МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Томский государственный университет систем управления и радиоэлектроники

Факультет систем управления

Кафедра автоматизированных систем управления

ЭВРИСТИЧЕСКИЕ МЕТОДЫ ОПТИМИЗАЦИИ

Учебное пособие

ТОМСК – 2024

УДК 519.852(075.8)
ББК 22.183я73
М701

Мицель, А. А. Эвристические методы оптимизации: учебное пособие/ А. А. Мицель. — Томск: ТУСУР, 2024. — 82 с.

В пособии представлены эвристические методы поиска минимума функции. Основное внимание уделено генетическим алгоритмам. В пособие включены 4 темы: эволюционные методы оптимизации; генетические алгоритмы оптимизации; многомерная безусловная оптимизация при помощи генетических алгоритмов; примеры задач, решаемых с помощью генетических алгоритмов
Пособие подготовлено для студентов, обучающихся по направлению 09.04.01 – «информатика и вычислительная техника (магистратура)»

Методические указания утверждены на заседании кафедры автоматизированных систем управлениям № 11 от “11” ноября 2023

УДК 519.852(075.8)
ББК 22.183я73
М701

.

СОДЕРЖАНИЕ

ТЕМА 1
ЭВРИСТИЧЕСКИЕ МЕТОДЫ ОПТИМИЗАЦИИ

За последние десятилетия были изобретены новые методы оптимизации, концептуально отличающиеся от традиционных. Большинство из них основаны на определенных характеристиках, присущих биологическим, молекулярным, физическим, нейробиологическим системам.

Эвристические методы основаны на подсознательном мышлении и характеризуются неосознанным (интуитивным) способом действий для достижения осознанных целей. Эвристические методы еще называют методами инженерного(изобретательного) творчества.

Эвристический алгоритм — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев. В действительности может быть даже известно (то есть доказано) то, что эвристический алгоритм формально неверен. Его всё равно можно применять, если при этом он даёт неверный результат только в отдельных, достаточно редких и хорошо выделяемых случаях или же даёт неточный, но всё же приемлемый результат.

Проще говоря, эвристика — это не полностью математически обоснованный (или даже «не совсем корректный»), но при этом практически полезный алгоритм.

Важно понимать, что эвристика, в отличие от корректного алгоритма решения задачи, обладает следующими особенностями.

- Она не гарантирует нахождение лучшего решения.
- Она не гарантирует нахождение решения, даже если оно заведомо существует (возможен «пропуск цели»).
- Она может дать неверное решение в некоторых случаях.

1.1 Классификация эвристических методов поиска экстремумов [1]

Дана целевая функция $f(x) = f(x_1,...,x_n)$, определенная на множестве допустимых решений $D \in R^n$. Требуется найти глобальный условный минимум функции $f(x)$ на множестве D

$$f(x^*) = \min_{x \in D}(f(x)), \tag{1.1}$$

где $D = \{x \mid x_i \in [a_i, b_i],\ i = 1,...,n\}$.

На рис.1.1 приведены некоторые классы эвристических методов.

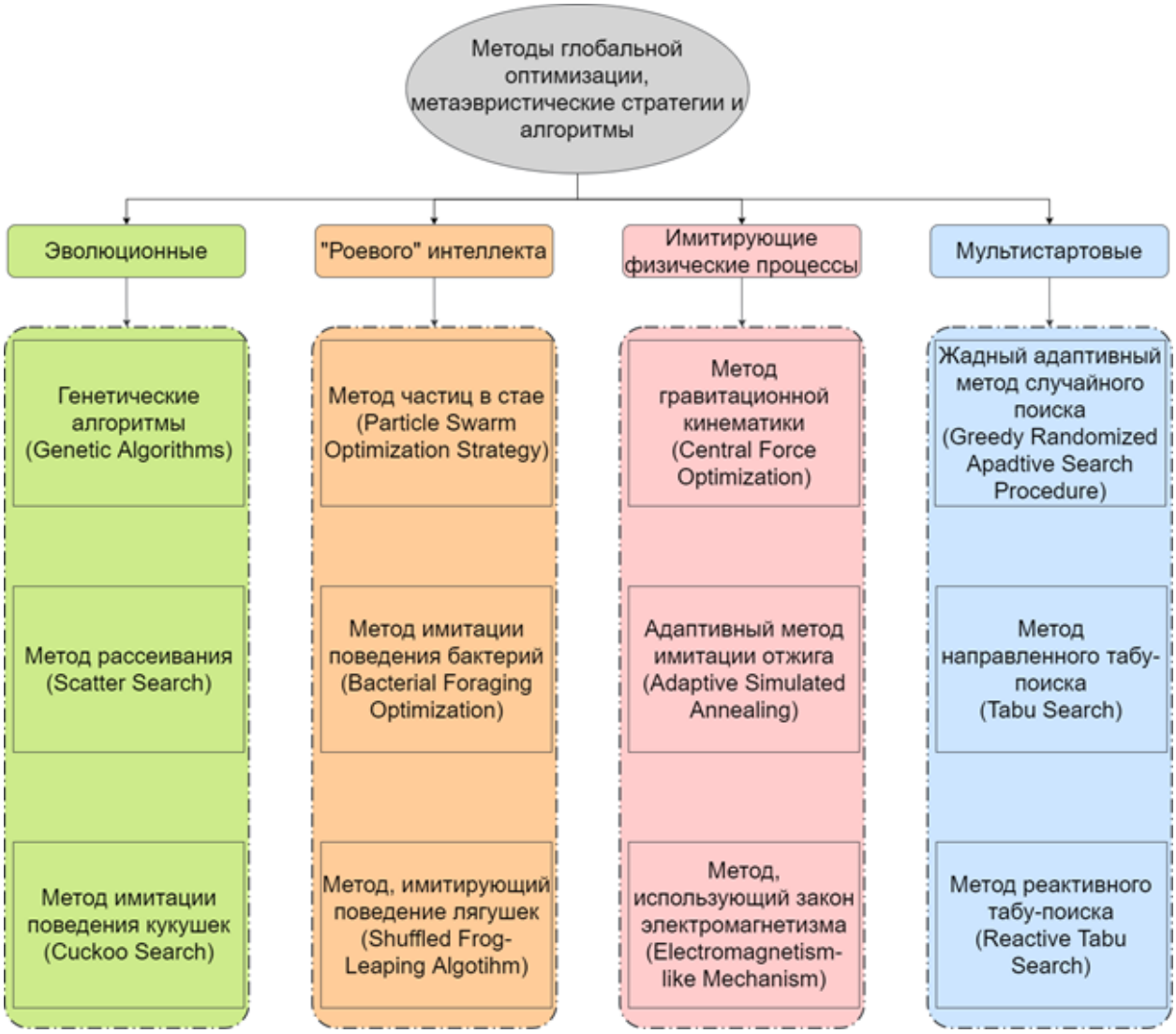


Рис. 1.1 – Классификация методов

1) ЭВОЛЮЦИОННЫЕ МЕТОДЫ

- генетические алгоритмы;
- методы, имитирующие иммунные системы организмов – методы искусственных иммунных систем;
- метод рассеивания;
- эволюционная стратегия преобразования ковариационной матрицы;
- метод динамических сеток;

- метод дифференциальной эволюции;
- метод, имитирующий распространение сорняков;
- метод, имитирующий поведение кукушек;
- метод растущих деревьев;
- метод эволюции разума;
- метод бактериальной оптимизации;
- культурный метод;
- меметический метод;
- самообразующийся миграционный метод;
- метод прокладки путей;
- метод интеллектуальных капель воды.

2) **МЕТОДЫ «РОЕВОГО» ИНТЕЛЛЕКТА**

- метод частиц в стае;
- метод муравьиных колоний;
- метод имитации поведения бактерий;
- методы пчелиных колоний;
- метод, имитирующий поведение стаи рыб в поисках корма;
- метод, имитирующий поведение летучих мышей;
- метод, имитирующий поведение светлячков;
- алгоритм, имитирующий поведение лягушек;
- метод, имитирующий динамику формирования рек;
- метод, имитирующий поведение котов;
- метод, имитирующий поведение обезьян.

3) **МЕТОДЫ, ИМИТИРУЮЩИЕ ФИЗИЧЕСКИЕ ПРОЦЕССЫ**

- метод гравитационной кинематики;
- метод имитации отжига и его модификации –параллельный, быстрый, адаптивный метод имитации отжига);
- метод поиска гармонии;
- метод, использующий закон электромагнетизма;
- метод экстремальной оптимизации;

- метод стохастического диффузионного поиска.

4) МУЛЬТИСТАРТОВЫЕ МЕТОДЫ

- жадный адаптивный метод случайного поиска;
- метод направленного табу-поиска;
- метод реактивного табу-поиска;

1.1.1. Классификация эволюционных методов

Эволюционные методы поиска (Evolutionary Methods) – эвристические методы, в основе которых лежит эволюционная теория Чарльза Дарвина. Способы реализации этих методов значительно отличаются друг от друга. Идея состоит в наличии популяции индивидов, зависимой от внешней среды, что влечет за собой естественный отбор. Каждая особь из популяции соответствует одному из значений целевой функции. Особь эволюционирует за счет мутаций и селекционирования. Получается новая популяция с лучшими характеристиками, то есть с лучшим значением целевой функции.

Генетический алгоритм (Genetic Algorithm) – классический эволюционный алгоритм, основанный на принципе естественного эволюционного отбора. Основной идеей является борьба за существование между решениями задачи. Каждое решение записывается в виде некоторого вектора значений (аллелей), который называется хромосомой, или особью. Совокупность решений называют популяцией. Каждая особь в популяции оценивается значением целевой функции, рассчитанной на основе значений из хромосомы. Более перспективные решения проходят в следующую стадию и оказывают влияние на «потомство», т.е. на вновь генерируемые решения.

Метод динамических сеток (Variable Mesh Optimization) – метод, работа которого заключается в эволюции начальной популяции, называемой сеткой, во время которой происходит смена одного поколения другим путем расширения и последующего сокращения популяции.

Метод дифференциальной эволюции (Differential Evolution) – метод, основанный на формировании случайно популяции, состоящей из векторов (возможных решений). В каждом поколении происходит образование новых векторов путем комбинации трех случайных векторов из прошлого поколения. Если этот новый вектор оказывается лучше старого, то он заменяется, иначе остается.

Метод рассеивания (Scatter Search) – метод, который был впервые представлен в 1977 г. как эвристический метод для целочисленного программирования. Однако в реальности он практически не применялся и не изучался вплоть до 1990 года, когда был

представлен на семинаре EPFL «Исследования и методы поиска с помощью искусственного интеллекта» (Лозанна, Швейцария).

Метод рассеивания активно выбирают при проектировании процедур решения многих сложных задач комбинаторной оптимизации. Его использование постоянно растет. Увеличивается количество ежегодных публикаций.

С помощью метода рассеивания решаются задачи для поточного цеха, мастерских, планирования проектов, дизайна территории, финансовые и статистические задачи, производственные задачи, в том числе для производства угля и стали, и другие.

Активное использование метода рассеивания обнаруживает резервы для его дальнейшего усовершенствования.

Методы искусственных иммунных систем (Artificial Immune System) – это класс автоматизированных вычислительных систем, которые основаны на принципах и процессах иммунной системы позвоночных. Обычно такие алгоритмы используют память и обучаемость иммунной системы для решения заданных проблем.

Искусственная иммунная система (ИИС) – адаптивная вычислительная система, использующая модели, принципы, механизмы и функции, описанные в теоретической иммунологии, которые применяются для решения прикладных задач.

ИИС появились в середине 1980-х годов в статьях Фармера, Паккарда и Перельсона (Farmer, Packard, Perelson, 1986), а также Берсини и Варела (Bersini, Varela, 1990) по иммунным сетям. Однако основы ИИС утвердились только к середине 1990-х годов. Форрест (Forrest) и Кепхарт (Kephart) опубликовали первую статью по ИИС (негативный отбор) в 1994, и Дасгупта (Dasgupta) провел обширные исследования негативного алгоритма отбора. Хант и Кук (Hunt, Cooke) начали работу над иммунным сетевым алгоритмом в 1995; Тиммис и Нил (Timmis, Neal) продолжили эту работу и внесли некоторые улучшения. Первая книга по искусственным иммунным системам вышла под редакцией Дасгупты (Dasgupta) в 1999 году.

Несмотря на то, что природные иммунные системы изучены далеко не полностью, на сегодня существуют, по меньшей мере, три теории, объясняющие функционирование иммунной системы и описывающие взаимодействие ее элементов, а именно: теория негативного отбора, теория клональной селекции и теория иммунной сети. Они легли в основу создания трех алгоритмов функционирования ИИС.

1.1.2. Методы «роевого» интеллекта

Алгоритмы роевого интеллекта (Swarm Intelligence) основаны на изучении поведения колоний живых организмов, реализующих для всей колонии «оптимальное»

поведение и зарекомендовавшие себя для решения сложных комбинаторных задач. Алгоритмы предполагают случайность и беспорядочность (аналогия со стаями муравьев, пчел и т.д.). Все начинается со случайной начальной позиции начальной популяции, затем происходит миграция агентов в пространстве поиска таким образом, чтобы приблизиться к искомому экстремуму целевой функции. Механика миграции агентов на каждом шаге идентична. Механизм обучения заимствуется из информации о поведении конкретных видов живых существ (муравьи, пчелы, светлячки и пр.).

Метод муравьиных колоний (Ant Colony Optimization) – метод, общий принцип которого заключается в моделировании поведения муравьиной колонии в естественных условиях. Муравьи выделяют некое химическое вещество, называемое феромоном, для них этот гормон является чем-то вроде языка общения. С помощью него (феромона) муравьи, например, объясняют, где искать пищу и т.п. При движении они оставляют за собой след, наткнувшись на который муравей принимает решение, следовать по нему или пойти другим путем. Чем больше особей прошло по одному пути, тем больше гормона «лежит на земле» и, следовательно, он (путь) становится более предпочтительным для остальных. В методе отсутствуют явные лидеры, каждый член колонии одинаково равно вносит вклад в поиск решения. Потому что выбор направления (генерация новых координат) определяется в два этапа:

выбирается случайная плотность вероятности методом рулетки с вероятностью выбора (чем лучше значение, тем больше шансов пойти по лучшему пути);

формируется новая координата, определяемая параметрами математического ожидания и среднеквадратическим отклонением.

Методы пчелиных колоний (Bees Algorithms) – методы, общий принцип которых заключается в моделировании действий пчел во время поиска и сбора нектара. Пчелы вылетают на разные участки и производят поиск нектара, вернувшись в улей, танцуют перед остальными специальный танец. Этим танцем они «объясняют», где находится нектар, как до него добраться и т.д. Группы незанятых летных пчел выдвигаются в лучшие места, следовательно, эти пчелы становятся занятыми. Они производят сбор в окрестностях лучших источников и возвращаются. Далее передают остальным свои результаты, и незанятые пчелы образуют группы, вылетают на точки, и так этот процесс повторяется.

