оптимальной рекомбинации для задачи о наибольшем независимом множестве.

**Упражнение 2.3.** *Показать, что для задачи о наименьшем вершинном покрытии при тех же предположениях о способе представления решений (т.е. $v_j \in C \Leftrightarrow \xi_j = 1$) задача оптимальной рекомбинации эффективно разрешима.*

Обзор по вычислительной сложности задач оптимальной рекомбинации, порожденных различными известными задачами комбинаторной оптимизации и алгоритмам их решения приводится в [26, 27]. Рассмотренная здесь постановка задачи оптимальной рекомбинации не является единственной возможной – см., например, [4, 6]. Выбор наиболее подходящей формулировки этой подзадачи и методов ее решения делается на основе вычислительного эксперимента.

## 2.4. Недвоичные представления решений

Существует большое число задач оптимизации, в которых наиболее естественным способом представления решений является вектор целых или рациональных чисел, перестановка элементов некоторого множества или его разбиение на подмножества. Во многих из этих случаев представление решений посредством двоичных генотипов не имеет смысла и рассматриваются более подходящие *недвоичные* представления с пространством генотипов

$$B = A_1 \times A_2 \times \ldots \times A_l,$$

$A_1, \ldots, A_l$ – некоторые конечные множества символов.

Недвоичное представление, в частности, может быть использовано для кодировки решений задачи оптимизации в евклидовом пространстве $X = R^q$, как в примере 4 из п. 1.2. Вместо двоичного представления для каждой координаты $x_i,\ i = 1, \ldots, q$, при этом используется формат с плавающей запятой. Успешные результаты на общедоступных тестовых примерах демонстрируют алгоритмы *дифференциальной эволюции* (см., например, [22]), которые могут рассматриваться как модификации ГА, использующие формат с плавающей запятой и специфический оператор мутации [41], где учитываются разности между векторами из текущей популяции.

Также недвоичные представления используются с целью обеспечения допустимости решения $x(\xi)$ для всякого генотипа $\xi \in B$. Пример такого представления решений для задачи о наименьшем покрытии множества содержится в п. 3.2.

Широкое применение недвоичные представления находят в ГА для задач, где множество допустимых решений составляют перестановки. Рассмотрим в качестве примера некоторые операторы ГА, предложенные для задачи коммивояжера.

Пусть дан полный граф $G = (V, E)$ с нумерацией вершин и указана длина $d(e)$ для каждого ребра $e \in E$. Требуется найти кратчайший обход, то есть цикл, проходящий по всем вершинам графа и имеющий минимальную длину. В данном случае в качестве недвоичного представления для обхода может быть выбран целочисленнвй вектор из $l = |V| - 1$ чисел, где $\xi_i$ содержит номер $i$-той вершины при обходе, начиная с вершины $v_1$.

**Оператор кроссинговера с частичным отображением.** Данный оператор был предложен в работе Д. Голдберга и Р. Лингле [31] и кратко обозначается PMX.[11] Рассмотрим действие кроссинговера PMX на иллюстративном примере.

Пусть даны следующие родительские генотипы с координатами скрещивания $\chi_1 = 3, \chi_2 = 7$:

[11] От английского *partially mapped crossover*.

$$\begin{array}{ll} \xi=( & 1 \quad 2 \quad 3 \mid 4 \quad 5 \quad 6 \quad 7 \mid 8 \quad 9) \\ \eta=( & 4 \quad 5 \quad 2 \mid 1 \quad 8 \quad 7 \quad 6 \mid 9 \quad 3). \end{array}$$

Сначала выполняется обмен средними участками генотипов, прочие гены при этом считаются неопределенными:

$$\begin{array}{l} (\ \mathrm{x} \quad \mathrm{x} \quad \mathrm{x} \mid 1 \quad 8 \quad 7 \quad 6 \mid \mathrm{x} \quad \mathrm{x}) \\ (\ \mathrm{x} \quad \mathrm{x} \quad \mathrm{x} \mid 4 \quad 5 \quad 6 \quad 7 \mid \mathrm{x} \quad \mathrm{x}). \end{array}$$

Далее, для каждого из неопределенных значений проверяется, можно ли в этой позиции оставить прежнее значение. Например, в первом из указанных генотипов нельзя оставить 1 на первой позиции, т.к. 1 уже зафиксирована в четвертом гене, однако можно оставить на месте значения 2, 3 и 9. Аналогично, во втором генотипе можно оставить на месте 2, 9 и 3:

$$\begin{array}{l} (\ \mathrm{x} \quad 2 \quad 3 \mid 1 \quad 8 \quad 7 \quad 6 \mid \mathrm{x} \quad 9) \\ (\ \mathrm{x} \quad \mathrm{x} \quad 2 \mid 4 \quad 5 \quad 6 \quad 7 \mid 9 \quad 3). \end{array}$$

