# Вариации эволюционных алгоритмов

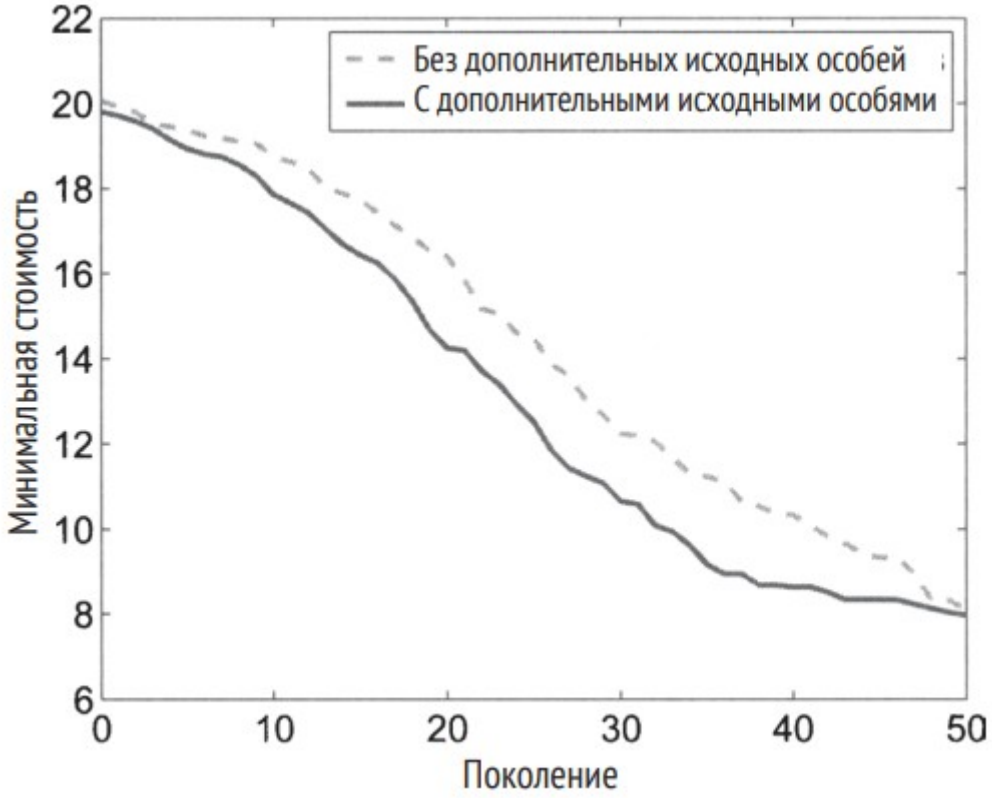
ПОДГОТОВИЛ: ЛУШКИН А.А.

ЛЕКЦИЯ 3

Родители ← {случайно сгенерированная популяция}
**While not** (критерий останова)
    Рассчитать приспособленность каждого родителя в популяции
    Дети ← ∅
    **While** | Дети | < | Родители |
        Применить приспособленности для вероятностного отбора пары родителей с целью их спаривания
        Спарить родителей для создания детей $c_1$ и $c_2$
        Дети ← Дети ∪ $\{c_1, c_2\}$
    **Loop**
    Случайно мутировать нескольких детей
    Родители ← Дети
**Next** поколение

# *Инициализация*

1. Сгенерировать случайно N особей
2. Сгенерировать 5N особей и выбрать N лучши
3. Сгенерировать N и выполнить градиентный с
4. Экспертный подбор




Сгенерировать 2N особей и выбрать N лучших

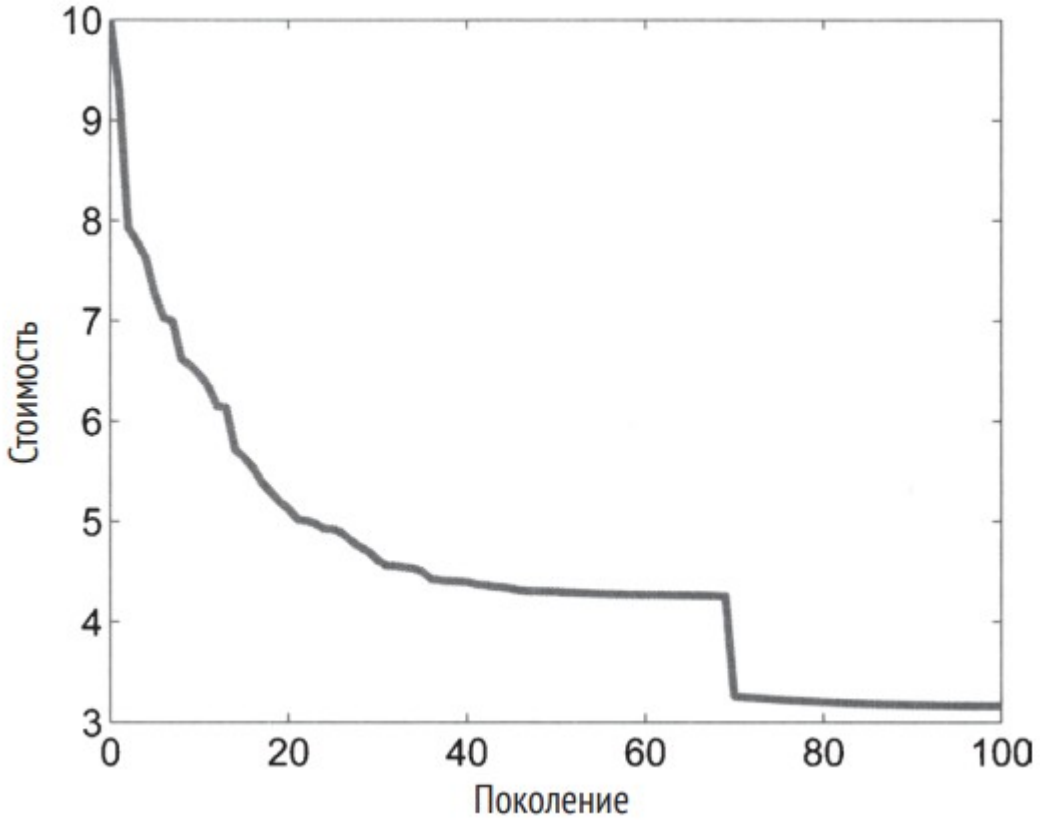
# *Критерий сходимости*

1. Остановить спустя n поколений

2. Решение обеспечивает удовлетворительный результат

3. Функция приспособленности не меняетс

4. Дисперсия разнообразия достигла поро



# *Как выбрать хромосомы в случае дискретной задачи*

## *Бинарный способ*

000 = 5-вольтовый шаговый двигатель
001 = 9-вольтовый шаговый двигатель
010 = 12-вольтовый шаговый двигатель
011 = 24-вольтовый шаговый двигатель
100 = 5-вольтовый серводвигатель
101 = 9-вольтовый серводвигатель
110 = 12-вольтовый серводвигатель
111 = 24-вольтовый серводвигатель