Искусственный метод пчелиных колоний (Artificial Bee Colony) – метод, идея которого состоит в том, чтобы смоделировать поведение пчел при поиске и сбора нектара. Пчелы вылетают на разные участки и производят поиск нектара в окрестности этих участков. Причем на один участок вылетает одна пчела. Каждая особь имеет память, она запоминает

параметры места, на которое вылетела. Впоследствии, если новый «источник» нектара оказался лучше, чем предыдущий (меньшее значение ЦФ), то пчела запоминает его. Вернувшись в улей, каждая пчела танцует перед остальными специальный танец, логика танца такая же, что и в методе роя пчел. Незанятые пчелы вероятностным образом выбирают, куда полетят. После этого пчелы повторяют процесс поиска, запоминания и передачи остальным сведений о позиции. Если в процессе поиска решение не обновляется в течение итераций, то пчела, вылетавшая на этот источник, «забывает» его и выбирает случайную точку в области допустимых решений.

Метод частиц в стае (Particle Swarm Optimization Strategy) – метод, идея которого состоит в том, чтобы смоделировать поведение животных (стаи) во время поиска пищи. Каждая особь из стаи рассматривается, как отдельная частица, имеющая свою скорость и положение в пространстве. Частицы используют свой прошлый опыт на каждой итерации, делятся своим положением и «качеством решения» со всеми, для того чтобы скорректировать направление движения, если это необходимо, но получают это не все, обуславливая тем, что имеются внешние факторы, например, ветер, шумы, погода или чувствительность особи (старость и т.п.). Так же метод перенимает такое качество, как лидерство, в случае локальной неудачи отдельной частицы, она принимает решение следовать за наблюдаемым лидером или ориентируется на собственный опыт.

Метод, имитирующий поведение светлячков (Glowworms Swarm Optimization) – метод, согласно которому светлячки случайным образом располагаются в области допустимых значений. Светлячки привлекают других особей с помощью излучаемого света. Те, у которых тусклее свет, следуют за более яркой особью. Чем больше расстояния от источника света, тем тусклее он виден. Если же в поле зрения не наблюдается более перспективной особи, перемещается в случайную точку.

Метод, имитирующий поведение рыб в стае (Fish School Search) – алгоритм, имитирующий поиск рыбой пищи для роста и размножения. Каждая рыба имеет память и вес, который набирается при питании и сбрасывается во время плавания. Косяк запоминает наилучшие места для подкормки, способные прокормить большую часть рыб. Эволюционирование косяка происходит путем размножения, обмена опытом и информацией, а также благодаря совместному движению.

Метод имитации поведения бактерий (Bacterial Foraging Optimization) – метод, разработанный Кевином Пассино в 2002 году как продолжение тенденции алгоритмов на основе роя. Применение стратегии группового кормодобывания роя бактерий E. coli для мультиэкстремальных функций является ключевой идеей данного алгоритма. Бактерии ищут

питательные вещества таким образом, чтобы получить максимальную энергию в единицу времени. Также бактерии общаются между собой, посылая сигналы. Бактерия принимает решение о добыче пищи после рассмотрения двух предыдущих факторов. Процесс, в котором бактерия перемещается, делая небольшие шаги в поисках питательных веществ, называется хемотаксисом, и ключевая идея метода имитации поведения бактерий – имитировать хемотаксическое движение виртуальных бактерий в пространстве поиска пищи.

1.1.3. Методы, имитирующие физические процессы

В основе методов лежат физические законы и явления, такие как реструктуризация кристаллической решетки металла в процессе отжига, замерзание или нагревание жидкости, всемирный закон тяготения и т.д.

Метод гравитационной кинематики (Central Force Optimization) –детерминированный метод, в основе которого лежит закон всемирного тяготения. Уникальность его состоит в том, что каждое решение целевой функции взято в качестве массы тела, которое участвует в гравитационном взаимодействии с другими телами.

Гравитационный поиск оперирует двумя законами:

- тяготения: каждая частица притягивает другие, и сила притяжения между двумя частицами прямо пропорциональна произведению их масс и обратно пропорциональна расстоянию между ними;
- движения: текущая скорость любой частицы равна сумме части скорости в предыдущий момент времени и изменению скорости, которое равно силе, с которой воздействует система на частицу, деленной на инерциальную массу частицы.

Метод имитации отжига (Simulated Annealing) – метод, моделирующий кристаллическую решетку во время нагревания и охлаждения металла. Под действием высокой температуры атомы стараются занять места с наименьшей энергией. По мере охлаждения кристаллическая решетка сжимается, и атомы все меньше и меньше передвигаются на новые места. При высокой температуре выше вероятность принять худшее решение, чтобы проскочить локальный минимум.

Адаптивный метод имитации отжига (Adaptive Simulated Annealing) – метод, устраняющий некоторые недостатки базового метода. Все алгоритмы имитации отжига допускают с некоторой вероятностью переход в состояние с более высоким значением целевой функции для того, чтобы точка могла покинуть окрестности локального минимума. Для обеспечения этого свойства необходимо снижать температуру очень медленно, что увеличивает время подсчета. В адаптивном методе отжига этот недостаток устраняется

путем ввода нового закона уменьшения температуры и возможностью ее изменения по каждой координате отдельно. Данный метод относится к классу стохастических методов. На каждой итерации разрешается переход в состояние с более высоким значением целевой функции с некоторой вероятностью, рассчитываемой исходя из закона Больцмана-Гиббса. В целом, метод сильно схож с обычным методом имитации отжига.

Метод поиска гармонии (Harmony Search) – метод, разработанный и предложенный в 2001 году автором Geem Zong Woo. Некоторые авторы заявляют, что данный алгоритм был навеян игрой джаз-музыкантов, другие говорят, что в основе лежит просто процесс создания приятной мелодии. Сути это не меняет. Важно лишь то, что опытный музыкант может достаточно быстро как сыграться с другими музыкантами, так и сымпровизировать что-нибудь прекрасное. Это и легло в основу алгоритма.

Классический алгоритм гармонического поиска оперирует понятием гармонической памяти (по аналогии с родительским пулом генетических алгоритмов). Здесь хранятся вектора, представляющие приближенные решения задачи оптимизации. Изначально они генерируются случайным образом в области, которая исследуется на наличие максимума или минимума (в зависимости от того, что вам нужно). Размер этой памяти, естественно, ограничен. Далее начинается магия импровизации (итеративная часть алгоритма).

Таким образом, метод имитирует процесс импровизации музыканта-исполнителя. В процессе исполнения он подбирает нужную ноту для достижения гармонии. Во время поиска каждое решение порождает значение ЦФ с целью достижения глобального экстремума. Используются идеи метода частиц в стае, метода отжига, стохастического градиента.

Сама импровизация заключается в следующем:

генерируется пробный вектор (либо абсолютно случайно, либо с использованием векторов из памяти);

пробный вектор модифицируется (происходит небольшой сдвиг в компонентах, если пробный вектор был создан с помощью памяти);

модифицированный пробный вектор заменяет худший из имеющихся в памяти векторов.

На множестве генерируется заданное количество решений, для каждого из которых вычисляется значение искомой функции. Координаты решения и значения функции помещаются по строкам в матрицу – память гармонии. Среди всех решений выбирается наихудшее. Затем генерируется новое решение и сравнивается с наихудшим. Если по значению функции оно лучше, то его помещают в память вместо наихудшего. Так продолжается до максимального количества итераций.

Для получения очередной координаты с некоторой вероятностью выбирается соответствующая координата решения, выбранного случайным образом из памяти. Иначе выбирается случайное значение на промежутке. Значение, взятое из памяти, с заданной вероятностью корректируется с помощью небольшого приращения. Если корректировки не было, то используется нескорректированное значение. Вся эта процедура позволяет выйти за область локального минимума, для поиска новой гармонии. Это и есть процесс импровизации.

1.1.4. Мультистартовые методы

Мультистартовые методы (Multi-start Methods) – методы, основанные на многократном отыскании локальных минимумов из разных начальных точек. При последующем поиске исключаются повторные спуски в те же локальные минимумы, после происходит генерирование новых перспективных начальных точек.

Основная сложность в том, что для высокой надежности поиска точки глобального минимума нужно взять количество начальных точек для локальных алгоритмов больше, чем число локальных минимумов функции, которое обычно неизвестно.

Жадный адаптивный метод случайного поиска (Greedy Randomized Adaptive Search Procedure) – метод, использующий идею мультистарта. На каждой итерации есть две фазы: построение и локальный поиск. В результате первой фазы порождаются решения хорошего качества для второй фазы. Затем после второй фазы полученные точки берутся в качестве начальных для первой фазы, и процедура продолжается до нахождения наилучшего приближенного решения. Этот вид жадного метода рандомизированного построения также известен как **полужадный эвристический метод**, впервые описанный авторами J. Pirie Hart и Andrew W. Shogan (1987).

Жадная рандомизированная адаптивная процедура поиска (GRASP) была впервые представлена Фео и Резенде (Feo, Resende, 1989).

Существуют вариации классического алгоритма, например, Reactive GRASP. В этом варианте основной параметр, определяющий ограниченность списка кандидатов (restrictive list of candidates, RCL) на этапе строительства, саморегулируется в соответствии с качеством найденных ранее решений. Существуют также методы ускорения поиска, такие как изменение стоимости, функции смещения, запоминание и обучение, а также локальный поиск частично построенных решений.

Метод направленного табу-поиска (Taboo Search) – алгоритм поиска с запретами, основоположником которого является Ф. Гловер, который предложил принципиально новую

схему локального поиска. Она включает три фазы: исследовательская, перераспределительная, интенсивно-уточняющая. На исследовательской фазе генерируются точки вблизи текущего решения, в тех областях, где еще не производился поиск. Посещенные области помещаются в список посещенных областей на перераспределительной фазе. На интенсивно-уточняющей фазе улучшаются хорошие точки, найденные в первых двух фазах, с целью получения результата с заданной точностью.

1.2 Эволюционные методы

Рассмотрим более детально некоторые эволюционные методы

1.2.1 Генетические алгоритмы

Генетические алгоритмы с бинарным кодированием

ГА имитируют в своей работе природные способы оптимизации:

- генетическое наследование
- естественный отбор.

Целевая функция $f(x)$ эквивалентна природному понятию *приспособленности* живого организма

Вектор параметров $x=(x_1,x_2,...,x_n)^T$ целевой функции называется *фенотипом*, а отдельные его параметры x_i *признаками,* $i=1,...,n$. Любой живой организм может быть представлен своим генотипом и фенотипом.

Генотип – это совокупность наследственных признаков, информация о которых заключена в хромосомном наборе.

Фенотип – совокупность всех признаков и свойств организма, формирующихся в процессе взаимодействия его генотипа и внешней среды. Каждый ген имеет свое отражение в фенотипе.

ГА ведут поиск решения только на уровне генотипа.

Каждую координату x_i вектора $x=(x_1,x_2,...,x_n)^T \in D$ представляют в некоторой форме s_i удобной для использования в ГА и называемой *геном*. Для этого необходимо осуществить преобразование, в общем случае не взаимно однозначное, вектора параметров $x=(x_1,x_2,...,x_n)^T \in D$ в некоторую структуру $s=(s_1,s_2,...,s_n)^T \in S$ называемую *хромосомой (генотипом, особью)*:

$$D \xrightarrow{e} S,$$

где e – функция кодирования, S – пространство представлений.

Для того чтобы уметь восстановить по хромосоме решение, необходимо задать обратное преобразование:

$$S \xrightarrow{e^{-1}} D,$$

где e^{-1} – функция декодирования.

В пространстве представлений S вводится так называемая *функция приспособленности* $\mu(s)$: $S \xrightarrow{\mu} R$, где R – множество вещественных чисел, аналогичная целевой функции $f(x)$ на множестве D. Функцией $\mu(s)$ может быть любая функция, удовлетворяющая следующему условию:

$$\forall x^1, x^2 \in D : s^1 = e(x^1), s^2 = e(x^2), s^1 \neq s^2, \text{если } f(x^1) > f(x^2), \text{то } \mu(s^1) > \mu(s^2).$$

Решение исходной оптимизационной задачи, например, $f(x^*) = \max\limits_{x \in D} f(x)$, сводится к поиску решения s^* другой задачи оптимизации:

$$\mu(s^*) = \max_{s \in S} \mu(s). \tag{1.2}$$

При решении используются конечные наборы

$$I = \left\{ s^k = (s_1^k, s_2^k, ..., s_n^k)^T,\ k = 1, 2, ..., K \right\} \subset S$$

возможных решений, называемые *популяциями*, где s^k – хромосома с номером k, K – размер популяции, s_i^k – ген с номером i.

Затем осуществляется обратное преобразование:

$$x^* = e^{-1}(s^*). \tag{1.3}$$

Существуют различные способы задания функции приспособленности:

а) как правило полагают $\mu(s) = f(e^{-1}(s))$;

б) в некоторых случаях необходимо, чтобы функция $\mu(s)$ принимала только положительные значения. Тогда функцию приспособленности можно представить, например, как

$$\mu(s^k) = f(s^k) + \left| \min_{k=1,2,...,m} f(s^k) \right| + 1 \text{ или } \mu(s^k) = \left(f(s^k) - \min_{k=1,2,...,m} f(s^k) + 1 \right)^2$$

Бинарное кодирование

а) Функция одной переменной.

Пусть задана сетка на интервале [a, b] $\{x_i\}: a = x_0, x_1, x_2 ..., x_n = b$.

Переменной x_i ставится в соответствие целое число $\gamma_i \in \{0,1,...,n\}$:

$$\gamma_i = \begin{cases} 0, & x_i = a, \\ i, & x_i \in (a,\ b),\ i = 1,2,...,n-1 \\ n, & x_i = b, \end{cases} \quad (1.4)$$

Далее производим кодирование чисел γ_i бинарным кодом. Число битов

$L = [\log_2(n+1)] + 1$, где $[\cdot]$ – целая часть числа. Таким образом, получаем закодированные числа γ_i в виде L - битовых строк c_i, состоящих из нулей и единиц.

б) Функция многих переменных.

Пусть для каждой координаты x_j, $j = 1,...,m$ задана сетка на интервале $[a_j,\ b_j]$

$\{x_{ji}\}: a_j = x_{j0}, x_{j1}, x_{j2} ..., x_{jn} = b_j$.