Наконец, остальные значения заполняются такими же попарными обменами, какими изменялись средние участки, только теперь обмен происходит не между генотипами, а внутри каждого из них. В рассматриваемом примере 4 меняется на 1, 5 – на 8, 6 – на 7, 7 – на 6. Таким образом, результат кроссинговера имеет вид:

$$\begin{array}{ll} \xi'=( & 4 \quad 2 \quad 3 \mid 1 \quad 8 \quad 7 \quad 6 \mid 5 \quad 9) \\ \eta'=( & 1 \quad 8 \quad 2 \mid 4 \quad 5 \quad 6 \quad 7 \mid 9 \quad 3). \end{array}$$

Как показали эксперименты, кроссинговер PMX показывает хорошие результаты в ряде задач составления расписаний, в то время как для задачи коммивояжера лучшие результаты показали операторы, основанные на наследовании свойства смежности вершин. Пример ГА с использованием кроссинговера PMX для задачи управления поставками продукции приводится в п. 3.3.

Рассмотрим несколько операторов мутации, применяемых в задаче коммивояжера и других задачах, где допустимыми решениями являются перестановки. При описании этих операторов под случайным выбором понимается выбор с равномерным распределением среди всех возможных вариантов.

**Мутация обмена**[12] состоит в обмене пары генов из случайно выбранных позиций в данной на вход перестановке. С точки зрения локального поиска для задачи коммивояжера, при действии данного оператора выполняется шаг в случайно выбранную точку из окрестности *2-city swap* [11].

**Мутация сдвига**[13] состоит в перемещении гена из случайно выбранной позиции на случайное число позиций влево или вправо. Содержимое всех промежуточных генов при этом сдвигается на одну позицию.

**Мутация «2-замена».** Последний пример оператора мутации определяется наиболее просто в случае задачи коммивояжера в терминах фенотипов, то есть обходов графа $G$. В обходе, заданном входным генотипом случайным образом выбираются два несмежных ребра и заменяются двумя новыми ребрами, которые в данном случае определяются однозначно. С точки зрения локального поиска, действие данного оператора представляет собой шаг в случайно выбранную точку из окрестности, определенной относительно 2-замены [14].

**Упражнение 2.4.** *Описать алгоритм, осуществляющий мутацию «2-замена» в указанной выше недвоичной кодировке решений задачи коммивояжера.*

[12] В англоязычной литературе принят термин *exchange mutation*.
[13] В англоязычной литературе принят термин *shift mutation*.

# 3. Гибридные генетические алгоритмы

На практике приведенные выше схемы генетических алгоритмов, как правило, используются в комбинации с некоторыми эвристическими процедурами, учитывающими специфику решаемой задачи. Среди таких процедур наиболее часто используются локальная оптимизация,[14] «жадные» алгоритмы корректировки решений и декодирующие эвристики. Ниже описываются основные принципы работы этих процедур.

## 3.1. Локальная оптимизация

Рассмотрим метод, состоящий в применении процедуры локальной оптимизации к каждому новому решению, полученному в результате мутации и кроссинговера. Улучшенное таким образом решение кодируется как генотип и добавляется в популяцию ГА. Пусть для всякого элемента $y \in D$ определена некоторая его *окрестность* $N(y) \subseteq D$. В качестве окрестности зачастую выбирается множество точек, находящихся в пределах заданного расстояния от $y$, например, в метрике Хэмминга.

Процедура локальной оптимизации предназначена для отыскания *локального оптимума*, то есть такого допустимого решения $y'$, в окрестности которого нет решений лучших его по целевой функции. Работа процедуры локальной оптимизации состоит в последовательном переходе от решения к решению, таким образом, что каждое новое решение является наилучшим по целевой функции в окрестности предыдущего,[15] или по крайней мере улучшает предыдущее решение в смысле целевой функции. Во многих случаях поиск по окрестности $N(y)$ выполняется перебором ее элементов. Тогда хорошие результаты дает упрощенная эвристика, в которой из текущего решения $y$ выполняется переход в первое найденное решение, улучшающее зна-

[14] ГА с локальной оптимизацией в англоязычной литературе принято называть *меметическим алгоритом (memetic algorithm)*.