000 = 12-вольтовая никель-кадмиевая батарея
001 = 24-вольтовая никель-кадмиевая батарея
010 = 12-вольтовая литий-ионная батарея
011 = 24-вольтовая литий-ионная батарея
100 = 12-вольтовая солнечная панель
101 = 24-вольтовая солнечная панель
110 = 12-вольтовый термоядерный реактор
111 = 24-вольтовый термоядерный реактор

# *Как выбрать хромосомы в случае дискретной задачи*

## *Бинарный способ*

| | |
|---|---|
| 000 = 0, | 001 = 1, |
| 010 = 2, | 011 = 3, |
| 100 = 4, | 101 = 5, |
| 110 = 6, | 111 = 7. |

Мы видим, что двоичные коды для соседних чисел могут отличаться более чем на один бит. Например, код для 3 равен 011, и код для 4 равен 100.
Двоичные коды для 3 и 4 отличаются во всех трех битах. Обратное также верно, то есть двоичные коды, которые очень похожи друг на друга, иногда представляют собой числа, которые очень далеки друг от друга.
Например, 000 представляет число 0; если мы изменим один бит, получив код 100, то мы теперь будем иметь представление числа 4, которое далеко от 0 относительно диапазона чисел, которые мы представляем.

# *Как выбрать хромосомы в случае дискретной задачи*

## *Код Грея*

| | |
|---|---|
| 000 = 0, | 001 = 1, |
| 011 = 2, | 010 = 3, |
| 110 = 4, | 111 = 5, |
| 101 = 6, | 100 = 7. |

Мы видим, что коды Грея для соседних чисел всегда отличаются ровно на один бит. Например, код 3 равен 010, и код для 4 равен 110. Коды Грея для 3 и 4 отличаются только самым левым битом. Кодирование с помощью кодов Грея удаляет скалы Хэмминга, то есть большие изменения целочисленных значений между представлениями, которые отличаются только одним битом.

# *Пример*

$$\min_x f(x), \quad \text{где } f(x) = x^4 + 5x^3 + 4x^2 - 4x + 1.$$

**Двоичное**

| | |
|---|---|
| 0000 = –4.00, | 0001 = –3.67, |
| 0010 = –3.33, | 0011 = –3.00, |
| 0100 = –2.67, | 0101 = –2.33, |
| 0110 = –2.00, | 0111 = –1.67, |
| 1000 = –1.33, | 1001 = –1.00, |
| 1010 = –0.67, | 1011 = –0.33, |
| 1100 = +0.00, | 1101 = +0.33, |
| 1110 = +0.67, | 1111 = +1.00. |

**Грей**

| | |
|---|---|
| 0000 = –4.00, | 0001 = –3.67, |
| 0011 = –3.33, | 0010 = –3.00, |
| 0110 = –2.67, | 0111 = –2.33, |
| 0101 = –2.00, | 0100 = –1.67, |
| 1100 = –1.33, | 1101 = –1.00, |
| 1111 = –0.67, | 1110 = –0.33, |
| 1010 = +0.00, | 1011 = +0.33, |
| 1001 = +0.67, | 1000 = +1.00. |

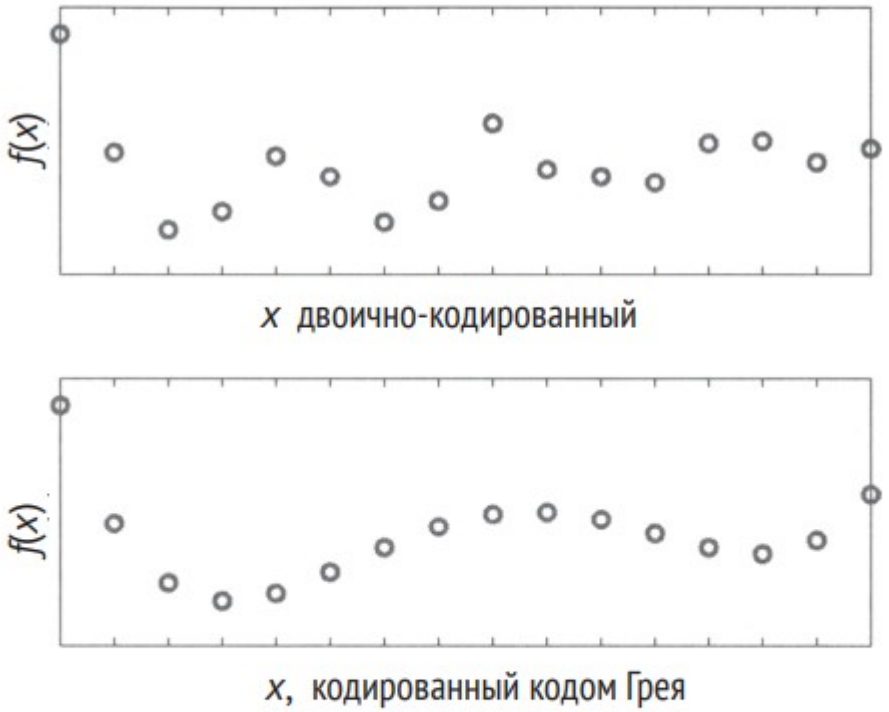



**Рис. 8.3.** Пример 8.2: в верхней части графика ось x расположена так, что однобитные изменения в x смежны, когда мы используем двоичное кодирование. Нижний график показывает то же самое, когда применяем кодирование Грея. Гладкая функция теряет свою гладкость, если мы используем двоичное кодирование