Переменной x_{ji} ставится в соответствие целое число $\gamma_{ji} \in \{0,1,...,n_j\}$:

$$\gamma_{ji} = \begin{cases} 0, & x_{ji} = a_j, \\ i, & x_{ji} \in (a_j, b_j), \\ n_j, & x_{ji} = b_j, \end{cases} \quad (1.5)$$

Далее производим кодирование чисел γ_{ji} бинарным кодом. Число битов

$L_j = [\log_2(n_j + 1)] + 1$, где $[\cdot]$ – целая часть числа. Таким образом, получаем закодированные числа γ_{ji} в виде L_j - битовых строк c_{ji}, состоящих из нулей и единиц.

Преобразование в битовую строку осуществляется с помощью обычного двоичного кодирования (позиционного кода) или *рефлексивного кода Грея*, который обладает свойством непрерывности бинарной комбинации: изменению кодируемого числа γ_i на единицу соответствует изменение бинарной комбинации только в одном разряде.

Декодирование. Сначала мы декодируем числа c_i (c_{ji}), т.е. переводим их в десятичные числа γ_i (γ_{ji}). И далее, каждому числу γ_i (γ_{ji}) ставим в соответствие значения переменных x_i (x_{ji}).

Общая схема алгоритма приведена на рис. 1.2. Более детальное рассмотрение алгоритма будет приведено в главах 2,3.

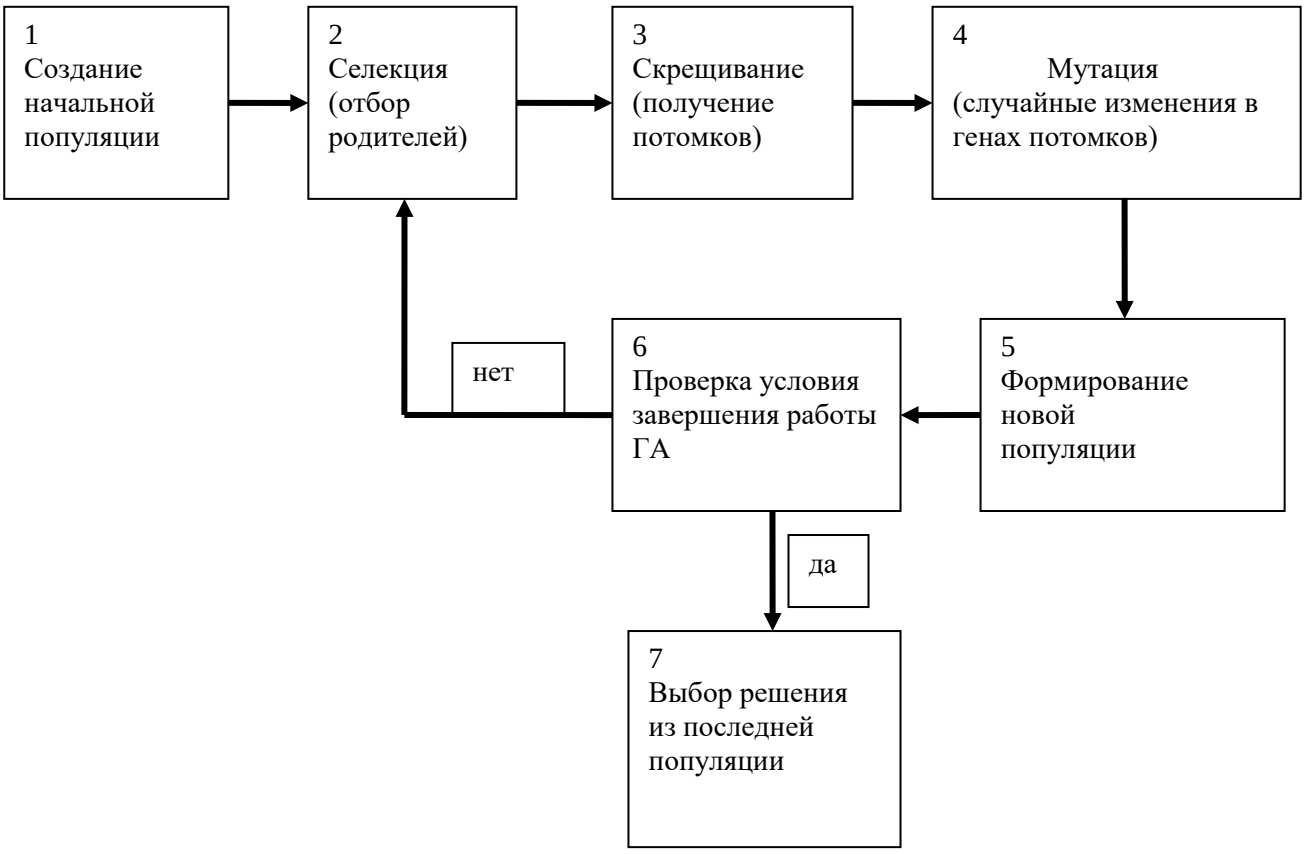


Рис 1.2 Общая схема работы генетического алгоритма (ГА)

Генетические алгоритмы с вещественным кодированием

Рассматриваемая в задаче (1.1) целевая функция $f(x)$ эквивалентна природному понятию приспособленности живого организма. Вектор параметров $x=(x_1,x_2,...,x_n)^T$ целевой функции называется фенотипом, а отдельные его координаты $x_i \in R$ – признаками, $i=1,...,n$. Любой вектор $x=(x_1,x_2,...,x_n)^T \in R^n$ считается хромосомой (генотипом,

особью), а каждая его координата $x_i \in R$ представляет собой ген. Фенотип совпадает с генотипом, а поиск фактически производится в пространстве фенотипов. На множестве допустимых решений D вводится так называемая функция приспособленности $\mu(x): D \xrightarrow{\mu} R$, где R – множество вещественных чисел, аналогичная целевой функции $f(x)$ Функцией $\mu(s)$ может быть любая функция, удовлетворяющая следующему условию:

$$\forall x^1, x^2 \in D: s^1 = e(x^1), s^2 = e(x^2), x^1 \neq x^2, если\ f(x^1) > f(x^2), то\ \mu(x^1) > \mu(x^2).$$

Решение исходной оптимизационной задачи $f(x^*) = \min_{x \in D}(f(x))$ сводится к поиску решения x_μ^* другой оптимизационной задачи:

$$\mu(x_\mu^*) = \min_{x \in D}(\mu(x)). \quad (1.6)$$

В силу выбора функции $\mu(x)$, решения задач (1.1) и (1.6) (хромосома) совпадают:

$$x^* = \arg\min_{x \in D} f(x) = \arg\min_{x \in D} \mu(x) = x_\mu^*. \quad (1.7)$$

При решении задачи (1.6) используются конечные наборы $I = \{x^k = (x_1^k, x_2^k, ..., x_n^k),\ k = 1, 2, ..., K\} \subset D$ возможных решений, называемых популяциями, где x^k – хромосома с номером k, K – размер популяции, x_i^k – ген с номером i.

Более детальное рассмотрение алгоритма будет приведено в главах 2,3.

1.2.2 Методы иммунных систем

Метод искусственных иммунных систем

Метод искусственных иммунных систем (ИИС) использует идеи, заимствованные из иммунологии, имитируя работу иммунной системы живого организма.

Иммунной системой живого организма называется подсистема, объединяющая органы и ткани, которые защищают организм от заболеваний. Назначение иммунной системы живого организма заключается в том, что она идентифицирует и уничтожает чужеродные тела, попавшие в организм, и совершенствуется, накапливая опыт борьбы с ними.

Антигеном называется вещество, которое воспринимается живым организмом как чужеродное и от которого организм пытается защититься. Для того чтобы организм смог

защититься от антигенов, в нем при помощи специальных иммунных клеток вырабатываются антитела.

Антителом называется вещество, которое распознает антиген и способствует его уничтожению. Если иммунные клетки выработали антитела, которые смогли распознать антиген, то информация об этих антителах сохраняется в клетках памяти.

Клеткой памяти называется иммунная клетка, которая сохраняет в себе информацию о новых антителах, способных распознать антиген, для того, чтобы в следующий раз, когда в организм попадет такой же или похожий антиген, иммунная система смогла работать эффективнее.

Целевая функция $f(x)$ эквивалентна природному понятию приспособленности иммунной клетки к борьбе с антигенами, т.е. способности клетки вырабатывать антитела. Поэтому будем называть целевую функцию $f(x)$ функцией приспособленности. Вектор параметров $x=(x_1,x_2,...,x_n)^T \in D$ целевой функции называется иммунной клеткой, которая вырабатывает антитела.

При решении задачи глобальной минимизации используются конечные наборы $I=\{x^k=(x_1^k,x_2^k,...,x_n^k),\ k=1,2,...,K\}\subset D$ возможных решений, называемых популяциями, где x^k – иммунная клетка с номером k, K – размер популяции. Чем меньше значение целевой функции $f(x^k)$, тем более иммунная клетка x^k приспособлена, т.е. способна вырабатывать антитела и подходит в качестве решения. Общая схема метода искусственных иммунных систем приведена на рис. 1.3.

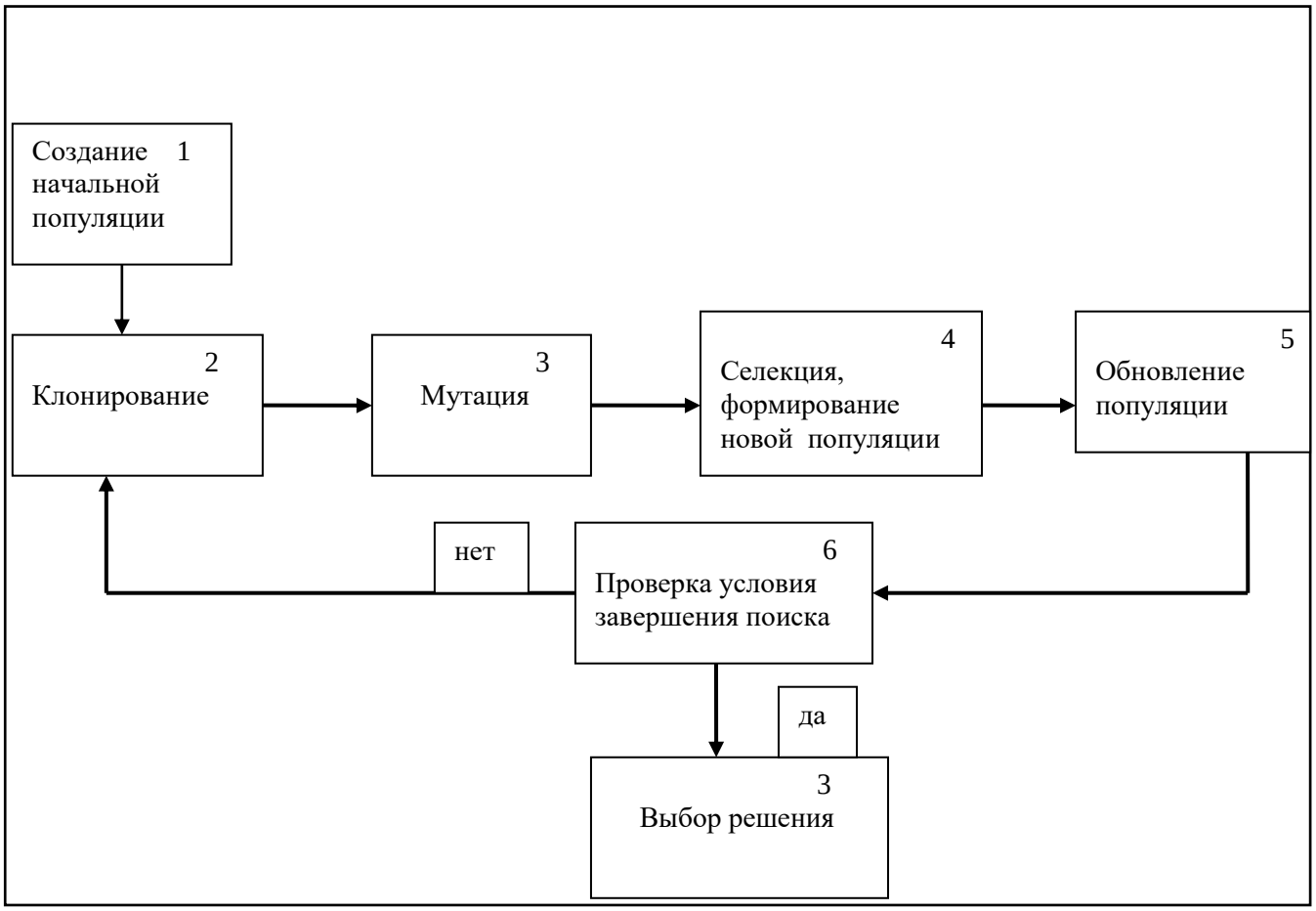


Рис. 1.3 Общая схема метода ИИС

Расширенный метод искусственных иммунных систем

Особенностью расширенного метода ИИС является сочетание локального и глобального поиска решения.

Локальный поиск реализуется в виде циклического итерационного процесса, во время которого к популяции применяются биологические операторы: клонирование, мутация и селекция. Таким образом, происходит смена популяции на новую, к которой, если условие окончания локального поиска не выполнены, опять применяются биологические операторы, и т.д. до выполнения условия окончания локального поиска. Средняя приспособленность популяции при этом будет расти. Если условие окончания
локального поиска выполнено, то начинается новая итерация глобального поиска.

Глобальный поиск реализуется в виде циклического итерационного процесса. Каждая итерация глобального поиска представляет собой локальный поиск. В конце каждой итерации происходит сокращение популяции с использованием идей кластеризации. Оставшиеся иммунные клетки помечаются как клетки памяти, после чего проверяются условия окончания работы метода. Если условия окончания не выполнены, то к популяции

присоединяются новые особи, количество которых пропорционально размеру популяции. Таким образом, размер популяции N в расширенном методе ИИС – переменная величина. Если условие окончания работы метода выполнено, то в качестве приближенного решения задачи из последней популяции выбираются иммунные клетки, которым соответствует наибольшее значение функции приспособленности.

Общий алгоритм расширенного метода ИИС приведен на рис. 1.4

Рис. 1.4 Общая схема работы расширенного метода ИИС

1.2.3 Метод рассеивания

В методе рассеивания рассматриваемая целевая функция $f(x)$ называется функцией приспособленности, а вектор параметров $x=(x_1,x_2,...,x_n)^T$ целевой функции – особью. Чем меньше значение целевой функции $f(x)$ тем более особь x приспособлена, т.е. подходит в качестве решения.

Свое название метод берет от способа формирования начального набора возможных решений, называемого базовым множеством особей. Генерация базового множества построена таким образом, чтобы особи в нем были достаточно различны (рассеяны) между собой. Это позволяет эффективно исследовать все множество допустимых решений в поставленной задаче.