[15] В англоязычной литературе используется термин *best improving move*.

чение $f(y)$.[16] Последнее полученное таким образом решение $y'$ (т.е. уже не улучшаемое в своей окрестности) является *локальным оптимумом*. Подробнее о методах поиска локальной оптимизации см., например, в [11, 14].

Мотивацией для использования данного метода является тот факт, что множество локальных оптимумов содержит, в частности, и глобальный оптимум. Поэтому естественно предполагать, что организация поиска на множестве локальных оптимумов или в окрестности этого множества увеличит шансы получить глобальный оптимум. В наибольшей степени описанная эвристика подходит для тех ситуаций, когда вычисление значения $f(y')$, $y' \in N(y)$ значительно упрощается при известном значении $f(y)$.

Рассмотрим в качестве примера задачу о максимальном разрезе, сформулированную в п. 1.2. Пусть окрестностью разреза $y = \{U_y, U'_y\}$ является множество всех тех разрезов $\{U, U'\}$, в которых $U$ или $U'$ отличается от $U_y$ не более чем $k$ вершинами, где $k$ – константа. В этом случае вычисление веса очередного разреза из окрестности осуществимо за $O(|V|)$ операций, в то время как непосредственное вычисление веса разреза требует $O(|E|)$ операций.

Как показала практика, во многих задачах отыскание локального оптимума после построения каждой новой особи оказывается слишком трудоемким и замедляет работу алгоритма. В таких случаях локальная оптимизация может применяться только к отдельным особям, либо вместо стандартного локального поиска, требующего многократных переходов к улучшающим решениям, для каждой новой особи выполняется только один шаг такой процедуры [24], так называемая локальная корректировка решений.

[16] *First improving move.*

### 3.2. «Жадные» алгоритмы корректировки решений

Данный метод подобен локальной оптимизации решений, но отличается тем, что к вновь полученному решению вместо локальной оптимизации применяют некоторый «жадный» алгоритм. Как правило, для этого требуется представить задачу в виде отыскания набора или размещения элементов из некоторого заданного множества $U$. Для задач такого типа известно большое число «жадных» алгоритмов, работа которых начинается с пустого множества и на каждой итерации к имеющемуся (возможно, еще недопустимому) набору или размещению добавляется новый элемент с использованием некоторого эвристического правила. Существуют также «жадные» алгоритмы, работа которых начинается с заданной совокупности элементов, и на каждой итерации из имеющегося набора исключается элемент с наименьшей «ценностью».

Для корректировки генотипа $\xi$, полученного на выходе операторов мутации и кроссинговера, к набору элементов $J(\xi)$, соответствующих $\xi$, применяется «жадный» алгоритм. В некоторых случаях в «жадном» алгоритме целесообразно использовать только элементы из $J(\xi)$ (см., например, [6]). В других случаях [21] в процессе работы «жадного» алгоритма в множество $J(\xi)$ добавляются элементы из $U \backslash J(\xi)$.

В качестве примера рассмотрим один из «жадных» алгоритмов корректировки решений, использованный в ГА для задачи о наименьшем покрытии множества [6]. Данная задача имеет следующую формулировку. Пусть даны множество $M = \{1, ..., m\}$ и набор его подмножеств $M_j \subseteq M$, где $j \in U = \{1, ..., n\}$. Подмножество $J \subseteq U$ называется *покрытием* $M$, если $\bigcup\limits_{j \in J} M_j = M$. Каждому $M_j$ приписана стоимость (или вес) $c_j > 0$, $c_j \in \mathbb{R}$. Требуется найти покрытие минимальной суммарной стоимости.

Рассматриваемый генетический алгоритм основан на недвоичном представлении решений, благодаря которому обеспечивается допустимость фенотипа при любом генотипе. Обозначим

через $U_i$ множество индексов подмножеств $M_j$, покрывающих элемент $i \in M$, т.е. $U_i = \{j : i \in M_j\}$. В данном ГА генотип $\xi$ состоит из $m$ генов $\xi_1, \xi_2, ..., \xi_m$, причем $\xi_i \in U_i$, где $i = 1, 2, ..., m$, т.е. каждый ген кодирует одно из множеств, покрывающих соответствующий элемент. Генотипу $\xi$ сопоставляется покрытие (фенотип) $J(\xi)$, содержащее номера всех тех покрывающих подмножеств, которые закодированы хотя бы в одном из генов $\xi$.