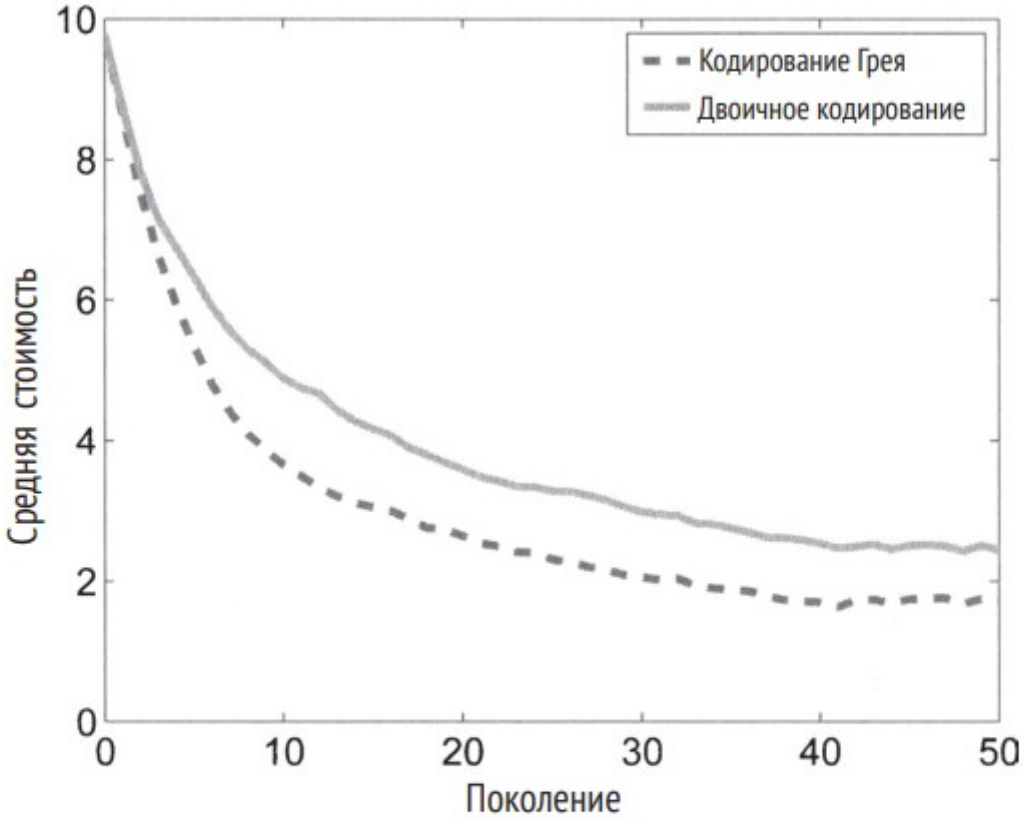



**Рис. 8.4.** Пример 8.3: результаты минимизации двумерной функции Экли, где каждая размерность кодируется шестью битами. График показывает среднюю стоимость всех генетико-алгоритмических особей в каждом поколении, усредненно по 50 симуляциям Монте-Карло. Генетический алгоритм с кодированием Грея работает заметно лучше, чем генетический алгоритм с двоичным кодированием

# *Грей vs Бинарный*

Предположим, что мы имеем задачу, соответствующую худшему случаю, для которой четные значения двоичного кода имеют стоимость 1, а нечетные значения – 2. Если особи представлены двоичными кодами, то значения стоимости для трехбитной задачи, соответствующей худшему случаю, будут

$$f(000) = 1, \quad f(001) = 2,$$
$$f(010) = 1, \quad f(011) = 2,$$
$$f(100) = 1, \quad f(101) = 2,$$
$$f(110) = 1, \quad f(111) = 2.$$

Если особи представлены кодами Грея, то значения стоимости для трехбитной задачи, соответствующей худшему случаю, записанные в том же порядке, что и двоичные представления выше, будут:

$$f(000) = 1, \quad f(001) = 2,$$
$$f(011) = 1, \quad f(010) = 2,$$
$$f(110) = 1, \quad f(111) = 2,$$
$$f(101) = 1, \quad f(100) = 2.$$

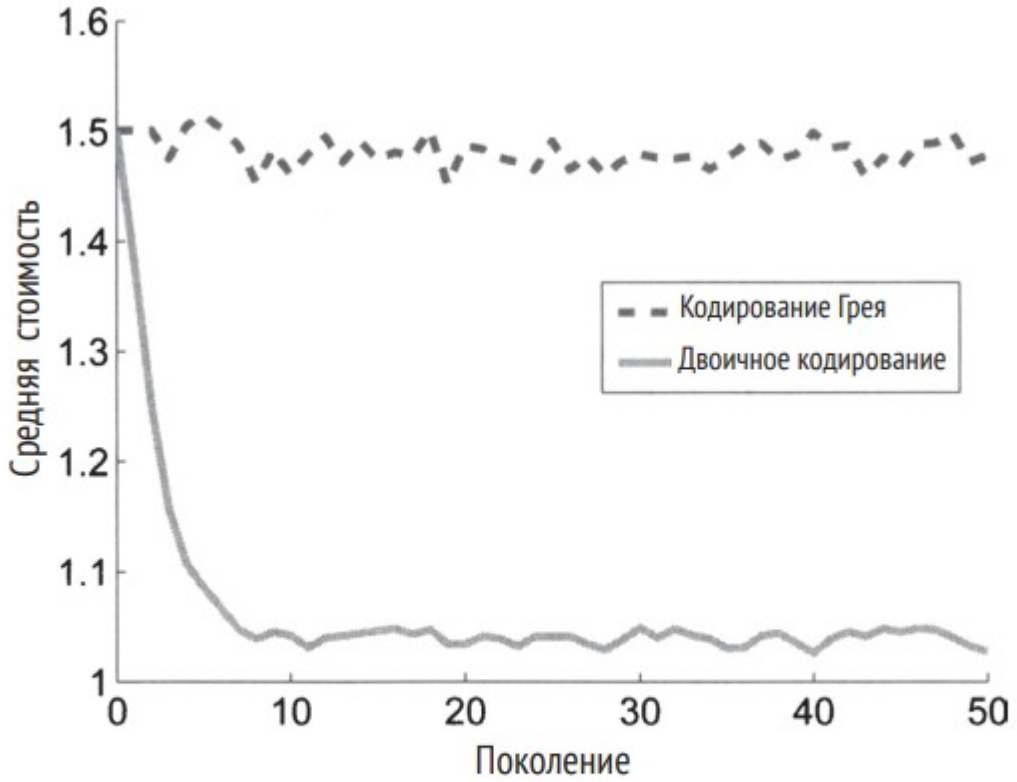



**Рис. 8.5.** Результаты примера 8.5. График показывает среднюю стоимость всех генетико-алгоритмических особей в каждом поколении 20-битной задачи, которая имеет много локальных минимумов, усредненно по 50 симуляциям Монте-Карло. При поиске нескольких локальных минимумов двоичное кодирование работает лучше, чем кодирование Грея

# *Как выбрать хромосомы в случае непрерывной задачи*

1. Определиться с точностью и границами решения:

$[a, c] = [-10, +10]$ $x \in [-10, +10]$ Точность: 3 знака после запятой

2. Определиться количество требуемых чисел:

Требуемое: (c – a) * 1000 = 20000

3. Определить размер бинарного числа:

$$16384 = 2^{14} < 20000 \leq 2^{15} = 32768$$

Вывод: для требование необходимо 15 разрядов

1. Сгенерировать какое-нибудь бинарное число длины 15

x_bin = 100100010000011

2. Перевести в десятичное

$$(< b_{14}b_{13}\dots b_0 >)_2 = \left(\sum_{i=0}^{14} b_i 2^i\right)_{10}$$

x′ = 18563

3. Перевести десятичное в вещественное

$$x = a + x' \cdot \frac{(c-a)}{2^{15}-1} = -10 + x' \cdot \frac{20}{2^{15}-1},$$

$$x' = -10 + 18563 * \frac{\big(10 - (-10)\big)}{2^{15}-1} = 1.330$$

# *Элитарность. Способ 1*

Родители ← {случайно сгенерированная популяция}
**While not** (критерий останова)
    Рассчитать приспособленность каждого родителя в популяции.
    Элитарные особи ← Лучшие *E* родителей
    Дети ← ∅
    **While** | Дети | < | Родители | – *E*
        Применить приспособленности для вероятностного отбора пары
            родителей с целью их спаривания.
        Спарить родителей для создания детей $c_1$ и $c_2$.
        Дети ← Дети ∪ $\{c_1, c_2\}$
    **Loop**
    Случайно мутировать нескольких детей.
    Родители ← Дети ∪ Элитарные особи
**Next** поколение