Процедура поиска начинается с генерации базового множества особей A . Для этого сначала на отрезке $[a_i, b_i]$ изменения каждой координаты выделяются s подынтервалов одинаковой длины. В базовое множество последовательно добавляется по одной особи:

1) для каждого отрезка $[a_i, b_i]$ генерируется номер j_i подынтервала с вероятностью, обратно пропорциональной числу раз, которое этот подынтервал уже выбирался;

2) генерируются значения координат x_i, $i = 1,...,n$ случайным образом внутри выбранных подынтервалов;

3) если минимальное расстояние между сформированной особью и особями базового множества больше заданной величины σ, то особь добавляется в базовое множество A . Иначе – особь не добавляется, процесс рассеивания продолжается с п.1.

Процесс формирования базового множества гарантирует разнообразие входящих в него особей и продолжается до тех пор, пока не будет выбрано нужное количество особей.

При решении задачи используются конечные наборы $I = \{x^k = (x_1^k, x_2^k, ..., x_n^k), k = 1, 2, ..., Np\} \subset D$ возможных решений, называемые популяциями, где x^k – особь с номером k, Np – размер популяции.

Идеи рассеивания применяются также и при формировании начальной популяции: $Np = b_1 + b_2$ особей выбираются из базового множества особей A. Здесь b_1 – количество особей, выбираемых по качеству (наилучшие особи по величине функции приспособленности), b_2 – количество особей, выбираемых по расстоянию т.е. суммарное расстояние от них до уже имеющихся в начальной популяции особей должно быть минимально.

Метод рассеивания имитирует эволюцию начальной популяции $I_0 = \{x^j, j = 1, 2, .., Np \,|\, x^j = (x_1^j, ..., x_n^j) \in D\}$ и представляет собой итерационный процесс, исследующий множество D. Во время работы метода на каждой итерации к популяции применяются биологические операторы: селекция и скрещивание, после чего происходит замена особей с низким уровнем приспособленности на новые. Таким образом, формируется

новая популяция. Метод заканчивает работу после того, как будет исчерпано заданное количество вычислений значения функции приспособленности. В качестве приближенного решения задачи из последней популяции выбираются особи, которым соответствует наименьшее значение целевой функции. Следует отметить, что размер базового множества особей A должен быть достаточно большим, чтобы обеспечить работу метода до выполнения условий окончания.

Метод рассеивания сочетает в себе локальный и глобальный поиск решения.

Глобальный поиск реализуется в виде циклического итерационного процесса. Каждая итерация глобального поиска представляет собой локальный поиск. В конце каждой итерации происходит сокращение популяции: удаляются b2 особей с наихудшей приспособленностью, после чего проверяются условия окончания работы метода. Если условия окончания не выполнены, то к популяции присоединяются b2 новых особей из базового множества таким же образом, как и при формировании начальной популяции, после чего начинается новая итерация глобального поиска. Если условие окончания работы метода выполнено, то в качестве приближенного решения задачи из последней популяции выбираются особи, которым соответствует наименьшее значение целевой функции.

Локальный поиск реализуется в виде циклического итерационного процесса, во время которого к популяции применяются биологические операторы: селекция и скрещивание. Во время селекции из особей текущей популяции составляются всевозможные родительские пары, каждая из которых во время скрещивания порождает новое решение, называемое потомком. Все потомки помещаются в множество потомков, после чего происходит их сравнение с особями из текущей популяции и принимается решение о замене особей потомками. Таким образом, происходит смена популяции на новую, к которой, если условия окончания локального поиска не выполнены, опять применяются биологические операторы, и т.д. до выполнения условия окончания локального поиска. Средняя приспособленность популяции при этом будет расти. Если условие окончания локального поиска выполнено, то начинается новая итерация глобального поиска.

На рис. 1.5 приведен общий алгоритм метода рассеивания.

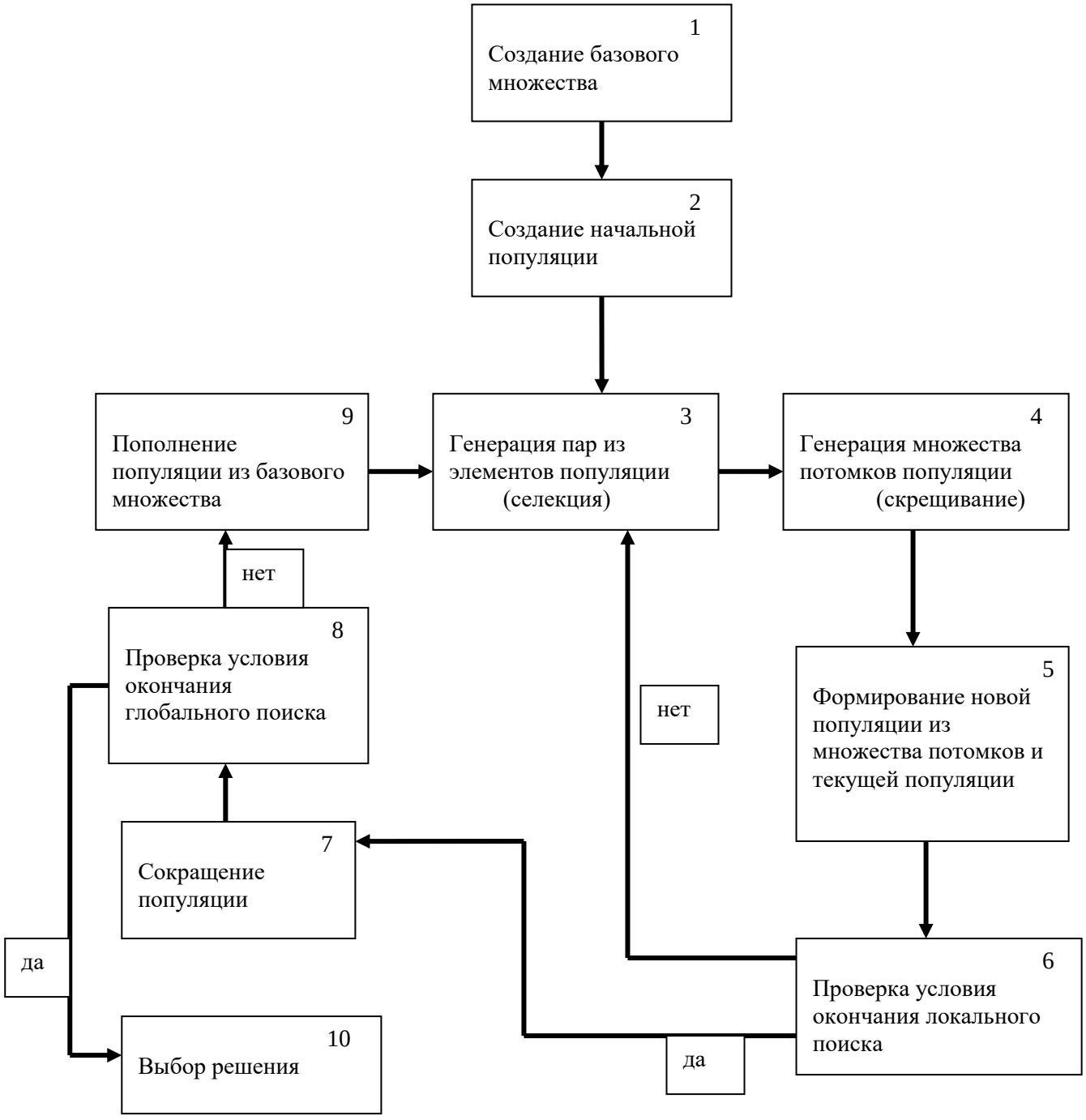


Рис. 1.5 Общий алгоритм метода рассеивания

1.2.4 Эволюционная стратегия преобразования ковариационной матрицы

Подобно другим эволюционным методам, в эволюционной стратегии на каждой итерации образуется новая популяция. Генерация новых особей, так называемая мутация, происходит случайным образом согласно формируемому эволюционной стратегией распределению. Лучшие найденные решения участвуют в дальнейшем эволюционном процессе. Для генерации новых особей (векторов) в эволюционных стратегиях применяется нормальное распределение, параметры которого влияют на эффективность работы реализующего стратегию алгоритма. Особенностью эволюционной стратегии является изменение параметров распределения на каждой итерации в соответствии с успешностью поиска. Стратегия сформирована таким образом, чтобы преобразовывать текущие параметры распределения таким образом, чтобы они как можно лучше подходили для успешной работы на последующих шагах.

В эволюционной стратегии преобразования ковариационной матрицы рассматриваемая целевая функция $f(x)$ называется функцией приспособленности, а вектор параметров $x=(x_1,x_2,...,x_n)^T$ целевой функции – особью. Чем меньше значение целевой функции $f(x)$, тем более особь x приспособлена, т.е. подходит в качестве решения. Каждый вектор $x=(x_1,x_2,...,x_n)^T \in D$ является возможным решением поставленной оптимизационной задачи.

При решении задачи глобальной оптимизации используются конечные наборы $I=\{x^k=(x_1^k,x_2^k,...,x_n^k), k=1,2,...,Np\}\subset D$ возможных решений, называемые популяциями, где x^k – особь с номером k, Np – размер популяции. Применение эволюционной стратегии сводится к исследованию множества D при помощи перехода от одной популяции к другой. Чем меньше значение целевой функции $f(x^k)$ тем лучше особь x^k приспособлена.

Рассматриваемая стратегия имитирует эволюцию начальной популяции $I_0=\{x^j, j=1,2,..,Np \,|\, x^j=(x_1^j,...,x_n^j)\in D\}$ и представляет собой итерационный процесс. На каждой итерации генерируется новая популяция согласно сформированному стратегией распределению. После этого производится анализ новой популяции и корректировка параметров распределения: шага μ и ковариационной матрицы C , после чего осуществляется переход на новую итерацию. Процедура поиска завершается после того, как сформируется заданное количество популяций. В качестве приближенного решения задачи

из последней популяции выбирается особь, которой соответствует наименьшее значение функции приспособленности.

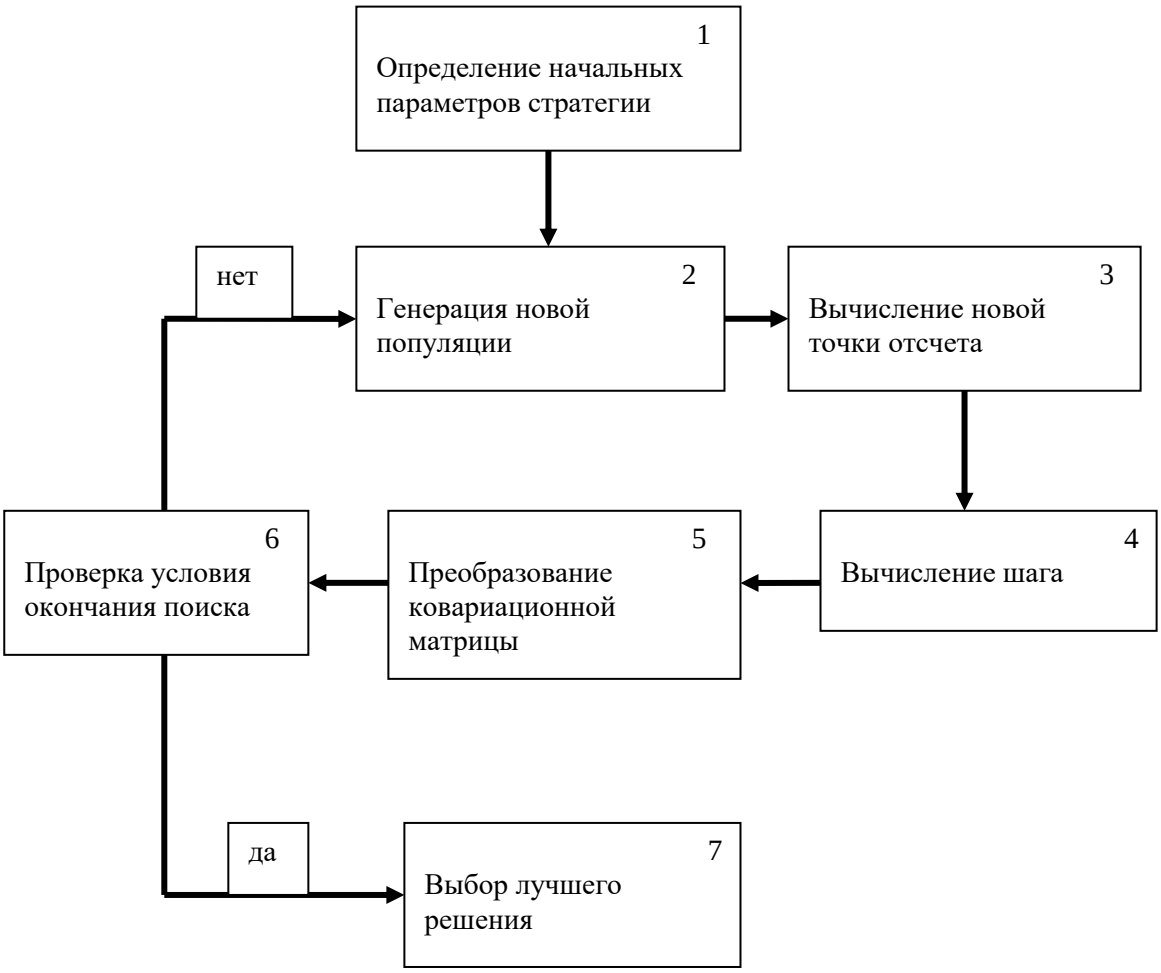


Рис. 1.6 Общий алгоритм эволюционной стратегии преобразования ковариационной матрицы

Алгоритм эволюционной стратегии преобразования ковариационной матрицы представлен на рис. 1.6.

1.2.5 Метод динамических сеток

При работе метода происходит эволюция начальной популяции – смена одного поколения другим путем расширения и последующего сокращения популяции. В методе динамических сеток популяция представляется в виде некоторой сетки, состоящей из набора решений, называемых узлами. В процессе поиска сетка подвергается изменениям: расширению – добавлению новых узлов в сетку, и сокращению – удалению узлов,

расположенных слишком близко друг к другу. В методе динамических сеток рассматривается целевая функция $f(x)$. Каждому узлу ставится в соответствие вектор параметров $x=(x_1,x_2,...,x_n)^T \in D$ целевой функции. Каждый вектор $x=(x_1,x_2,...,x_n)^T \in D$ является возможным решением поставленной оптимизационной задачи. При решении задачи используются конечные наборы $I=\{x^j=(x_1^j,...,x_n^j), j=1,2,..,Np\} \subset D$ возможных решений, называемые популяциями или сеткой, где x^j – узел с номером j, Np – количество узлов в сетке. Применение метода динамических сеток сводится к исследованию множества D при помощи изменения сетки и перехода от одной популяции к другой. Чем меньше значение целевой функции $f(x^j)$, тем более узел x^j подходит в качестве решения.