В качестве оператора рекомбинации может использоваться, например, равномерный кроссинговер. При мутации в случайно выбранном подмножестве генов выполняется замена их значений некоторым случайным образом. Для эффективной работы данного ГА целесообразно применить оператор корректировки решений, исключающий из генотипа $\xi$, полученного в результате кроссинговера и мутации, некоторые «лишние» подмножества $M_j$. С этой целью в ГА [6] используются три «жадные» эвристики, каждая из которых находит некоторое приближенное решение подзадачи наименьшего покрытия множества $M$ посредством подмножеств с номерами из $J(\xi)$. Рассмотрим одну из этих эвристик.

Данная эвристика обозначается Lmin, ввиду того что результатом ее работы является лексикографически минимальное из покрытий, содержащихся в подмножестве $J(\xi)$. Lmin начинает работу с данного на вход покрытия $J(\xi)$, и, просматривая его элементы в порядке возрастания номеров, исключает те из них, удаление которых не нарушает допустимости решения. Предполагается, что изначально элементы множества $U$ упорядочены по невозрастанию их стоимости.

**Алгоритм Lmin**

**1.** Положить $J := J(\xi)$.
**2.** Для всех $j$ из $J$ в порядке возрастания индексов выполнять:
**2.2.** Если $J\backslash\{j\}$ - покрытие, то положить $J := J\backslash\{j\}$.
**3.** Для всех $i \in M$ в результирующем генотипе $\eta$ положить
$\eta_i := j$, где $j \in U_i \cap J$.

На шаге 3 здесь строится генотип потомка, кодирующий полученное покрытие. После применения процедур локальной корректировки решения полученный генотип добавляется в популяцию. Данный подход принято называть *ламарковой эволюцией.*[17]

Отметим, что в некоторых реализациях генетических алгоритмов с локальной оптимизацией решений или с использованием «жадных» эвристик корректировки оказывается более выгодно добавлять в популяцию исходный генотип, полученный сразу после рекомбинации и мутации, вместо скорректированного решения [42]. В качестве значения приспособленности для такого генотипа принимается приспособленность особи, полученной в результате корректировки. В этом случае говорят, что в ГА реализована *балдвинова эволюция.*[18] Как правило, на начальном этапе такой ГА затрачивает большее число итераций чем при ламарковой эволюции для получения решений того же качества, однако, при достаточно большом времени работы балдвинова эволюция может иметь преимущество. Балдвинову эволюцию можно также рассматривать как частный случай ГА с декодирующей эвристикой (см. п. 3.3).

[17] По Ж.Б.Ламарку, интенсивно функционирующие органы развиваются, а не находящие употребления ослабевают, и эти функционально-морфологические изменения передаются по наследству. Законы Ламарка были опровергнуты открытиями генетики в начале 20 в.

[18] Дж. Балдвин предложил механизм, согласно которому первоначально приобретенные навыки организмов могут в дальнейшем стать наследуемыми. На первом этапе организмы (благодаря соответствующим мутациям) получают свойство обучиться некоторому полезному навыку. Приспособленность таких организмов увеличивается и они распространяются по популяции. На втором этапе приобретенный полезный навык может быть «повторно изобретен» генетической эволюцией, в результате чего он записывается непосредственно в геном и становится наследуемым.

### 3.3. Декодирующие эвристики

Для многих задач дискретной оптимизации имеется возможность декомпозиции исходной задачи на ряд более простых подзадач, причем получающиеся подзадачи оказываются однотипными по своей постановке и могут быть эффективно решены точно или приближенно. К числу задач рассматриваемого класса относятся такие, где требуется найти оптимальный набор или размещение элементов из некоторого заданного множества, задачи составления расписаний, маршрутизации перевозок и др. Как правило, решение подзадач может быть осуществлено с помощью одного или нескольких альтернативных «жадных» эвристических правил, в результате чего к имеющемуся (возможно, еще недопустимому) частичному решению добавляется один или более новых элементов. При этом предшествующие подзадачи могут давать результаты, затрудняющие или упрощающие решение последующих. Даже оптимальные решения отдельных подзадач зачастую оказываются не рациональными с точки зрения исходной задачи в целом.

Существенное повышение точности окончательного решения может быть достигнуто за счет подбора последовательности эвристик, применяемых для решения подзадач или за счет выбора порядка обработки элементов решения. Для поиска наиболее подходящего упорядочивания может быть использован генетический алгоритм, где генотип содержит информацию о последовательности применения эвристик или обработки элементов решения. Таким образом, «жадные» эвристики осуществляют декодирование информации, записанной в генотипе. Данный подход был, в частности, успешно применен к задачам составления расписаний, двумерной упаковки, распределения заданий по локальной вычислительной сети [13].