**Рис. 8.6.** Вариант элитарности 1: простой генетический алгоритм, модифицированный для элитарности. *N* – это размер популяции, *E* – число элитарных особей, которые оставляются из поколения в поколение, и каждое поколение производит (*N* – *E*) детей

# *Элитарность. Способ 2*

Родители ← {случайно сгенерированная популяция}
**While not** (критерий останова)
    Рассчитать приспособленность каждого родителя в популяции.
    Элитарные особи ← Лучшие *E* родителей
    Дети ← ∅
    **While** | Дети | < | Родители |
        Применить приспособленности для вероятностного отбора пары
            родителей с целью их спаривания.
        Спарить родителей для создания детей $c_1$ и $c_2$.
        Дети ← Дети U $\{c_1, c_2\}$
    **Loop**
    Случайно мутировать нескольких детей.
    Родители ← Дети U Элитарные особи
    Родители ← Лучшие *N* родителей
**Next** поколение

**Рис. 8.7.** Варинат элитарности 2: простой генетический алгоритм, модифицированный для элитарности. *N* – это размер популяции, *E* – число элитарных особей, которые оставляются из поколения в поколение, и каждое поколение производит *N* детей

Элитарный генетический алгоритм оставляет двух лучших особей каждое поколение и использует вариант элитарности 1

- Используем популяцию размером 20
- скорость мутации 2 % на бит на поколение

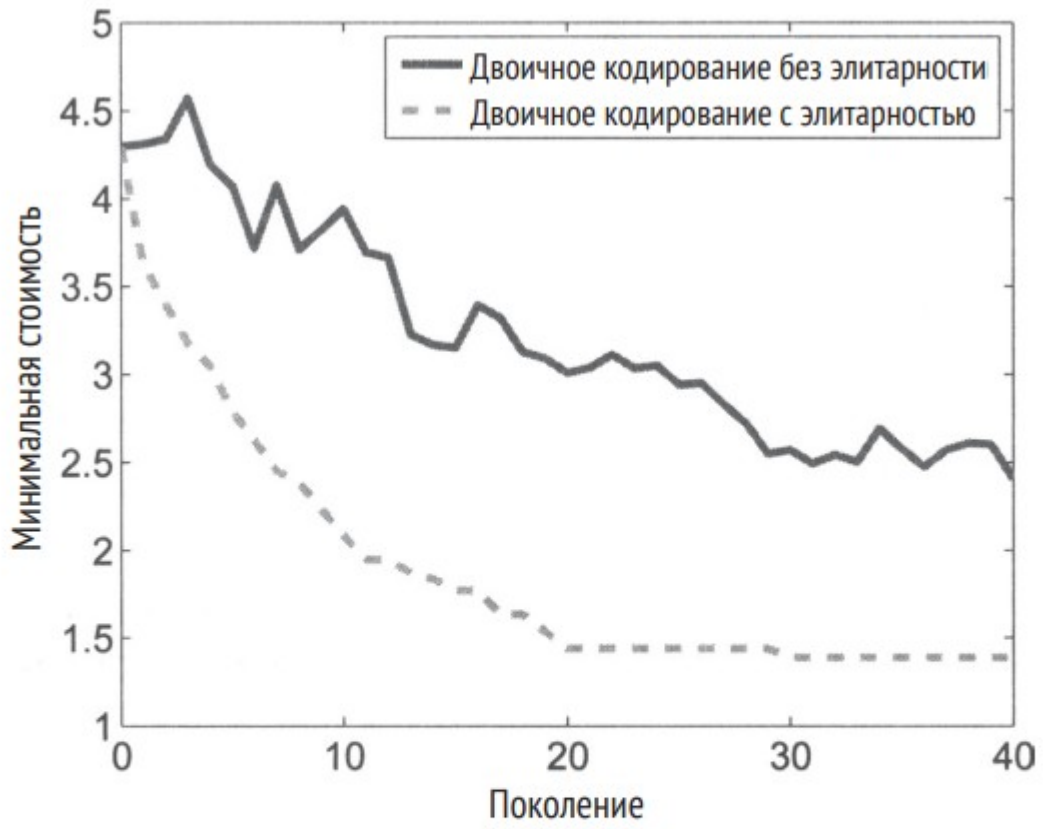



**Рис. 8.8.** Результаты примера 8.6 для минимизации двумерной функции Экли, где каждая размерность кодируется шестью битами. График показывает минимальную стоимость всех генетико-алгоритмических особей в каждом поколении, усредненно по 20 симуляциям Монте-Карло. Элитарный генетический алгоритм работает значительно лучше, чем неэлитарный генетический алгоритм

# *Отбор родителей. Проблема классического отбора*

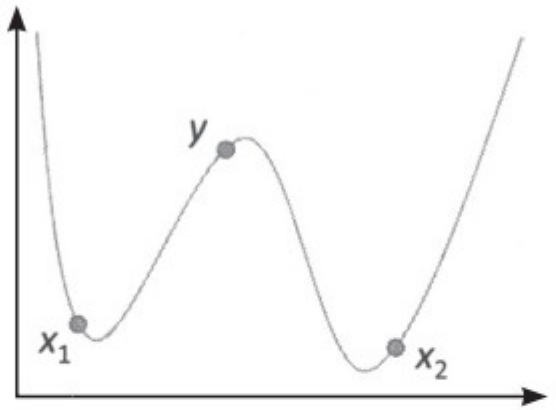



**Рис. 8.10.** Данный рисунок демонстрирует задачу мультимодальной минимизации. Этот пример функции иллюстрирует проблему с рекомбинацией двух сильно отличающихся друг от друга особей. Родители $x_1$ и $x_2$ имеют низкую стоимость, но их ребенок $y$ имеет высокую стоимость. Кроме того, если ребенок $y$ заменяет одного из родителей (например, $x_2$), то эволюционному алгоритму будет трудно найти глобальный оптимум, близкий к $x_2$

# *Отбор родителей. Проблема классического отбора*

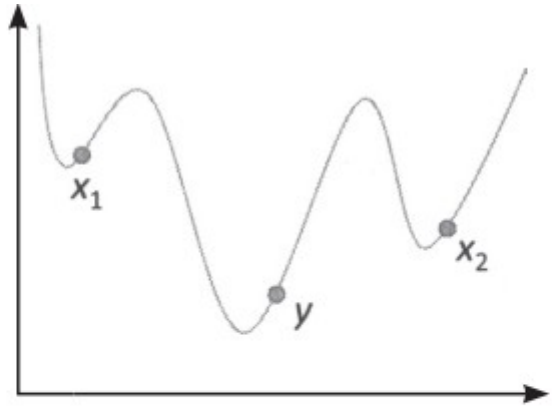



**Рис. 8.11.** Данный рисунок демонстрирует задачу мультимодальной минимизации. Этот пример функции иллюстрирует проблему, возникающую, когда сильно отличающимся друг от друга особям не разрешается рекомбинировать. Если $x_1$ и $x_2$ не разрешается рекомбинировать из-за их несходства, то может оказаться трудно найти глобальный минимум, близкий к $y$