Метод динамических сеток имитирует эволюцию начальной популяции $I_0=\{x^j, j=1,2,..,Np \,|\, x^j=(x_1^j,...,x_n^j) \in D\}$ и представляет собой итерационный процесс. Во время работы метода на каждой итерации происходит расширение (локальное, глобальное и дополнительное) и последующее сокращение сетки. Таким образом, формируется новая сетка. Критерием окончания поиска является достижение заранее заданного количества n вычислений целевой функции. В качестве приближенного решения задачи из последней популяции выбирается узел, которому соответствует наименьшее значение целевой функции.

В процессе расширения происходит добавление новых узлов в сетку. Стадия расширения состоит из нескольких этапов: 1) локальное расширение, 2) глобальное расширение, 3) дополнительное расширение. На этапе локального расширения в окрестности каждого p-го узла, $p=1,2,...,P$, выбирается некоторое заранее заданное число K, ближайших по расстоянию узлов, называемых соседними узлами. Среди соседних узлов выбирается наилучший и, если данный соседний узел лучше p-го узла, то производится генерация нового узла в направлении наилучшего из K соседних узлов.

На этапе *глобального* расширения для всех узлов сетки, кроме наилучшего узла сетки x^{best}, производится генерация нового узла в направлении наилучшего узла сетки.

Если при локальном и глобальном расширении сгенерировано узлов меньше, чем заранее заданное число *N*, то выполняется дополнительное расширение сетки. Путем генерации новых узлов производится исследование новых участков множества допустимых

решений. На последующей за расширением стадии сокращения в методе динамических сеток происходит удаление слишком близких друг к другу решений, таким образом, стратегия метода направлена на поддержание достаточного разнообразия узлов в сетке.

Общий алгоритм динамических сеток представлен на рис. 1.7

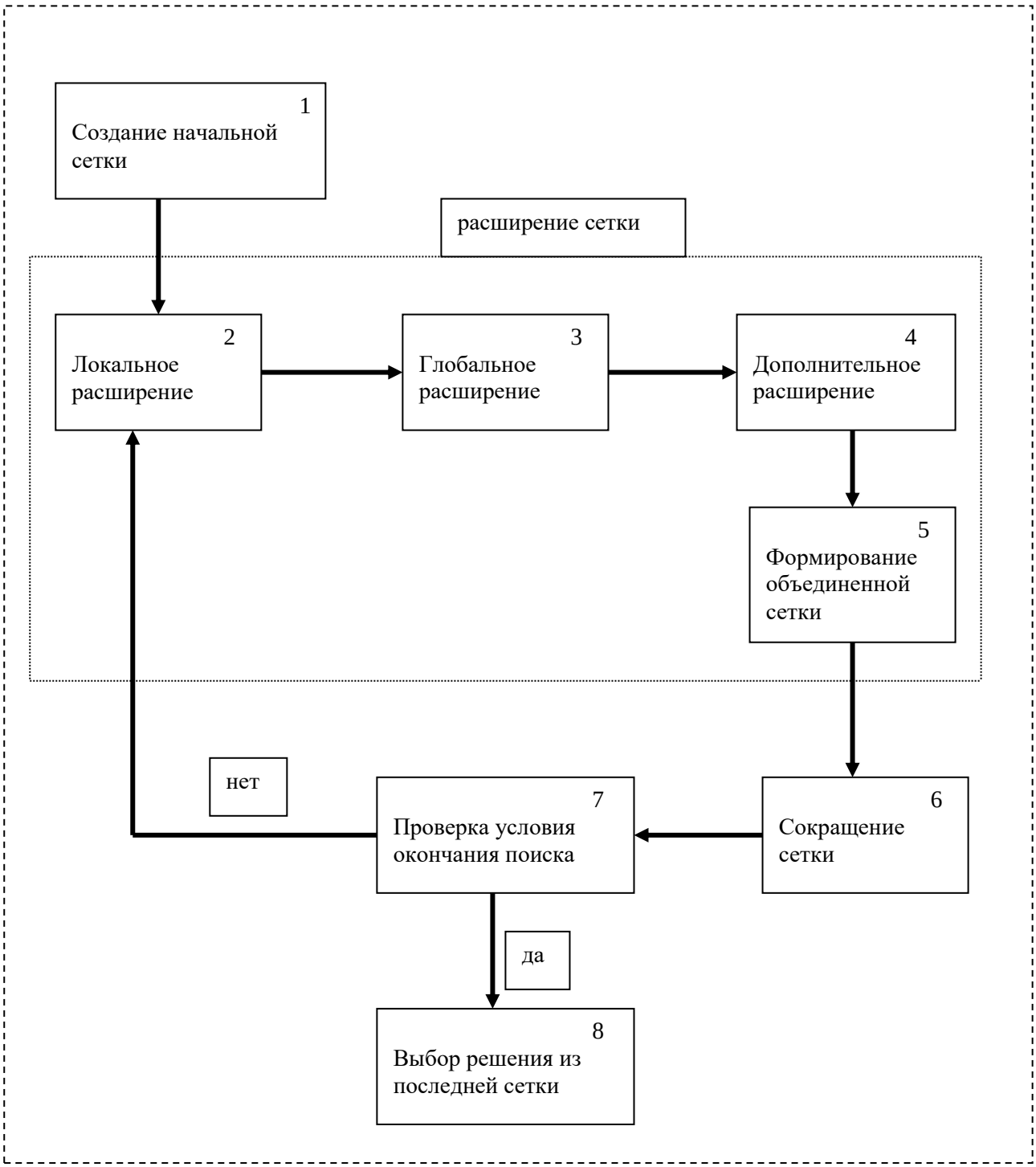


Рис. 1.7 Алгоритм метода динамических сеток

Вопросы для самопроверки

1. Приведите классификацию эвристических алгоритмов оптимизации.
2. Перечислите эволюционные алгоритмы поиска глобального минимума функции.
3. Перечислите методы роевого интеллекта.
4. Перечислите методы, имитирующие физические процессы.
5. С помощью какой операции осуществляется преобразование вектора параметров $x = (x_1, x_2, ..., x_n)^T \in D$ в хромосому пространства представлений S?
6. С помощью какой операции осуществляется обратное преобразование хромосомы из пространства представлений S в решение, заданном на пространстве параметров D?
7. Что такое бинарное кодирование и декодирование?
8. Что понимают под вещественным кодированием?
9. Дайте понятие искусственной иммунной системы.
10. Опишите расширенный метод искусственных иммунных систем.
11. Опишите процедуру поиска в методе рассеивания.
12. Особенности эволюционной стратегия преобразования ковариационной матрицы.
13. Особенности метода динамических сеток.

Литература к теме 1

1. Пантелеев А.В. Метаэвристические алгоритмы поиска глобального экстремума. – М: Изд-во МАИ-ПРИНТ, 2009. – 160с.
2. Пантелеев А.В. Метаэвристические алгоритмы поиска условного экстремума. [Электронный ресурс]. – Режим доступа: hse.ru›data/2014/02/10/1327531464/14 03 2013 A (дата обращения 10.11.2023)

ТЕМА 2
ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ ОПТИМИЗАЦИИ

2.1 Основные понятия генетических алгоритмов

2.1.1. Постановка задачи

Идея использования принципов биологической эволюции для решения оптимизационных задач возникала в различных вариантах у ряда авторов. Первые публикации на эту тему появились в 1960-х годах, а уже в 1975 г. вышла книга Холланда, в которой был предложен первый генетический алгоритм [1].

В настоящее время с помощью генетических алгоритмов решено множество различных задач [2-20]. Генетические алгоритмы нашли широкое практическое применение в менеджменте и управлении для решения задач поиска оптимальных решений, формирования моделей и прогнозирования значений различных показателей, поиске глобального минимума многоэкстремальных функций.

Генетический алгоритм (англ. genetic algorithm) — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию. Является разновидностью эволюционных вычислений (англ. evolutionary computation). Другими словами, генетический алгоритм — это метод перебора решений для тех задач, в которых невозможно найти решение с помощью математических формул. Однако простой перебор решений в сложной многомерной задаче – это бесконечно долго. Поэтому генетический алгоритм перебирает не все решения, а только лучшие. Алгоритм берёт группу решений и ищет среди них наиболее подходящие. Затем немного изменяет их – получает новые решения, среди которых снова отбирает лучшие, а худшие отбрасывает. Таким образом, на каждом шаге работы алгоритм отбирает наиболее подходящие решения (проводит селекцию), считая, что они наследующем шаге дадут ещё более лучшие решения (см. рис.2.1).

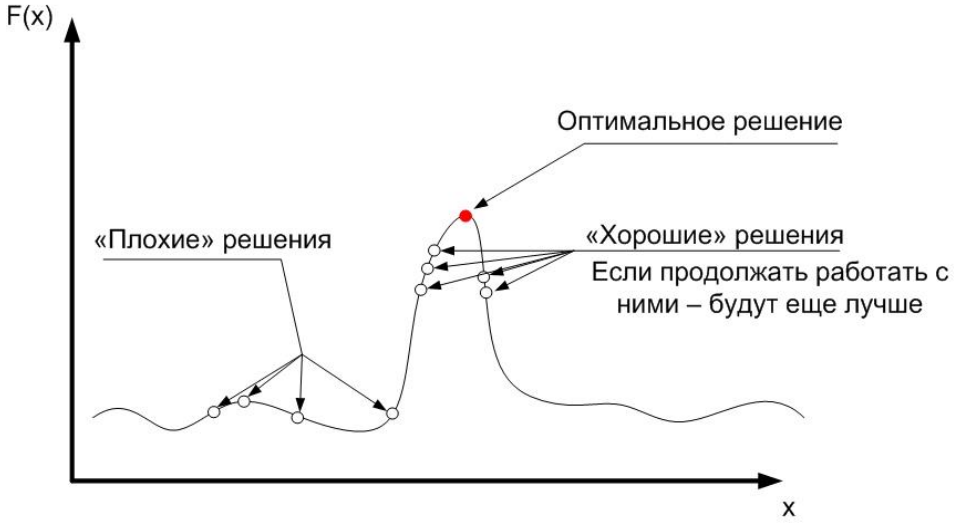


Рис. 2.1. Выбор наилучших решений [22].

Для генетических алгоритмов (ГА) характерны все черты, принадлежащие классу эвристических методов.

Эвристический алгоритм, или эвристика, определяется как алгоритм со следующими свойствами:

1. Он обычно находит хорошие, хотя не обязательные оптимальные решения (см. рис. 2.1)
2. Его можно быстрее и проще реализовать, чем любой известный точный алгоритм (т.е. тот, который гарантирует оптимальное решение) [23].

Основная общая черта всех эвристических методов оптимизации в том, что все они начинают с более или менее случайного начального решения задачи. Затем происходит итеративный поиск нового решения по какому-либо правилу и на каждом шаге осуществляется оценка промежуточных результатов, в конце концов сообщается лучшее найденное в процессе поиска решение. Выполнение итеративного поиска обычно заканчивается тогда, когда через заданное число итераций никаких существенных улучшений решения найдено не было. На каждом шаге новое решение формируется на основе предыдущих результатов, и чтобы не останавливаться на локальном экстремуме, эвристические методы обычно берут в расчет решения, которые не ведут к немедленному улучшению результатов.

Как правило, ГА состоят из четырех основных операторов: селекции, кроссинговера (скрещивания, репродукции), мутации, создания нового поколения. Цикл скрещивания и мутации с последующей оценкой приспособленности называется поколением. Поэтапно алгоритм процесса формирования нового поколения можно представить так:

Шаг 1. Создать начальную популяцию из K хромосом (особей). Для этого случайным образом генерируется конечный набор пробных решений $X^k = (x_1^k, x_2^k, ..., x_n^k)$, $x_i^k \in X$ (первое поколение), $k = 1, 2, ..., K$, K - размер популяции).

Шаг 2. Оценить степень приспособленности каждой особи (рассчитать значения целевой функции $f_k(x) = f_k(x_1^k, x_2^k, ..., x_n^k)$.

Шаг 3. Выход, если выполняется критерий останова, иначе на шаг 4.

Шаг 4. Выбрать K родителей из популяции при помощи метода селекции (вероятность выбора родителя должна зависеть от степени его приспособленности).

Шаг 5. Выбрать из родительского пула пару родителей для репродукции. При помощи оператора кроссинговера с вероятностью p_c получить потомка.

Шаг 6. Подвергнуть потомков мутации с вероятностью p_m.

Шаг 7. Повторяем шаги 5–6, пока не будет сгенерировано новое поколение популяции, содержащее K хромосом

Шаг 8. Оценить степень приспособленности каждой особи в новой популяции.

Шаг 9. Перейти к шагу 4, если количество поколений не превышает допустимого.

Типовой алгоритм представлен на рис. 2.2. Все эти операторы могут изменяться от задачи к задаче, поэтому существует множество модификаций ГА, применимых к разным условиям. Перед началом работы алгоритма определяются вероятность кроссинговера и мутации. Вероятность кроссинговера рекомендуется выбирать равной 80-90%, а вероятность мутации – 1-3%. Эти числа могут варьироваться в зависимости от решаемой задачи [20].