В качестве примера рассмотрим гибридный алгоритм, основанный на ГА и «жадной» эвристике для *задачи управления поставками продукции*, представляющей собой обобщение классической транспортной задачи.

Пусть количество поставщиков равно $k$, а количество потребителей – $m$. Обозначим через $M_i$ максимальный объем продукции, который способен предоставить поставщик $i$, $i = 1, ..., k$, $A_j$ – объем продукции, необходимый потребителю $j$, $j = 1, ..., m$. Кроме того, если осуществляется некоторая поставка от поставщика $i$ потребителю $j$, то она не может быть меньше заданного значения $m_{ij}$. Модель частично целочисленного линейного программирования для этой задачи выглядит следующим образом:

$$F(z,x) = \sum_{i=1}^{k}\sum_{j=1}^{m}(a_{ij}z_{ij} + c_{ij}x_{ij}) \to \min \tag{14}$$

$$\sum_{j=1}^{m} x_{ij} \le M_i,\ i = 1, \ldots, k, \tag{15}$$

$$\sum_{i=1}^{k} x_{ij} = A_j,\ j = 1, \ldots, m, \tag{16}$$

$$z_{ij} \in \{0, 1\},\ i = 1, \ldots, k, j = 1, \ldots, m, \tag{17}$$

$$m_{ij}z_{ij} \le x_{ij} \le M_i z_{ij},\ i = 1, \ldots, k, j = 1, \ldots, m. \tag{18}$$

Булевы переменные $z_{ij}$ указывают на наличие или отсутствие поставки от поставщика $i$ потребителю $j$, а вещественные переменные $x_{ij}$ обозначают ее объем.

С помощью полиномиальной сводимости от NP-полной задачи РАЗБИЕНИЕ [5] нетрудно показать, что задача поиска допустимого решения для системы (15)-(18) является NP-трудной даже при $m = 1$. В случае, когда все $m_{ij}$ равны нулю, рассматриваемая задача является хорошо известной транспортной задачей с фиксированными доплатами [2].

Рассмотрим генетический алгоритм GAgrd, основанный на идее последовательного решения задачи управления поставками для каждого потребителя в отдельности. В алгоритме декодирования потребители рассматриваются поочередно и каждому из них назначаются поставки из оставшегося у поставщиков

объема продукции. В процессе работы ГА осуществляется поиск наилучшего порядка просмотра потребителей. Так как рассматривается задача минимизации, то в качестве функции приспособленности можно принять, например, величину $1/F(z,x)$ при условии $F(z,x) > 0$. Пусть порядок задается с помощью перестановки индексов потребителей $\sigma = (j_1, ..., j_m)$. Тогда решение может быть построено по генотипу $\xi = \sigma$, используя следующую процедуру:

**Декодирование($\sigma$)**
Для $k := 1$ до $m$ выполнять:
**1.** Решить задачу для одного потребителя $j_k$,
т.е. получить вектор $x_{ij_k},\ i = 1, ..., k$.
**2.** Пересчитать объемы продукции: $M_i := M_i - x_{ij_k}$.

Задача с одним потребителем приближенно решается с помощью «жадного» алгоритма, схема которого будет приведена ниже. Полученное решение может оказаться недопустимым – в этом случае ему назначается штраф.

Аналогичная задача поиска наилучшей перестановки возникает при решении задачи коммивояжера. Для задачи коммивояжера известно множество успешных реализаций генетического алгоритма, и алгоритм GAgrd можно рассматривать как адаптацию одного из них для задачи управления поставками. GAgrd был реализован с использованием турнирной селекции и стационарной схемы управления популяцией. В качестве мутации был выбран оператор обмена, в качестве кроссинговера – оператор PMX (см. п. 2.3).

Опишем алгоритм решения задачи с одним потребителем, т.е. в случае $m = 1$. Обозначим для краткости $A = A_1,\ c_i = c_{i1},\ a_i = a_{i1},\ m_i = m_{i1},\ x_i = x_{i1}$.

**Жадный алгоритм для случая $m = 1$:**
Пока $A > 0$ и $\{i \mid M_i \geq m_i\} \neq \emptyset$, выполнять:
**1.** $S_i := \min\{\max\{A, m_i\}, M_i\}$ для $1 \leq i \leq k$, таких что $M_i \geq m_i$.
**2.** Выбрать $1 \leq i \leq k$, для которого $M_i \geq m_i$ и величина
$(a_i + c_i S_i)/S_i$ минимальна.
**3.** Положить $x_i := S_i,\ A := A - x_i,\ M_i := M_i - x_i$.