# *Отбор родителей. Ниширование. Обмен информацией о приспособленности*

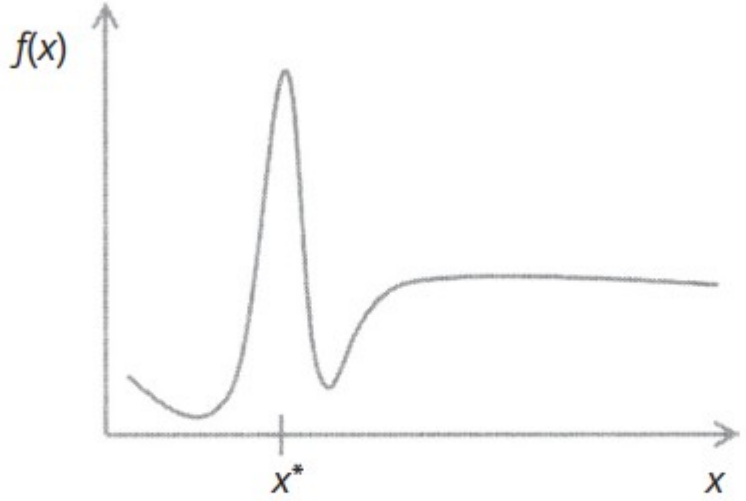



**Рис. 8.12.** Эта функция имеет глобальный максимум в $x^*$, но особи возле $x^*$ вряд ли будут выбраны для рекомбинации из-за их низких значений приспособленности по отношению к другим особям в поисковом пространстве

Рассмотрим рис. 8.12. Мы видим, что x* является глобальным максимумом, но особи возле x* вряд ли выживут до следующего поколения из-за их низких значений приспособленности, по сравнению с другими особями в поисковом пространстве. Для того чтобы поощрять многообразие в популяции, мы можем искусственно увеличивать значения приспособленности относительно уникальных особей и уменьшать значения приспособленности относительно распространенных особей.

Предположим, что у нас есть эволюционно-алгоритмическая популяция {xi} из N особей и что fi является приспособленностью xi . Метод обмена информацией о приспособленности вычисляет модифицированные значения приспособленности следующим образом

$$f_i' = f_i / m_i,$$

где число mi , так называемое число ниш в xi , связано с числом особей, подобных xi . Число ниш вычисляется формулой

$$m_i = \sum_{j=1}^{N} s(d_{ij}),$$

где s(·) – это функция обмена, а dij – расстояние между особями xi и xj . Для получения dij мы часто используем евклидово расстояние. Общепринято использовать функцию обмена

$$s(d) = \begin{cases} 1 - (d/\sigma)^{\alpha}, & \text{если } d < \sigma \\ 0 & \text{в противном случае} \end{cases},$$

где $\sigma$ – это определяемый пользователем параметр, называемый порогом несходства, порогом разнородности, расстоянием отсечения или нишевым радиусом, и $\alpha$ – определяемый пользователем параметр. Мы обычно используем  = 1, в результате получая треугольную функцию обмена. Исследователи предложили разные методы установки порога несходства [Deb и Goldberg, 1989]. Например,

$$\sigma = rq^{-1/n},$$

$$r = \frac{1}{2}\sum_{k=1}^{n}\left(\max_i x_i(k) - \min_i x_i(k)\right)^2,$$

где n – это размерность задачи, xi (k) – k-й элемент xi и q – ожидаемое число локальных оптимумов в функции приспособленности. В обмене информацией о приспособленности при отборе родителей для рекомбинации мы используем модифицированные значения приспособленности из уравнения (8.7). Обратите внимание, что если у нас минимизационная задача, то перед применением уравнения (8.7) нам нужно конвертировать значения стоимости в значения приспособленности, а затем конвертировать модифицированные значения приспособленности в модифицированные значения стоимости (см. задачу 8.7).

# *Отбор родителей. Очистка.*

Очистка аналогична обмену информацией о приспособленности, но, вместо того чтобы делиться значениями приспособленности между особями, которые разделяют одну и ту же нишу, мы уменьшаем приспособленность некоторых из этих особей [Pétrowski, 1996], [Sareni и Krahenbiihl, 1998]. Данную идею можно реализовать несколькими способами, в том числе следующими. Во-первых, мы определяем множество ниш Di каждой особи в популяции:

$$D_i = \{x_j : d_{ij} < \sigma\},$$

где dij – это такое же расстояние, как и в уравнении (8.8), а  – определяемый пользователем параметр. Затем мы ранжируем особей в каждой нише в соответствии с их приспособл

$$r_{ki} = \text{rank of } x_k \text{ in } D_i,$$

где лучшая особь в каждой нише имеет ранг 1, вторая лучшая особь имеет ранг 2 и т. д

Наконец, мы определяем параметр R как число особей, которые по нашему желанию должны выжить в каждой нише, и получаем модифицированные значения приспособленности следующим образом, где N – это размер популяци

**For** $i = 1$ **to** $N$
  **For** $k = 1$ **to** $|D_i|$
    **If** $r_{ki} \leq R$ **then**
      $f'_k \leftarrow f_k$
    **else**
      $f'_k \leftarrow -\infty$
    **End if**
  **Next** ниша
**Next** особь

Приведенный выше алгоритм гарантирует, что наименее приспособленные особи в каждой нише недоступны для отбора или рекомбинации. Тем не менее это не гарантирует, что наиболее приспособленные особи будут доступны для отбора, потому что особи могут принадлежать более чем к одной нише.

# *Отбор родителей. Скучивание.*

Стандартное скучивание используется в сочетании со стационарной рекомбинацией (см. раздел 8.5). Стандартное скучивание производит $M$ детей в каждом поколении, а затем сравнивает этих детей со случайно отобранными родителями $C_f$, где $M$ и $C_f$ – это задаваемые пользователем параметры. $C_f$ называется фактором скученности. Каждый ребенок заменяет наиболее похожую особь из группы случайно отобранных родительских особей. Обычно используются значения параметров $M = N/10$ и $C_f = 3$, где $N$ – это размер популяции [Mahfoud, 1992]. На рис. 8.13 показана реализация стандартого скучивания.