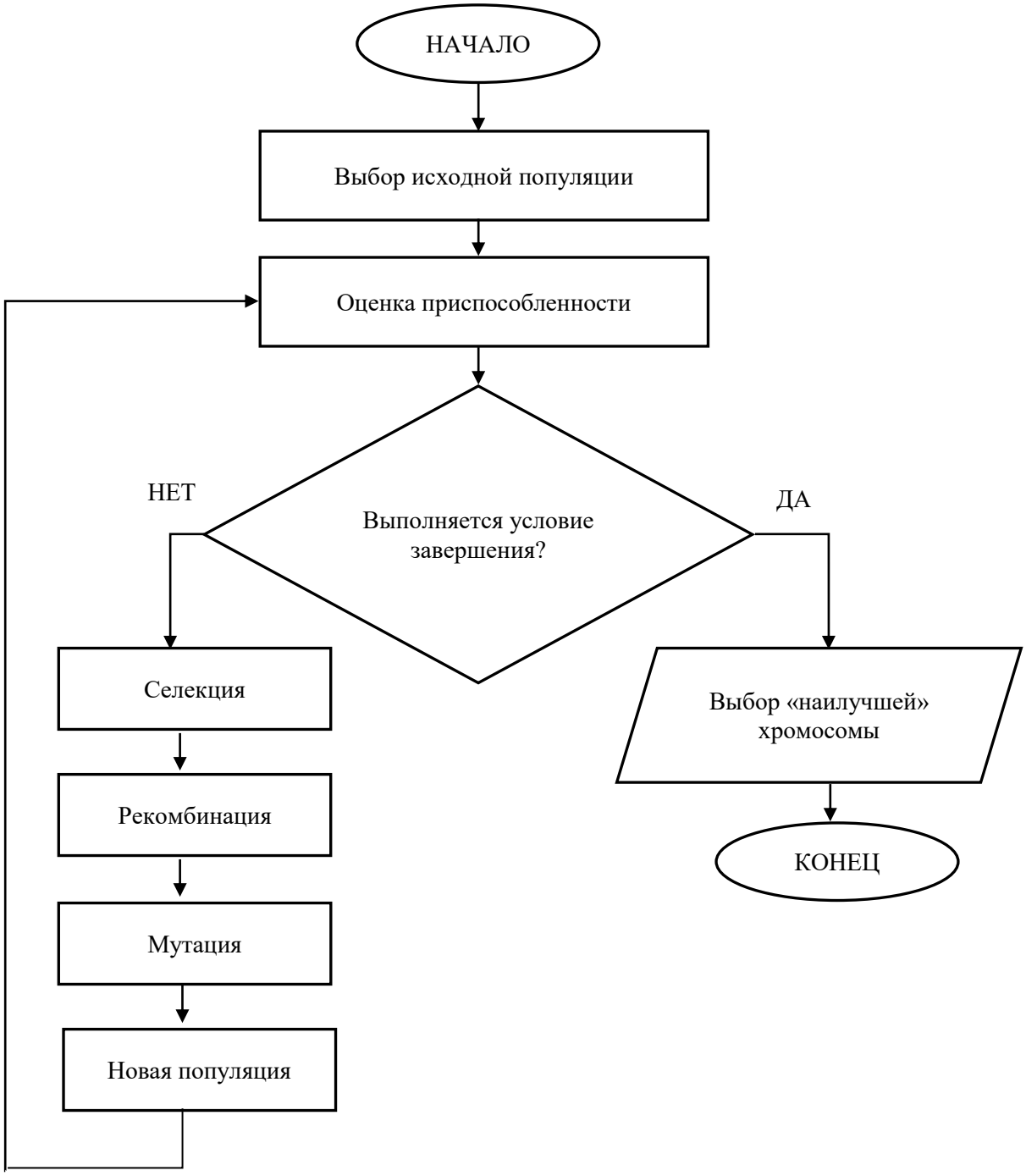


Рис. 2.2 – Блок-схема классического генетического алгоритма

Основные отличия ГА от других методов оптимизации:

- алгоритм начинается с некоторого набора начальных приближений, называемых начальной популяцией. Из-за этого у ГА меньше шансов остаться в ловушке локального экстремума;

- для ГА используется только целевая функция;

- применяются вероятностные методы.

Из-за того, что ГА основаны на принципе «выживает сильнейший», естественнее всего максимизировать целевую функцию, называемую функцией приспособленности. Поэтому ГА подходят для решения задач безусловной максимизации. Функция приспособленности, $F(x)$, может совпадать с целевой функцией $f(x)$ безусловной задачи максимизации $F(x) = f(x)$. Задача минимизации может быть легко трансформирована в задачу максимизации, обычно выбирается положительная функция приспособленности. Часто для перехода от задачи минимизации к задаче максимизации используется формула [17].

$$F(x) = \frac{1}{1 + f(x)} \tag{2.1}$$

Благодаря своей адаптивной природе и возможности определять операторы в зависимости от области применения, ГА используются для решения разнообразных задач [2-20]:

- поиска глобального экстремума многомерной функции;
- аппроксимации функций;
- создания искусственного интеллекта;
- поиска кратчайшего пути;
- составления расписаний.

2.1.2 Особенности терминологии

Ген – атомарный элемент хромосомы (может быть битом, числом или неким другим объектом.

Аллель – значение конкретного гена.

Локус – положение конкретного гена в хромосоме.

Хромосома (цепочка) – упорядоченная последовательность генов (строка из каких-либо чисел). Если эта строка представлена бинарной строкой из нулей и единиц, например, 101010, то она получена либо с использованием двоичного кодирования, либо кода Грея. Каждая позиция хромосомы называется *геном*.

Генотип(код) – упорядоченная последовательность хромосом (одно из решений).

Фенотип – набор значений, соответствующих генотипу.

Особь (индивидуум) – конкретный экземпляр генотипа (вариант решения задачи). Особи представляются хромосомами с закодированными в них множествами параметров задачи, т.е. решений. Обычно особь состоит из одной хромосомы, поэтому в дальнейшем особь и

хромосома идентичные понятия.

Генетические алгоритмы оперируют популяциями, т.е. конечным их числом особей.

Популяция – набор особей (набор решений задачи). В начале алгоритма случайным образом генерируется начальная популяция (набор решений). Эти решения будут становиться лучше (эволюционировать) в процессе работы алгоритма до тех пор, пока не удовлетворят условиям задачи.

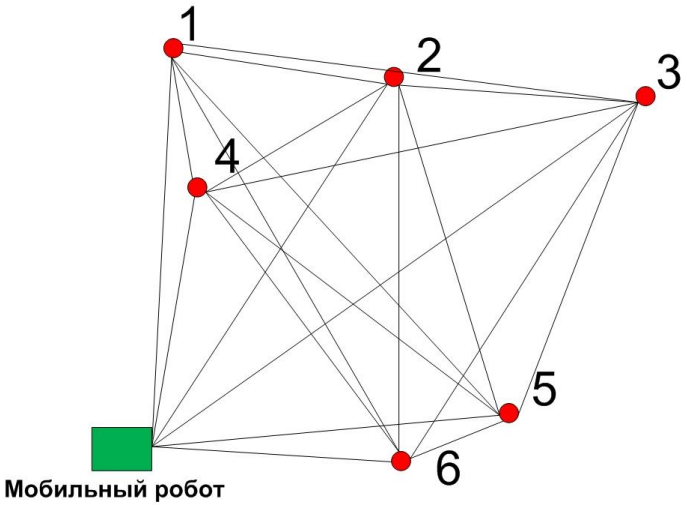
Кроссинговер (кроссовер) – операция, при которой две хромосомы обмениваются своими частями. Например, 1100&1010 → 1110&1000. Здесь произошел обмен вторым геном (вторым младшим разрядом).

Мутация – случайное изменение одной или нескольких позиций в хромосоме. Например, 1010011 →1010001.

Инверсия – изменение порядка следования битов в хромосоме или в ее фрагменте. Например, 1100 → 0011.

Рекомбинация – операция, при которой две хромосомы обмениваются своими частями.

◄ **Пример 2.1** Допустим, роботу необходимо объехать шесть контрольных точек за наименьшее время. Расстояние от каждой точки до каждой задано в виде матрицы расстояний.



	1	2	3	4	5	6
1		6	24	4	40	38
2	6		7	5	30	32
3	24	7		34	27	31
4	4	5	34		23	25
5	40	30	27	23		4
6	38	32	31	25	4	

Матрица расстояний

Рис. 2.3 Задача о поиске кратчайшего пути [22].

Возьмём несколько возможных решений (особей)– это и есть *начальная популяция*.

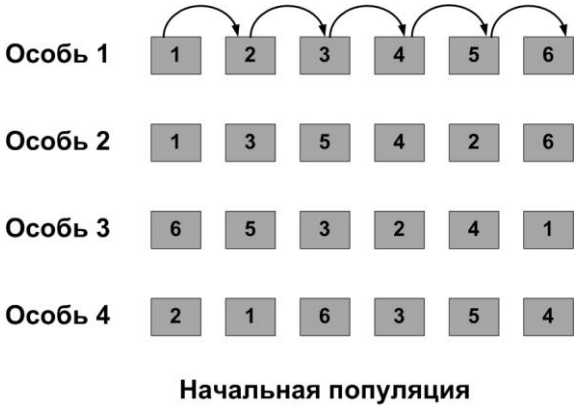


Рис. 2.4. Начальная популяция (начальные пробные решения)

Функция приспособленности. Одним из центральных понятий является *функция приспособленности* или *функция пригодности*. Она оценивает то, насколько приспособлена данная особь в популяции, другими словами, она определяет качество особей популяции.

В нашем примере это будет сумма расстояний от точки до точки в выбранном маршруте.

Рассчитаем функции пригодности. Для первой особи: $f_1(x) = 6 + 7 + 34 + 23 + 4 = 74$. Для остальных особей таким же образом получаем: $f_2(x) = 111$, $f_3(x) = 47$, $f_4 = 125$. Таким образом, особь №3 – лучшая, а №4 – самая плохая ►

Функции приспособленности всегда зависит от задачи и в её роли часто выступает целевая функция [24]. Иногда оценку приспособленности проводят в две стадии. Первая стадия – это собственно оценка $f_k(x) = f_k(x_1^k, x_2^k, ..., x_n^k)$. Вторая - дополнительные преобразования. Например, ею может быть нормировка к виду

$$F_k = \frac{f_k - f_k^0}{f_k^1 - f_k^0}, \tag{2.2}$$

где f_k^1, f_k^0 – соответственно, лучший и худший показатели в текущей популяции.

2.2 Кодирование параметров задачи

Успех в решении задачи с помощью ГА напрямую зависит от способа кодирования её параметров. Кодирование определяет то, каким образом будут выглядеть хромосомы. Хромосомы могут представлять из себя битовые строки, вещественные числа, перестановки элементов, список правил и практически любую структуру данных. В классическом ГА, представленном Холландом, производится бинарное кодирование параметров. Например, целые числа из интервала от 0 до 31 можно представить последовательностями нулей и

единиц, используя их представление в двоичной системе счисления. Число 0 при этом записывается как 00000, а число 31 – как 11111. В данном случае хромосомы приобретают вид двоичных последовательностей, состоящих из 5 битов, т.е. цепочками длиной 5.

При бинарном кодировании операции мутации и кроссинговера достаточно просты: мутация происходит инвертированием бита, а кроссинговер – обменом частями битовых строк. Если представлять целые числа двоичным кодом, то многие соседние числа, например, 15 и 16 (в двоичном представлении 01111 и 10000), различаются между собой несколькими разрядами, поэтому за одну операцию мутации из 15 нельзя получить 16 и наоборот. Эта проблема решается при помощи кода Грея, благодаря которому соседние числа, закодированные в бинарную строку, различаются только на один разряд.

Итак, будем считать, что каждая переменная x_i кодируется определенным фрагментом хромосомы, состоящим из фиксированного количества генов (см. рис. 2.5). Все локусы хромосом диаллельны — то есть в любой позиции фрагмента может стоять как ноль, так и единица. Рядом стоящие фрагменты не отделяют друг от друга какими-либо маркерами, тем не менее, при декодировании хромосомы в вектор переменных на протяжении всего моделируемого периода эволюции используется одна и та же маска картирования.

x_1				x_2				x_3				x_4				x_5				x_6			
1	1	0	0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	0	0	1	1	1	1

Рис. 2.5. Простейшая маска картирования хромосомы, определяющая план распределения наследственной информации по длине хромосомы

Хромосомы генерируются случайным образом, путем последовательного заполнения разрядов (генов), сразу в бинарном виде. Таблица 2.1 воспроизводит в полном объеме процедуру декодирования фрагмента 4-х битовой хромосомы в проекцию вектора переменных $x_i \in [a_i, b_i]$ [25].

Таблица 2.1

Декодирование фрагментов хромосом в проекции вектора переменных [25].

Десятичное значение сдвига	Двоично-десятичный код	Код Грея	Вещественное значение координаты
0	0000	0000	a
1	0001	0001	a+1(b-a)/15
2	0010	0011	a+2(b-a)/15

3	0011	0010	a+3(b-a)/15
4	0100	0110	a+4(b-a)/15
5	0101	0111	a+5(b-a)/15
6	0110	0101	a+6(b-a)/15
7	0111	0100	a+7(b-a)/15
8	1000	1100	a+8(b-a)/15
9	1001	1101	a+9(b-a)/15
10	1010	1111	a+10(b-a)/15
11	1011	1110	a+11(b-a)/15
12	1100	1010	a+12(b-a)/15
13	1101	1011	a+13(b-a)/15
14	1110	1001	a+14(b-a)/15
15	1111	1000	b

Аналогичную таблицу можно составить для 5-и битовых хромосом. В этом случае в четвертом столбце преобразование двоичной последовательности в вещественные числа будет выполняться по формуле $a + i \cdot (b - a)/31,\ i = 1,...,31.$

В общем случае число интервалов будет равно $2^L - 1$, где L – число битов (при L=4 получим $2^4 - 1 = 15$, при L=5 получим $2^5 - 1 = 31$).

Строка $c = (c_1,...,c_L),\ c_j \in \{0,1\},\ j = 1,...,L$, закодированная с помощью двоичного кода, представляет целое число $v \in \{0,...,2^L - 1\}$

$$v = \sum_{j=1}^{L} c_j 2^{(L-j)}. \tag{2.3}$$

Точки разбиения интервала $[a,b]$ при двоичном кодировании определяются по формуле

$$a + i \cdot (b - a)/(2^{L-1} - 1),\ i = 1,...,2^{L-1} - 1. \tag{2.4}$$

Преобразование строки $c = (c_1,...,c_L),\ c \in \{0,1\}$, закодированной двоичным кодом, в строку Грэя $g = (g_1,...,g_L)$ осуществляется по формуле [20]

$$g_k = \begin{cases} c_1, \ \textit{если}\ \ k = 1, \\ c_{k-1} \oplus c_k, \ \textit{если}\ \ k > 1, \end{cases} \tag{2.5}$$

где $\oplus$ – означает сложение по модулю 2. В случае бинарного и тернарного сложения по модулю 2 результаты приведены в таблице 2.2

Таблица 2.2 Сложение по модулю 2

Бинарное сложение				Тернарное сложение			
x	y	$a \oplus b$		x	y	z	$a \oplus b$
0	0	0		0	0	0	0
1	0	1		1	0	0	1
0	1	1		0	1	0	1
1	1	0		1	1	0	0
				0	0	1	1
				1	0	1	0
				0	1	1	0
				1	1	1	1

Общее правило сложения по модулю 2: результат равен 0, если нет операндов, равных 1, либо число операндов, равных 1 равно чётному количеству.

От кода Грея переходим к двоично-десятичному коду с помощью обратного преобразования по формуле

$$c_k = \sum_{i=1}^{k} (g_i \mid \otimes), \tag{2.6}$$

где суммирование выполняется по модулю 2. Далее, переходим к натуральным целым числам. Отношение полученного числа к максимальному числу, доступному для кодирования данным количеством разрядов фрагмента (в табл. 2.1 – число 15) и дает искомое значение сдвига переменной относительно левой границы a допустимого диапазона ее изменения ($b - a$).