На каждой итерации жадного алгоритма добавляется наиболее дешевая поставка в относительном выражении за единицу продукции. После завершения работы жадного алгоритма применяются простейшие процедуры удаления излишнего объ-

ема продукции, который может появиться за счет ограничения $x_i \geq m_i$ при $x_i > 0$. Если же по завершении жадного алгоритма окажется, что $A \neq 0$, то полученное решение является недопустимым, и соответствующему генотипу, в качестве значения функции приспособленности, назначается величина $\alpha/|A|$, где $\alpha$ достаточно мало (штрафной коэффициент). Заметим, что не каждое решение задачи (14)–(18) может быть получено с помощью подходящей перестановки $\sigma$.

Экспериментальные исследования алгоритма GAgrd показали, что использование специализированного декодера дает значительное преимущество как по времени, так и по качеству решений в сравнении с непосредственным применением двоичного представления в задаче управления поставками [3].

## 3.4. Меметические алгоритмы

Меметический алгоритм представляет собой гибридизацию метода поиска на основе эволюционного алгоритма, такого как генетический алгоритм, и одного или нескольких методов локального поиска и/или декодирующих эвристик (см., например, [36]). Меметические алгоритмы являются наиболее известными представителями методов из области меметических вычислений [35], которые возникли благодаря комбинации методов из нескольких отраслей информатики (эволюционные вычисления, многокритериальная оптимизация, машинное обучение, приближенные алгоритмы и др.), а также переноса методов из других наук, таких как биология, социология и физика. Локальный поиск может использоваться на разных этапах меметического алгоритма: на этапе инициализации, постобработки, или чередоваться с процедурами эволюционных алгоритмов.

Координация компонентов меметических алгоритмов часто осуществляется с помощью адаптивных правил и методов машинного обучения, учитывающих приспособленность особей и расстояние между ними [35, 34].

### 3.5. Точные гибридные алгоритмы

Кроме перечисленных выше схем гибридизации генетических алгоритмов с другими приближенными методами, необходимо отметить большое практическое значение комбинации этих алгоритмов с такими точными методами, как метод ветвей и границ, перебор $L$-классов и декомпозиция Бендерса (см., например, [2, 9, 10, 17, 25]).

Рассмотрим для примера комбинацию ГА и точного алгоритма ветвей и границ Лэнд и Дойг [17] для задачи целочисленного линейного программирования на максимум (8)–(10) из п. 1.2. Заметим, что процессе работы алгоритма ветвей и границ происходит уточнение *рекорда*, то есть оценки снизу для оптимального значения целевой функции, причем увеличение точности этой оценки способствует ускорению алгоритма.

Работа гибридного алгоритма начинается с получения верхней оценки $U$ для оптимального значения целевой функции решением ЛП-релаксации исходной задачи. Далее, выполняется поиск приближенного решения с помощью ГА. Если на некоторой итерации $t$ в ГА выполнится равенство $U = f(x(\tilde{\xi}^t))$, это означает оптимальность найденного решения и работа гибридного алгоритма завершается. В противном случае ГА продолжает свою работу до срабатывания другого критерия остановки, например, достижения некоторого заданного числа итераций или ограничения по времени счета.

После завершения первого этапа начинает работу точный алгоритм ветвей и границ. При этом, если в ГА найдено некоторое допустимое решение $x(\tilde{\xi}^{t_{max}})$, то величина $\rho = f(x(\tilde{\xi}^{t_{max}}))$ используется в качестве рекорда. В случае целочисленных коэффициентов целевой функции $c_1, ..., c_q$ можно положить $\rho = f(x(\tilde{\xi}^{t_{max}})) + 1$. Результатом работы гибридного алгоритма является оптимальное решение.

Аналогичный подход может применяться для гибридизации ГА с методами перебора $L$-классов [9, 25] и декомпозиции Бендерса [10]. При этом работа ГА и точного алгоритма может осу-

ществляться параллельно, с обменом наилучшими найденными решениями [25]. Вычислительные эксперименты с гибридными алгоритмами описанного типа показали, что применение ГА в комбинации с точными методами позволяет существенно сократить время поиска оптимального решения и полное время работы алгоритма.

## Контрольные вопросы

1) В чем заключается аналогия работы генетического алгоритма с развитием биологической популяции?

2) Каковы основные принципы работы генетического алгоритма?

3) Каким образом строятся генотипы потомков в классическом генетическом алгоритме?