# *Отбор родителей. Скучивание.*

Родители $\{p_k\} \leftarrow$ {случайно сгенерированная популяция из $N$ особей}
Рассчитать приспособленность каждого родителя $p_k$, $k \in [1, N]$.
**While not** (критерий останова)
    Применить приспособленности для вероятностного отбора $M$ родителей
        с целью рекомбинации.
    Рекомбинировать родителей для создания $M$ детей $c_i$, $i \in [1, M]$.
    Случайно мутировать каждого ребенка $c_i$, $i \in [1, M]$.
    Рассчитать приспособленность каждого ребенка $c_i$, $i \in [1, M]$.
    **For** $i = 1$ **to** $M$
      Случайно отобрать $C_f$ особей $I$ из родительской популяции $\{p_k\}$.
      $p_{\min} = \arg\min_p \| p - c_i \| : \mathrm{p} \in I$
      $p_{\min} \leftarrow c_i$
    **Next** ребенок
**Next** поколение

**Рис. 8.13.** Стационарный эволюционный алгоритм со стандартным скучиванием. $M$ и $C_f$ – это отобранные пользователем параметры, а $\|p - c_i\|$ – определяемая пользователем функция расстояния

# *Отбор родителей. Рулетка.*

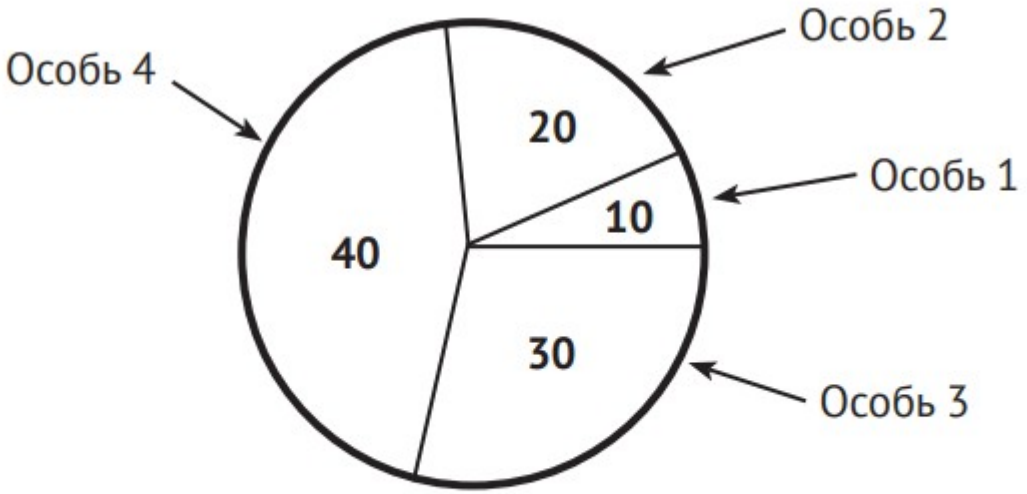



**Рис. 8.16.** Иллюстрация рулеточного отбора для четырехчленной популяции. Каждой особи присваивается сектор, площадь которого пропорциональна ее приспособленности. Отбор каждой особи в качестве родителя пропорционален площади ее сектора в колесе рулетки

# *Отбор родителей. Рулетка. Модификация*

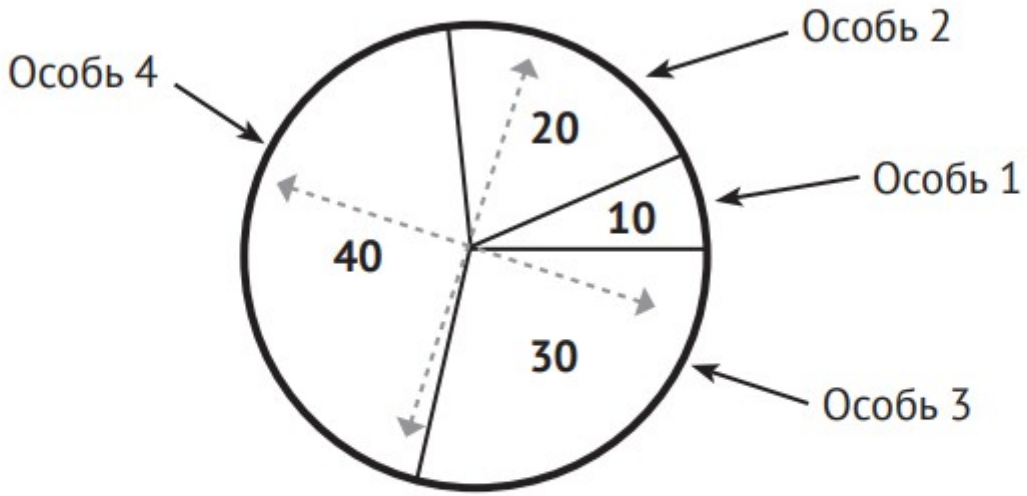



**Рис. 8.17.** Стохастическая универсальная выборка для четырехчленной популяции. Каждой особи назначается сектор, пропорциональный ее приспособленности. Вертушка с четырьмя равноотстоящими указателями вращается один раз, для того чтобы получить четырех родителей

$x_i$ = $i$-й особь в популяции, $i \in [1, N]$

$f_i \leftarrow$ приспособленность $x_i$, для $i \in [1, N]$

$f_{sum} \leftarrow \sum_{i=1}^{N} f_i$

Сгенерировать равномерно распределеное случайное число $r \in [0, f_{sum}/N]$.

$f_{accum} \leftarrow 0$

Родители $\leftarrow \emptyset$

$k \leftarrow 0$

**While** | Родители | $< N$

    $k \leftarrow k + 1$

    $f_{accum} \leftarrow f_{accum} + f_k$

    **While** $f_{accum} > r$

        Родители $\leftarrow$ Родители $\cup$ $x_k$

        $r \leftarrow r + f_{sum}/N$

    **End while**

**Next** родитель

**Рис. 8.18.** Псевдокод для отбора $N$ родителей из $N$ особей с помощью стохастической универсальной выборки. Этот код исходит из того, что $f_i \geq 0$ для всех $i \in [1, N]$

# *Отбор родителей. Избыточный отбор.*

Избыточный отбор (over-selection) – это метод, первоначально предложенный Дж. Козой в контексте генетического программирования [Koza, 1992, глава 6]. Избыточный выбор модифицирует рулеточный отбор путем непропорционального взвешивания значений приспособленности высоко приспособленных особей, для того чтобы увеличить их шансы на отбор. В версии Дж. Козы избыточного отбора лучшие 32 % популяции имеют 80%-ный шанс быть отобранными, и худшие 68 % популяции имеют 20%-ный шанс быть отобранными. Точные проценты не слишком важны; ключевой особенностью избыточного отбора является то, что приспособленные особи имеют значительно более высокую вероятность быть отобранными. Это один из типов шкалирования по значению приспособленности.

# *Отбор родителей. Сигма шкалирование.*

Сигма-шкалирование (sigma scaling) нормализует значения приспособленности относительно стандартного отклонения приспособленности всей популяции в целом. Шкалированные значения приспособленности следующие

$$f'(x_i) = \begin{cases} \max[1 + (f(x_i) - \bar{f})/(2\sigma), \epsilon] & \text{если } \sigma \neq 0 \\ 1 & \text{если } \sigma = 0 \end{cases},$$

где f(xi ) – это приспособленность i-й особи в популяции, f – среднее значений приспособленности, σ – стандартное отклонение значений приспособленности и σ – неотрицательное определяемое пользователем минимально допустимое шкалированное значение приспособленности

$$\sigma = \left( \frac{1}{N} \sum_{i=1}^{N} \left( f(x_i) - \bar{f} \right)^2 \right)^{1/2}.$$

# *Пример*

Предположим, что у нас есть четырехчленная популяция со значениями приспособленности

$$f(x_1) = 10, \qquad f(x_2) = 5,$$
$$f(x_3) = 40, \qquad f(x_4) = 15.$$

Рулеточный отбор дает особям следующие селекционные вероятности:

$$\Pr(x_1) = 14\ \%, \qquad \Pr(x_2) = 7\ \%,$$
$$\Pr(x_3) = 57\ \%, \qquad \Pr(x_4) = 22\ \%.$$