Из таблицы хорошо видно, почему код Грея имеет явные преимущества по сравнению с двоично-десятичным кодом, который при некотором стечении обстоятельств порождает своеобразные тупики для поискового процесса. В качестве примера рассмотрим любые три рядом стоящие строки из таблицы 2.1, например, кодирующие сдвиг в 4, 5 и 6 единиц.

Предположим, фрагменты хромосом, стоящие в пятой строке и кодирующие число 5, принадлежат оптимальному вектору, являющемуся решением некоторой задачи, а лучшая особь из текущей популяции содержит фрагмент хромосомы из строки 4. Такая ситуация благоприятна для обоих кодов. Достаточно выполнить всего одну операцию — заменить в четвертом разряде фрагмента 0 на 1 – и решение будет найдено. Более интересный случай получается, если лучшая особь содержит фрагмент из строки 6. Для кода Грея эта ситуация ничуть не сложнее предыдущей – замена 0 на 1 в третьем разряде опять приведет к успеху. В

то же время двоично-десятичный код ставит нас в необходимость выполнить последовательно две операции — заменить 1 на 0 в третьем разряде и 0 на 1 в четвертом. С какой бы из них мы ни начали, результат не приблизит нас к решению (первый вариант замены переместит нас в четвертую строку, а второй — вообще в седьмую). А ведь это не самый худший пример — работать с сочетаниями 3-4, 7-8, 11-12 и т. д. строк в двоично-десятичном коде еще сложнее. Иначе говоря, если привлечь геометрические интерпретации, код Грея гарантирует, что две соседние, принадлежащие одному ребру, вершины гиперкуба, на котором осуществляется поиск, всегда декодируются в две ближайшие точки пространства вещественных чисел R^N, отстоящие друг от друга на одну дискрету точности. Двоично-десятичный код подобным свойством не обладает.

2.3 Оператор селекции

Оператор селекции предназначен для улучшения средней приспособленности популяции в новом поколении. В качестве оператора селекции часто используется метод рулетки (roulette-wheel selection). Название этот метод получил благодаря тому, что своим принципом выбора родительских особей напоминает игру в рулетку. Колесо рулетки делится на секторы, после запуска и остановки колеса стрелка с большей вероятностью укажет на сектор, площадь которого будет больше остальных. Хоть и нельзя предсказать, какой сектор будет в итоге выбран, можно найти вероятность выбора каждого сектора. В ГА рулетка делится на количество секторов равное размеру популяции, а площади секторов сопоставляются со значениями приспособленности каждой особи. Таким образом, отношение приспособленности одной особи к суммарной приспособленности популяции определяет площадь сектора соответствующей особи. Благодаря этому родительские особи выбираются с вероятностью, пропорциональной их функции приспособленности: чем «лучше» особь, тем больше у нее шансов размножиться. Вероятность выбора каждой особи вычисляется по формуле:

$$p_s(x) = \frac{f_s(x)}{\sum_{i=1}^{K} f_i(x)}\ , \tag{2.7}$$

где $f_i(x)$ – значение функции приспособленности i-й особи;

K – размер популяции.

Площадь сектора выражается в процентах от площади колеса по формуле:

$$v_s(x) = p_s(x) \cdot 100\%\ , \tag{2.8}$$

где $p_s(x)$ – вероятность выбора i-й особи.

При вращении рулетки K раз получают родительский пул размера K. Особи с высокой приспособленностью могут быть включены в родительский пул несколько раз, такая схема репродукции способствует «сильнейшим» жить и размножаться, а менее приспособленным – умирать.

Известны и другие способы отбора. Например, отбор может быть ранжированным: все особи ранжируются, т. е. упорядочиваются по приспособленностям, и заданная часть лучших особей (например, лучшая половина) отбирается для формирования следующего поколения.

◄ **Пример 2.2.** Найти максимум функции $f(x) = 2x^2 + 1$ для целочисленной переменной $x \in [0, 31]$. Закодируем значения переменной в виде двоичных последовательностей. Очевидно, что целые числа из интервала от 0 до 31 можно представить последовательностями нулей и единиц, используя их представление в двоичной системе счисления. Число 0 при этом записывается как 00000, а число 31 – как 11111. В данном случае хромосомы приобретают вид двоичных последовательностей, состоящих из 5 битов, т.е. цепочками длиной 5. В роли функции приспособленности будет выступать целевая функция $f(x) = 2x^2 + 1$. Тогда приспособленность хромосомы ch_i (i=1,…,N) будет определяться значением функции $f(x)$ для $x_i = ch_i$, равного фенотипу, соответствующему генотипу ch_i. Обозначим эти фенотипы x_i. В таком случае значение функции приспособленности хромосомы ch_i будет равно $f(x_i)$. Выберем случайным образом исходную популяцию, состоящую из 6 кодовых последовательностей:

$ch_1 = 10011,\ ch_2 = 00011,\ ch_3 = 00111,$
$ch_4 = 10101,\ ch_5 = 01000,\ ch_6 = 11101.$

Соответствующие им фенотипы – это представленные ниже числа:

$x_1 = 19,\ x_2 = 3,\ x_3 = 7,$
$x_4 = 21,\ x_5 = 8,\ x_6 = 29.$

Справка. Перевод числа из двоичной системы в десятичную выполняют по следующему алгоритму. Нумеруем разряды числа справа налево, начиная с нуля: И вычисляем результат:

$$111100110_2 = 1\cdot 2^8 + 1\cdot 2^7 + 1\cdot 2^6 + 1\cdot 2^5 + 0\cdot 2^4 +$$
$$+ 0\cdot 2^3 + 1\cdot 2^2 + 1\cdot 2^1 + 0\cdot 2^0 =$$
$$= 256 + 128 + 64 + 32 + 4 + 2 = 486_{10}$$

По формуле $f(x) = 2x^2 + 1$ рассчитываем значения функции приспособленности для каждой хромосомы в популяции:

$$f(x_1) = 723,\ f(x_2) = 19,\ f(x_3) = 99,$$
$$f(x_4) = 883,\ f(x_5) = 129,\ f(x_6) = 1683.$$

Среднее число приспособленности составляет $\overline{f}(x) \approx 589$.

Селекцию хромосом проводим **методом рулетки**. Для этого выбираем 6 хромосом для репродукции. Колесо рулетки представлено на рисунке. Размеры секторов пропорциональны числам $\sum_{k=1}^{i} f(x_k) / \sum_{k=1}^{6} f(x_k),\ i = 1,\ldots,6$. Площади секторов для каждой особи соответственно равны: 20,4%; 0,6%; 2,8%; 24,9%; 3,7%; 47,6%.

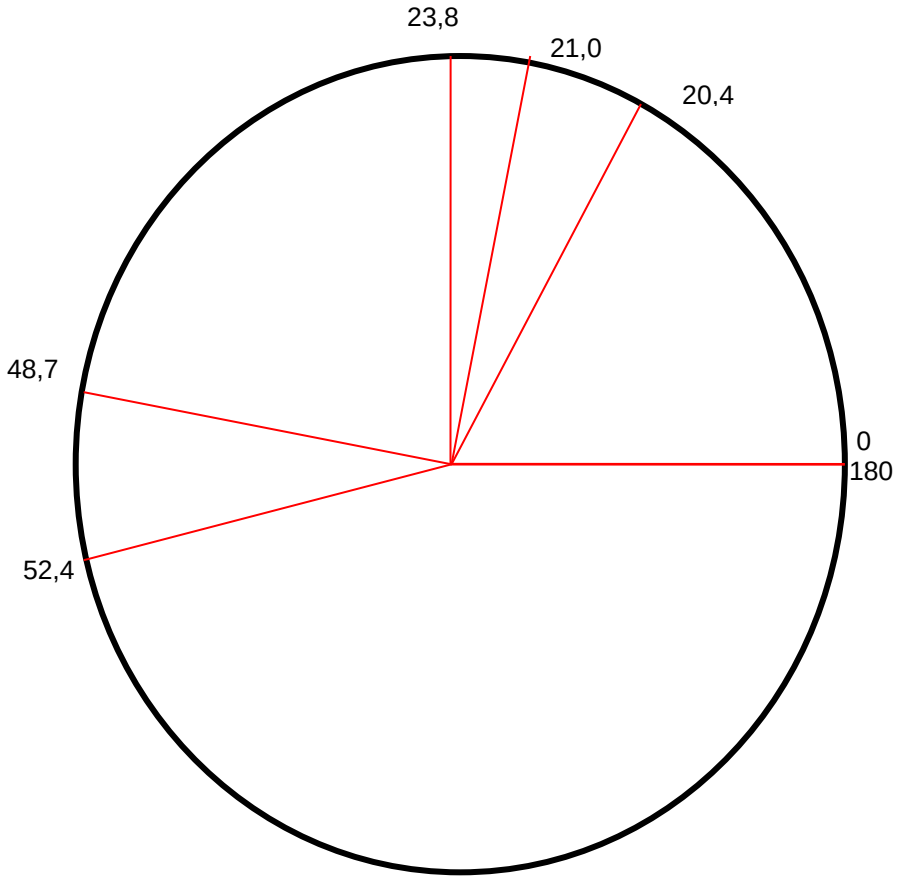


Рис. 2.6. Селекция хромосом методом рулетки

Пусть при розыгрыше случайных чисел из интервала от 0 до 100 выпали следующие числа: 97, 26, 54, 13, 31, 88. Это означает выбор хромосом ch_6, ch_4, ch_6, ch_1, ch_4, ch_6. Сформируем для скрещивания пары $(ch_1 - ch_6)$; $(ch_4 - ch_6)$; $(ch_6 - ch_6)$. Зададим случайным образом точку скрещивания, равной 4 для пары $(ch_1 - ch_6)$, а для пар $(ch_4 - ch_6)$; $(ch_6 - ch_6)$ – точку скрещивания, равной 2. Результаты скрещивания приведены в таблицах 2.3 – 2.5

Таблица 2.3.

	Родители							Потомки				
	1	2	3	**4**	5			1	2	3	**4**	5
ch_1	1	0	0	1	1	скрещивание	$ch1_1$	1	0	0	0	1
ch_6	1	1	1	0	1		$ch1_2$	1	1	1	1	1

Таблица 2.4.

	Родители							Потомки				
	1	**2**	3	4	5			1	**2**	3	4	5
ch_4	1	0	1	0	1	скрещивание	$ch1_3$	1	1	1	0	1
ch_6	1	1	1	0	1		$ch1_4$	1	0	1	0	1

Таблица 2.5.

	Родители							Потомки				
	1	**2**	3	4	5			1	**2**	3	4	5
ch_6	1	1	1	0	1	скрещивание	$ch1_5$	1	1	1	0	1
ch_6	1	1	1	0	1		$ch1_6$	1	1	1	0	1

Итак, в новую популяцию включаются хромосомы:

$$ch1_1 = 10001,\ ch1_2 = 11111,\ ch1_3 = 11101,$$
$$ch1_4 = 10101,\ ch1_5 = 11101,\ ch1_6 = 11101.$$

В результате декодирования получим соответствующие им фенотипы:

$$x1_1 = 17,\ x1_2 = 31,\ x1_3 = 29,$$
$$x1_4 = 21,\ x1_5 = 29,\ x1_6 = 29.$$

Соответственно, значения функции приспособленности хромосом новой популяции

$$f(x1_1) = 579,\ f(x1_2) = 1923,\ f(x1_3) = 1683,$$
$$f(x1_4) = 883,\ f(x1_5) = 1683,\ f(x1_6) = 1683.$$

Среднее значение функции приспособленности составляет $\overline{f}(x1) \approx 1406$, т.е.

возросло с 589 до 1406. При этом, для второго потомка получили максимальное значение функции приспособленности, равное 1923.►

Турнирный метод также является оператором селекции, отбирающим особей в зависимости от значения функции приспособленности. Главная идея заключается в том, чтобы выбрать особь с самой высокой приспособленностью из некоторого числа особей и позволить ей размножиться (записать в родительский пул). В этом методе из популяции, содержащей K особей, выбираются случайным образом t особей, и лучшая из них особь записывается в промежуточный массив (рис. 2.7). Эта операция повторяется K раз. Особи в полученном промежуточном массиве затем используются для скрещивания (также случайным образом). Размер группы строк, отбираемых для турнира, обычно равен 2 (t =2) или 3 (t =3). [5].

Достоинство этого метода состоит в том, что отсутствие дополнительных вычислений делает его эффективным и простым в реализации.

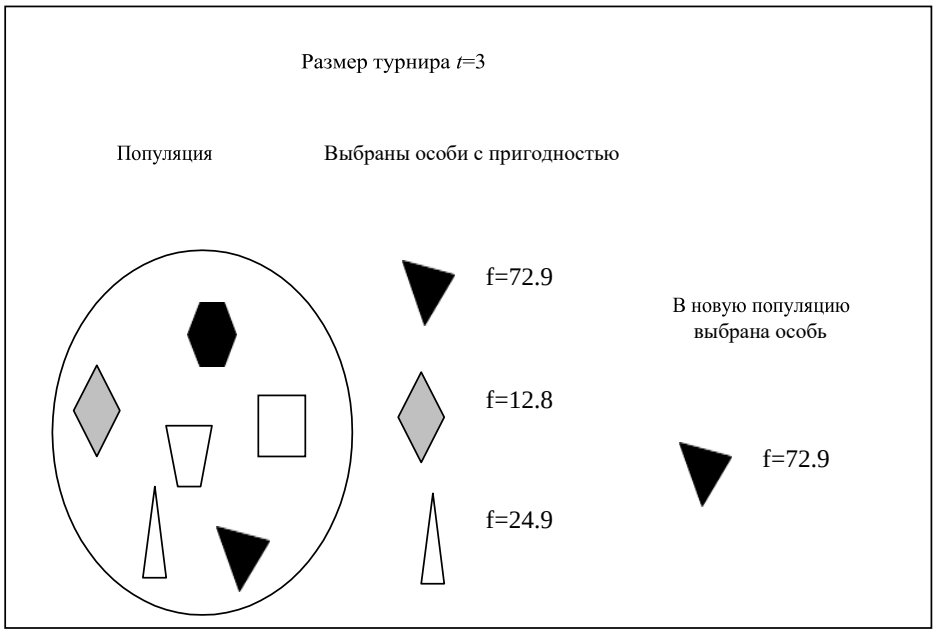


Рис. 2.7. Турнирный метод селекции [5].

2.4 Кроссинговер

Кроссинговер представляет собой обмен генетическим материалом внутри популяции. Причем подразумевается, что свой генетический материал будут в основном передавать особи с хорошей приспособленностью, потому что цель кроссинговера – улучшение среднего качества популяции. Проблема в том, что, когда особи найдут локальный экстремум, они будут стремиться занять всю популяцию. Процесс, когда все

особи начинают быть похожими друг на друга, называется вырождением популяции. Для того, чтобы противостоять алгоритму сходиться к локальному экстремуму используется оператор мутации.