4) Что регулируется с помощью параметров $P_{\mathtt{m}}$ и $P_{\mathtt{c}}$ в классическом генетическом алгоритме?

5) Сформулируйте задачу о максимальном разрезе графа.

6) В чем отличие стационарной стратегии управления популяцией от схемы КГА?

7) В чем состоит преимущество селекции, основанной на ранжировании, по сравнению с селекцией из КГА?

8) Предложите свой вариант кодировки решений для задачи коммивояжера.

9) Сформулируйте задачу управления поставками продукции.

10) Предложите генетический алгоритм для выяснения содержит ли граф гамильтонов цикл, т.е. цикл, проходящий через каждую вершину в точности один раз.

## Список литературы


[1] *Алтухов Ю.П.* Генетические процессы в популяциях. – М.: Академкнига, 2003.

[2] *Бахтин А.Б., Колоколов А.А., Коробкова З.В.* Дискретные задачи производственно-транспортного типа. – Новосибирск: Наука, 1978.

[3] *Борисовский П.А.* Генетические алгоритмы для задачи о поставках продукции // Материалы V Междунар. науч.-техн. конф. "Динамика систем, механизмов и машин". – Омск: Изд-во ОмГТУ, 2004, Кн. 2. – С.255-258.

[4] *Борисовский П.А., Еремеев А.В.* Генетический алгоритм для задачи о вершинном покрытии графа // "Математика и информатика: наука и образование". Межвузовский сборник научных трудов. Вып. 7. – Омск: Изд-во ОмГПУ, 2008. – С. 49-54.

[5] *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982.

[6] *Еремеев А.В.* Генетический алгоритм для задачи о покрытии // Дискретный анализ и исследование операций. Серия 2, 2000, Т. 7, № 1, – С.47-60.

`http://iitam.omsk.net.ru/~eremeev/papers_r.htm`

[7] *Ивахненко А.Г.* Системы эвристической самоорганизации в технической кибернетике. – Киев: Техника, 1971.

[8] *Карманов В.Г.* Математическое программирование. – М.: Наука, 1986.

[9] *Колоколов А.А.* Регулярные разбиения и отсечения в целочисленном программировании // Сиб. журн. исслед. операций. – Новосибирск, 1994, Т. 1, № 2. – С.18-39.

[10] *Колоколов А.А., Леванова Т.В.* Алгоритмы декомпозиции и перебора $L$-классов для решения некоторых задач размещения // Вестник Омского университета. - Омск: ОмГУ, 1996. – № 1. – С.21-23.
`http://www.omsu.omskreg.ru/vestnik/index_ru.html`

[11] *Кочетов Ю.А.* Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения: Сборник лекций молодежных научных школ по дискретной математике и ее приложениям. – М.: Изд-во центра прикладных исследований при механико-математическм факультете МГУ, 2001. – С 84-117.
`http://math.nsc.ru/LBRT/k5/Kochetov/publ-rus.html`

[12] *Леванова Т.В., Лореш М.А.* Алгоритмы муравьиной колонии и имитации отжига для задачи о $p$-медиане // Автоматика и телемеханика, 2004, № 3. – С.80-88.

[13] *Норенков И.П.* Эвристики и их комбинации в генетических методах дискретной оптимизации // Информационные технологии, 1999, № 1. – С. 2-7.

[14] *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. – М.: Мир, 1985.

[15] *Растригин Л.А.* Статистические методы поиска. – М.: Наука, 1968.

[16] *Рутковская Д., Пилиньский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. – М.: Горячая линия – Телеком, 2006.

[17] *Схрейвер А.* Теория линейного и целочисленного программирования. Т. 2. – М.: Мир, 1991.

[18] *Тутевич В.Н.* Телемеханика. – М.: Высшая школа, 1984.

[19] *Фогель Л., Оуэнс А., Уолш М.* Искусственный интеллект и эволюционное моделирование. – М.: Мир, 1969.

[20] *Balas E., Niehaus W.* Optimized Crossover-Based Genetic Algorithms for the Maximum Cardinality and Maximum Weight Clique Problems // Journ. of Heuristics. – 1998. – Vol. 4, № 4. – P. 107-122.

[21] *Beasley J.E., Chu P.C.* A Genetic Algorithm for the Set Covering Problem // European J. Oper. Res. – 1996. – Vol. 94, № 2. – P. 394-404.

[22] *Brest J., Maučec M.S., Bočsković B.* Il-shade: Improved L-shade algorithm for single objective real-parameter optimization // In: Proc. of 2016 IEEE Congress on Evolutionary Computation (CEC'16). – IEEE, 2016. – P. 1188-1195.