Среднее и стандартное отклонение значений приспособленности уравнения (8.19) равны f = 17.5 и σ = 15.5, где для вычисления σ мы используем уравнение, которое дает шкалированные значения приспособленности

$$f'(x_1) = 0.76, \qquad f'(x_2) = 0.60,$$
$$f'(x_3) = 1.72, \qquad f'(x_4) = 0.92.$$

Если мы применим эти шкалированные значения приспособленности в алгоритме рулеточного отбора, то получим селекционные вероятности:

$$\Pr(x_1) = 19\ \%, \qquad \Pr(x_2) = 15\ \%,$$
$$\Pr(x_3) = 43\ \%, \qquad \Pr(x_4) = 22\ \%.$$

# *Отбор родителей. Ранговый отбор.*

Ранговый отбор ранжирует особей в соответствии со значениями приспособленности, давая лучшей особи ранг N (где N – это размер популяции) и худшей особи – ранг 1

$$R(x_1) = 2, \qquad R(x_2) = 1,$$
$$R(x_3) = 4, \qquad R(x_4) = 3.$$

Предположим, что мы используем ранговый отбор в сочетании с рулеточным отбором для рангов выше. В результате получаем следующие селекционные вероятности:

$$\Pr(x_1) = 20\ \%, \qquad \Pr(x_2) = 10\ \%,$$
$$\Pr(x_3) = 40\ \%, \qquad \Pr(x_4) = 30\ \%.$$

Мы видим, что когда значения приспособленности друг от друга сильно отличаются, ранговый отбор выравнивает селекционные вероятности. Это не дает высоко приспособленным особям доминировать в популяции на ранних стадиях эволюции, то есть предотвращает преждевременное схождение.

# *Отбор родителей. Турнирный отбор.*

Турнирный отбор (tournament selection) снижает вычислительные затраты, связанные с отбором. На рис. 3.5 показано, что рулеточный отбор N родителей из популяции N особей требует вложенных циклов, что может быть вычислительно затратно для больших популяций. В случае турнирного отбора мы случайно отбираем $\tau$ особей из популяции, где  > 2 – это определяемый пользователем размер турнира. Затем мы сравниваем значения приспособленности отобранных особей и отбираем наиболее приспособленных для рекомбинации.

# *Рекомбинация*

Предположим, что у нас есть популяция особей {x1 , x2 , ..., xN}. Каждая особь имеет n признаков. Обозначим k-й признак i-й особи как xi (k) для k ∈ [1, n]. Таким образом, мы можем представить xi как вектор

$$x_i = [x_i(1) \quad x_i(2) \quad \dots \quad x_i(n)].$$

Мы обозначаем детскую особь, то есть ребенка, который является результатом рекомбинации, как y. Обозначим k-й признак ребенка как y(k), поэтому

$$y = [y(1) \quad y(2) \quad \dots \quad y(n)].$$

# *Одноточечное скрещивание (в бинарных или непрерывных эволюционных алгоритмах*

Предположим, что у нас есть два родителя, xa и xb , где a ∈ [1, N] и b ∈ [1, N]. Одноточечное скрещивание, так называемое простое скрещивание или дискретное скрещивание, является типом скрещивания, который впервые был использован в бинарном генетическом алгоритме

$$y(k) \leftarrow [x_a(1) \ \ldots \ x_a(m) \quad x_b(m+1) \ \ldots \ x_b(n)],$$

где m – это случайно отобранная точка скрещивания, то есть m ~ U[0, n]. Если m = 0, то y является клоном xb . Если m = n, то y является клоном xa . Одноточечное скрещивание часто реализуется для получения двух детей от пары родителей. Это делается путем отбора каждого признака второго ребенка y2 у противоположного родителя, чем у того, у которого y1 получил свой признак:

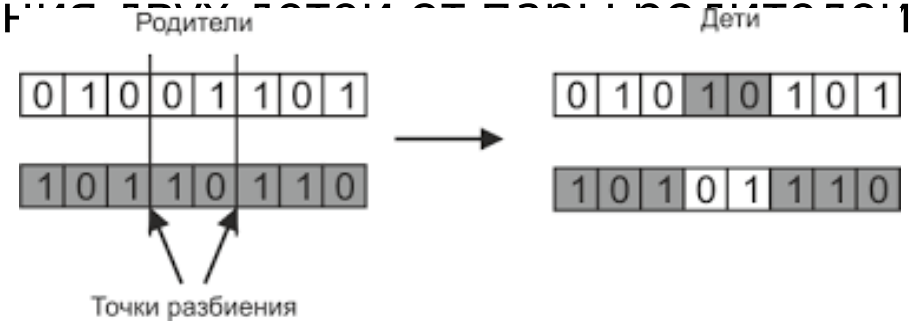
$$y_1(k) \leftarrow [x_a(1) \ \ldots \ x_a(m) \quad x_b(m+1) \ \ldots \ x_b(n)]$$
$$y_2(k) \leftarrow [x_b(1) \ \ldots \ x_b(m) \quad x_a(m+1) \ \ldots \ x_a(n)]$$

# *Многоточечное скрещивание (в бинарных или непрерывных эволюционных алгоритмах)*

Двухточечное скрещивание сводится к

$$y(k) \leftarrow [\, x_a(1) \ \ldots \ x_a(m_1) \ x_b(m_1+1) \ \ldots \ x_b(m_2) \ x_a(m_2+1) \ \ldots \ x_a(n) \,]$$

где двумя точками скрещивания являются m1 ~ U[0, n] и m2 ~ U[m1 + 1, n]. Если m1 = 0 или m2 = n, то двухточечное скрещивание сводится к одноточечному скрещиванию. Если m1 = n, то y является клоном xa . Уравнение (8.46) может быть расширено до трехточечного скрещивания или M-точечного скрещивания для любого M > 2. Как и в случае с одноточечным скрещиванием, многоточечное скрещивание часто реализуется для получения двух детей от пары родителей.



# *Сегментированное скрещивание (в бинарных или непрерывных эволюционных алгоритмах*

Можно представить как обобщение многоточечного скрещивания. Ребенок 1 получает свой первый признак от родителя 1. Затем, с вероятностью p, мы переключаемся на родителя 2 и получаем второй для ребенка 1 признак; и с вероятностью (1 – p) мы получаем второй для ребенка 1 признак от родителя 1. Каждый раз, когда мы получаем признак для ребенка 1, то переключаемся на другого родителя, чтобы получить следующий признак с вероятностью p. Ребенок 1 и ребенок 2 получают свои признаки от разных родителей, поэтому если ребенок 1 получает признак k от родителя 1, то ребенок 2 получает признак k от родителя 2, для $k \in [1, n]$. Аналогичным образом, если ребенок 1 получает признак k от родителя 2, то ребенок 2 получает признак k от родителя 1. Сегментированное скрещивание эквивалентно многоточечному скрещиванию, где число точек скрещивания является случайным числом.