[23] *Corus D., Dang D.-C., Eremeev A.V., Lehre P.K.* Level-based analysis of genetic algorithms and other search processes // IEEE Transactions on Evolutionary Computation. – 2018. – Vol. 22, Issue 5. –P. 707-719.

[24] *Dolgui A., Eremeev A.V., Sigaev V.S.* HBBA: hybrid algorithm for buffer allocation in tandem production lines // Journal of Intelligent Manufacturing. – 2007. – Vol. 18, № 3. – P. 411-420.

[25] *Eremeev A.V., Kolokolov A.A.* On Some Genetic and L-class Enumeration Algorithms in Integer Programming // Proc. of The First International Conference on Evolutionary Computation and its Applications. – Moscow, 1996. – P. 297-303.

[26] *Eremeev A.V., Kovalenko J.V.* Optimal recombination in genetic algorithms for combinatorial optimization problems:

Part I // Yugoslav Journal of Operations Research. – 2014. – Vol. 24, № 1. – P. 1-20.

[27] *Eremeev A.V., Kovalenko J.V.* Optimal recombination in genetic algorithms for combinatorial optimization problems: Part II // Yugoslav Journal of Operations Research. – 2014. – Vol. 24, № 2. – P. 165-186.

[28] *Faizullin R.T.* An approximations for genetic algorithms and star's pattern. In: Proc. of The First Online Workshop on Soft Computing, Nagoya, 1996. – P. 77-79. `http://www.pa.info.mie-u.ac.jp/bioele/wsc1 /papers/p006.html`

[29] *Goldberg D.E.* Genetic Algorithms in Search, Optimization and Machine Learning. - Reading: Addison Wesley, 1989.

[30] *Goldberg D. E. and Deb K.* A comparative analysis of selection schemes used in genetic algorithms. In: Foundations of Genetic Algorithms, Morgan Kaufmann, 1991. – P. 69–93.

[31] *Goldberg D.E., Lingle R.* Alleles, Loci and the Traveling Salesman Problem. In: Proceedings of International Conference on Genetic Algorithms and Their Applications, J.J. Grefenstette (ed.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1985. – P. 154-159.

[32] *Holland J.H.* Adaptation in Natural and Artificial systems. – University of Michigan Press, Ann Arbor, MI, 1975.

[33] *Koza J.R.* Genetic Programming II : Automatic Discovery of Reusable Programs (Complex Adaptive Systems). – MIT Press, 1994.

[34] *Neri F., Toivanen J., Cascella G.L., Ong Y.S.* An adaptive multimeme algorithm for designing HIV multidrug therapies // IEEE/ACM Transactions on Computational Biology and Bioinformatics. – 2007. – Vol. 4. – P. 264-278.

[35] *Neri F., Cotta C.* Memetic algorithms and memetic computing optimization: A literature review // Swarm and Evolutionary Computation. – 2012. – Vol. 2. 1-14.

[36] *Neri F., Cotta C., Moscato P.* Handbook of Memetic Algorithms. – Springer-Verlag, Berlin, 2012.

[37] *Rechenberg I.* Evolutionsstrategie:Optimerung Technischer Systeme nach Prinzipen der Biologischen Evolution. – Formann-Holzboog Verlag, Stuttgart, 1973.

[38] *Radcliffe, N. J.* Forma analysis and random respectful recombination. In: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, 1991. – P. 31-38.
`http://users.breathe.com/njr/formaPapers.html`

[39] *Reeves C., Rowe J.* Genetic algorithms – principles and perspectives. A guide to GA theory. Kluwer Academic Publishers, 2003.

[40] *Rudolph, G.* Finite Markov Chain Results in Evolutionary Computation: A Tour d'Horizon // Fundamenta Informaticae. Vol. 35. № 1-4. – 1998. – P. 67-89.
`http://citeseer.ist.psu.edu/article`
`/rudolph98finite.html`

[41] *Storn, R., Price, K.* Differential evolution - A simple and effcient heuristic for global optimization over continuous spaces // Journal of Global Optimization. – Vol. 11, № 4. – 1997. – P. 341-359. (1997)

[42] *Whitley D.L., Gordon V.S. and Mathias K.E.* Lamarckian Evolution, The Baldwin Effect and Function Optimization. In: Proc. Parallel Problem Solving from Nature – PPSN III, Y.Davidor, H.-P. Schwefel and R.Männer, eds. Springer. Berlin. 1994. – P. 6-15.