$S \leftarrow$ true

**For** $k = 1$ **to** $n$

  **If** $S$ **then**

    $c_1(k) \leftarrow p_1(k)$

    $c_2(k) \leftarrow p_2(k)$

  **else**

    $c_1(k) \leftarrow p_2(k)$

    $c_2(k) \leftarrow p_1(k)$

  **End if**

  $r \leftarrow U[0, 1]$

  **If** $r < \rho$ **then** $S \leftarrow$ not $S$

**Next** признак решения

## *Равномерное скрещивание (в бинарных или непрерывных эволюционных алгоритмах*

Предположим, что у нас есть два родителя, xa и xb . Равномерное скрещивание приводит к ребенку y, где k-й признак ребенка y задается следующим образом:

$$y(k) \leftarrow x_{i(k)}(k)$$

для каждого $k \in [1, n]$, где мы случайно отбираем i(k) из множества {a, b}. То есть мы случайно отбираем каждый детский признак у одного из двух родителей, каждый с вероятностью 50 %

# *Многородительское скрещивание (в бинарных или непрерывных эволюционных алгоритмах*

Многородительское скрещивание, которое мы здесь обсуждаем, является обобщением однородного скрещивания и употребляется под несколькими другими названиями, в том числе генофондовая рекомбинация, сканирующее скрещивание, а также многополое скрещивание (в отличие от двухродительского скрещивания, которое называется двуполым скрещиванием). В многородительском скрещивании мы случайно отбираем каждый детский признак от одного из родителей, где число родителей больше двух. Этот метод был предложен еще в 1966 году. Многородительское скрещивание

$$y_k \leftarrow x_{i(k)}(k)$$

для каждого $k \in [1, n]$, где мы случайно отбираем $i(k)$ из подмножества $[1, N]$ (напомним, что в популяции есть N потенциальных родителей). При реализации многородительского скрещивания нам нужно принять несколько решений. Например, сколько особей должно быть в фонде потенциальных родителей? Как отбирать особей для родительского фонда? После того как фонд был определен, как отбирать родителей из родительского фонда? Наконец, отметим, что к многородительскому скрещиванию также предлагались и другие подходы.

## *Глобальное однородное скрещивание (в бинарных или непрерывных эволюционных алгоритмах)*

Один из способов реализации многородительского скрещивания заключается в случайном отборе каждого детского признака у одного из родителей, где родительский фонд эквивалентен всей популяции. В результате получается глобальная равномерная рекомбинация:

$$y_k \leftarrow x_{i(k)}(k)$$

для каждой k ∈ [1, n], где i(k) отбирается случайно из равномерного распределения [1, N] для каждой k. В качестве альтернативы i(k) может быть отобрана на основе приспособленности. То есть вероятность того, что i(k) = m, может быть пропорциональна приспособленности xm для всех k ∈ [1, n] и m ∈[1, N]

# *Перетасованное скрещивание (в бинарных или непрерывных эволюционных алгоритмах)*

Перетасованное скрещивание случайно переставляет признаки решения в родителях. Во всех родителях, которые вносят вклад в данного ребенка, мы используем одну и ту же перестановку признаков решения. Затем мы выполняем один из упомянутых выше методов скрещивания (обычно одноточечное скрещивание) для получения детей. Потом отменяем перестановку признаков решения в ребенке. На рис. 8.23 показан алгоритм перетасованного скрещивания в сочетании с одноточечным скрещиванием.

$\{r_1, ..., r_n\} \leftarrow$ случайная перестановка элементов $\{1, ..., n\}$
Точка скрещивания $m \leftarrow U[1, n-1]$
**For** $k = 1$ **to** $m$
    $t_1(k) \leftarrow p_1(r_k)$
    $t_2(k) \leftarrow p_2(r_k)$
**Next** $k$
**For** $k = m + 1$ **to** $n$
    $t_1(k) \leftarrow p_2(r_k)$
    $t_2(k) \leftarrow p_1(r_k)$
**Next** $k$
**For** $k = 1$ **to** $n$
    $c_1(r_k) \leftarrow t_1(k)$
    $c_2(r_k) \leftarrow t_2(k)$
**Next** $k$

# *Плоское скрещивание и арифметическое скрещивание (в непрерывных эволюционных алгоритмах)*

Плоское скрещивание и арифметическое скрещивание (в непрерывных эволюционных алгоритмах)

$$y(k) \leftarrow U[x_a(k), x_b(k)]$$
$$= \alpha x_a(k) + (1 - \alpha)x_b(k),$$

де a ~ U[0, 1]. То есть y(k) – это случайное число, взятое из равномерного распределения между k-ми признаками двух его родителей. Это равносильно утверждению, что ребенок представляет собой линейную комбинацию признаков двух своих родителей. Иногда плоское и арифметическое скрещивания отличаются тем, что плоское скрещивание дает одного ребенка, тогда как арифметическое скрещивание дает двух детей:

плоское скрещивание: $y(k) = \alpha x_a(k) + (1 - \alpha)x_b(k);$

арифметическое скрещивание:
$$\begin{cases} y_1(k) = \alpha x_a(k) + (1 - \alpha)x_b(k) \\ y_2(k) = (1 - \alpha)x_a(k) + \alpha x_b(k). \end{cases}$$

## *Линейное скрещивание (в непрерывных эволюционных алгоритмах)*

Линейное скрещивание создает трех детей от родителей xa и xb

$$
\begin{aligned}
y_1(k) &\leftarrow (1/2)x_a(k) + (1/2)x_b(k) \\
y_2(k) &\leftarrow (3/2)x_a(k) + (1/2)x_b(k) \\
y_3(k) &\leftarrow (-1/2)x_a(k) + (3/2)x_b(k)
\end{aligned}
$$

В зависимости от конкретной реализации эволюционного алгоритма мы оставляем наиболее приспособленного либо двух наиболее приспособленных из трех детей для следующего поколения

# *Равномерная мутация с центром в xi (k)*

$$r \leftarrow U[0, 1]$$

$$x_i(k) \leftarrow \begin{cases} x_i(k) & \text{если } r \geq \rho \\ U[x_i(k) - \alpha_i(k),\ x_i(k) + \alpha_i(k)] & \text{если } r < \rho \end{cases}$$

для i ∈ [1, N] и k ∈ [1, n], где i (k) – это определяемый пользователем параметр, который задает величину мутации. Мы часто выбираем i (k) как можно больше, обеспечивая при этом, чтобы мутация оставалась в поисковой области:

$$\alpha_i(k) = \min(x_i(k) - x_{\min}(k),\ x_{\max}(k) - x_i(k)).$$

# Равномерная мутация с центром в середине поисковой области

Равномерная мутация с центром в середине поисковой области может быть записана как

$$r \leftarrow U[0, 1]$$

$$x_i(k) \leftarrow \begin{cases} x_i(k) & \text{если } r \geq \rho \\ U[x_{\min}(k),\ x_{\max}(k)] & \text{если } r < \rho \end{cases} \qquad (8.60)$$

для $i \in [1, N]$ и $k \in [1, n]$.