Пусть i-я и j-я особи k-й популяции представлены как $p_i^k = (a_1, a_2, ..., a_n) \in P^k$ и $p_j^k = (b_1, b_2, ..., b_n) \in P^k$. В одноточечном варианте скрещивания возникнут два элемента популяции $k+1$: $p_i^{k+1} = (a_1, ..., a_c, b_{c+1}, ..., b_n) \in P^{k+1}$, $p_j^{k+1} = (b_1, ..., b_c, a_{c+1}, ..., a_n) \in P^{k+1}$, где точка c выбирается случайно.

В двухточечном варианте, соответственно, точек пересечения будет две, и они также выбираются случайно. Легко расширить эту конструкцию и до n точек. Нужно заметить, что в случае нечётного n, возникает $n+1$-точечный кроссинговер с $n+1$-ой точкой между последней и первой компонентами.

Скрещиванием с маской является результат в виде двух потомков с компонентами, принадлежность которых определяется по битовой маске. Т.е. результатом скрещивания родителей $p_i^k = (a_1, a_2, ..., a_n) \in P^k$, $p_j^k = (b_1, b_2, ..., b_n) \in P^k$ в k-ой популяции станут два элемента популяции $k+1$, такие что $p_i^{k+1} = (c_1, c_2, ..., c_n) \in P^{k+1}$, $p_j^{k+1} = (d_1, d_2, ..., d_n) \in P^{k+1}$, где $c_i = a_i$ при $m_i = 0$ и $c_i = b_i$ при $m_i \neq 0$, и противоположные условия для второго отпрыска. Маска выбирается случайно. Для простоты, ею может быть третья особь.

2.5 Оператор мутации

Мутации необходимы для того, чтобы поддерживать разнообразие особей в популяции и не позволять решению сходиться к локальному оптимуму. Мутации играют роль как в восстановлении утерянного генетического материала, так и в появлении у особей новых признаков. Традиционно оператор мутации рассматривают как простой оператор поиска, в то время как кроссинговер использует уже существующие хорошо приспособленные особи для создания потомков. Благодаря оператору мутации случайно изменяются части хромосом. Для каждого способа представления параметров задачи существуют различные операторы мутации. Вероятность возникновения мутации обычно принимают за $\frac{1}{L}$, где L – длина хромосомы. Также возможно осуществлять только те мутации, которые способны увеличить приспособленность особи. Такой вариант может

улучшить результаты поиска, но необходимо иметь в виду, что он также может уменьшить разнообразие в популяции и обеспечить сходимость к локальному оптимуму [19].

◄ **Пример 2.3.** Рассмотрим популяцию, полученную в примере 2.2

$ch1_1 = 10001,\ ch1_2 = 11111,\ ch1_3 = 11101,$

$ch1_4 = 10101,\ ch1_5 = 11101,\ ch1_6 = 11101.$

Примем вероятность мутации, равной $1/5 = 0,2$. Для каждого потомка возьмем случайное число на отрезке [0; 1], и если это число меньше 0,2, то инвертируем случайно выбранный ген (заменим 0 на 1 или наоборот) (см. табл. 2.5).

Таблица 2.6. Мутация потомков

№	Особи-потомки	Случайное число	Выбранный ген для мутации	Потомок после мутации	Приспособленность потомка до мутации	Приспособленность потомка после мутации
1	10001	0,6	1	10001	579	579
2	11111	0,1	5	11110	1923	1801
3	11101	0,3	1	11101	1683	1683
4	10101	0,04	2	11101	883	1683
5	11101	0,7	1	11101	1683	1683
6	11101	0,35	3	11101	1683	1683
	Среднее значение $f(x)$				1406	1519

Среднее значение функции приспособленности после мутации возросло и составило 1519. ►

2.6 Операторы отбора особей в новую популяцию

Классическим способом отбора в новую популяцию является замещение старой популяции своими потомками. В том случае, когда полученные потомки получились менее приспособленными, чем их родители, это становится не очень эффективно, поэтому существует стратегия элитарного отбора. Элитаризм бывает двух видов: конкурентный и неконкурентный. Первый представляет собой отбор лучших особей из родительского пула и пула потомков. Второй способ подразумевает переход лучших особей из старой популяции в новую, даже если их приспособленность хуже приспособленности потомков. Стратегия элитаризма позволяет не терять лучшие решения, найденные при мутациях [24].

2.7 Основные отличия генетических алгоритмов от традиционных методов поиска решений

1. Генетические алгоритмы работают с кодовыми строками, от которых зависят значения аргументов целевой функции и, соответственно, значение самой целевой функции. Интерпретация этих кодов выполняется только в операторе редукции (декодирование или переход к фенотипу). В остальном работа алгоритма не зависит от смысловой интерпретации кодов.

2. Для поиска лучшего решения генетический алгоритм на отдельном шаге использует сразу несколько точек поискового пространства (несколько вариантов решения задачи) одновременно, а не переходит от точки к точке, как это делается в традиционных методах. Это позволяет преодолеть один из их недостатков – опасность попадания в локальный экстремум целевой функции, если она не является унимодальной (т.е. имеет несколько экстремумов). Использование нескольких точек одновременно значительно снижает такую возможность.

3. Генетический алгоритм использует как вероятностные правила для порождения новых точек для анализа, так и детерминированные правила для перехода от одних точек к другим. Одновременное использование элементов случайности и детерминированности дает значительно больший эффект, чем раздельное [26].

2.8. Поиск минимума функции одной переменной [26]

Пусть требуется найти глобальный минимум функции

$$f(x) = 5 - 24x + 17x^2 - \frac{11}{3}x^3 + \frac{1}{4}x^4$$

на отрезке [0; 7] (рис. 2.8). На этом отрезке функция принимает минимальное значение в точке x = 1.

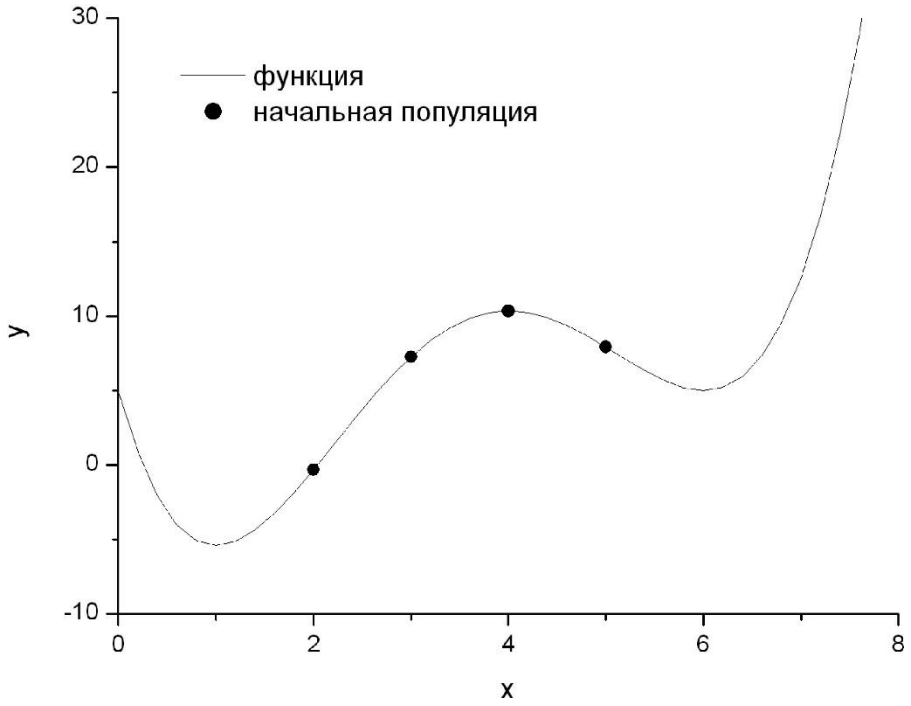


Рис. 2.8. Целевая функция с выбранными значениями пробных решений.

Очевидно, что в точке x = 6 функция попадает в локальный минимум. Если для нахождения глобального минимума использовать градиентные методы, то в зависимости от начального приближения можно попасть в данный локальный минимум.

Рассмотрим на примере данной задачи принцип работы генетических алгоритмов. Для простоты положим, что x принимает лишь целые значения, т.е. $x \in \{0, 1, 2, 3, 4, 5, 6, 7\}$. Это предположение существенно упростит изложение, сохранив все основные особенности работы генетического алгоритма.

Выберем случайным образом несколько чисел на отрезке [0; 7]: {2, 3, 5, 4}. Будем рассматривать эти числа в качестве пробных решений задачи (см. рис. 2.8).

Запишем пробные решения в двоичной форме: 010, 011, 101, 100 . Как известно, принцип естественного отбора заключается в том, что в конкурентной борьбе выживает наиболее приспособленный. В нашем случае приспособленность особи определяется целевой функцией: чем меньше значение целевой функции, тем более приспособленной является особь, т.е. пробное решение, использовавшееся в качестве аргумента целевой функции (см. табл. 2.7).

Таблица 2.7 Исходная популяция

№	Особи		
	Целое число	Двоичное число	Приспособленность
1	2	010	−0,33
2	3	011	7,25
3	5	101	7,92
4	4	100	10,33

Теперь приступим к процессу размножения: попробуем на основе исходной популяции создать новую, так чтобы пробные решения в новой популяции были бы ближе к искомому глобальному минимуму целевой функции. Для этого сформируем из исходной популяции брачные пары для скрещивания. Поставим в соответствие каждой особи исходной популяции случайное целое число из диапазона от 1 до 4. Будем рассматривать эти числа как номера членов популяции. При таком выборе какие-то из членов популяции не будут участвовать в процессе размножения, так как образуют пару сами с собой. Какие-то члены популяции примут участие в процессе размножения неоднократно с различными особями популяции. Процесс размножения (рекомбинация) заключается в обмене участками хромосом между родителями.

Для нашей популяции процесс создания первого поколения потомков показан в таблице 2.8.

Таблица 2.8. Одноточечный кроссинговер

№	Особь популяции	Выбранный номер	Вторая особь-родитель	Точка кроссинговера	Особи-потомки
1	010	1	010	1	010
2	011	4	100	1	000 111
3	101	3	101	2	101
4	100	3	101	2	100 101

Следующим шагом в работе генетического алгоритма являются мутации, т.е. случайные изменения полученных в результате скрещивания хромосом. Пусть вероятность

мутации равна 0,3. Для каждого потомка возьмем случайное число на отрезке [0; 1], и если это число меньше 0,3, то инвертируем случайно выбранный ген (заменим 0 на 1 или наоборот) (см. табл. 2.9).

Как видно на примере, мутации способны улучшить (первый и второй потомки) или ухудшить (четвертый потомок) приспособленность особи-потомка. В результате скрещивания хромосомы обмениваются «хвостами», т.е. младшими разрядами в двоичном представлении числа. В результате мутаций изменению может подвергнуться любой разряд, в том числе, старший.

Таблица 2.9 Мутация потомков

№	Особи-потомки	Случайное число	Выбранный ген для мутации	Потомок после мутации	Приспособленность потомка до мутации	Приспособленность потомка после мутации
1	000	0,1	3	001	5	−5,42
2	111	0,2	3	110	12,58	5
3	100	0,5	1	100	10,33	10,33
4	101	0,2	2	111	7,92	12,58

Таким образом, если скрещивание приводит к относительно небольшим изменениям пробных решений, то мутации могут привести к существенным изменениям значений пробных решений.

Теперь из четырех особей-родителей и четырех полученных особей потомков необходимо сформировать новую популяцию. В новую популяцию отберем четыре наиболее приспособленных особей из числа «старых» особей и особей-потомков (см. табл. 2.10).

В результате получим новое поколение, которое представлено на рис. 3. Получившуюся популяцию можно будет вновь подвергнуть кроссинговеру, мутации и отбору особей в новое поколение. Таким образом, через несколько поколений мы получим популяцию из похожих и наиболее приспособленных особей. Значение приспособленности наиболее «хорошей» особи (или средняя приспособленность по популяции) и будет являться

решением нашей задачи. Следуя этому, в данном случае, взяв наиболее приспособленную особь 001 во втором поколении, можно сказать, что минимумом целевой функции является значение 5,42, соответствующее аргументу x = 1. Тем самым попадания в локальный минимум удалось избежать! На данном примере разобран вариант простого генетического алгоритма. При дальнейшем использования ГА к разным задачам возможно моделирование основных операторов алгоритма.

Таблица 2.10 Формирование новой популяции из особей-родителей и особей-потомков

№	Особи	Приспособленность	Новая популяция	Приспособленность особей в новой популяции
1	010	−0,33	001	−5,42
2	011	7,25	010	−0,33
3	101	7,92	110	5
4	100	10,33	011	7,25
5	001	−5,42		
6	110	5		
7	100	10,33		
8	111	12,58		

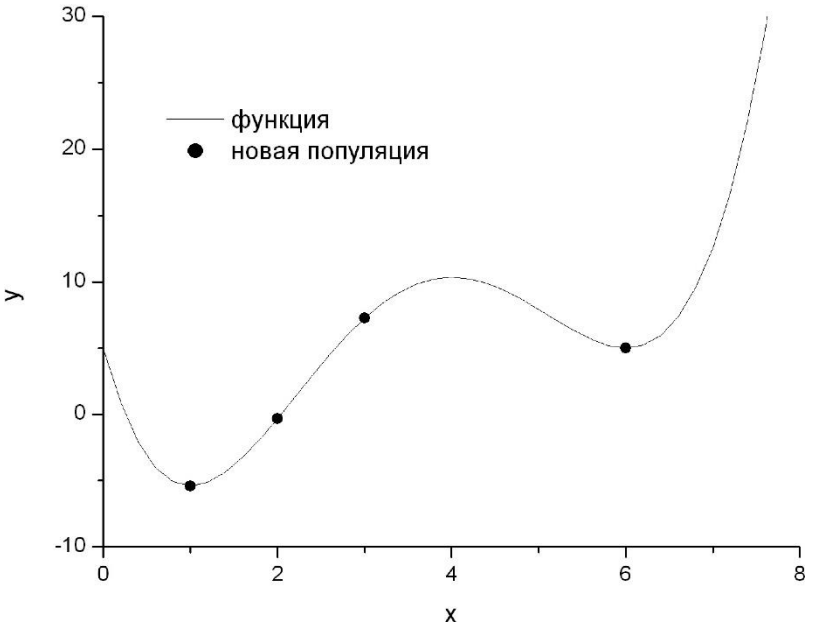


Рис. 2.9. Изменение популяции в процессе естественного отбора

Вопросы для самопроверки

1. Дайте определение генетическому алгоритму.
2. Что такое эвристические алгоритмы?
3. Дайте определение понятиям: «ген», «хромосома», «генотип», «особь» и «фенотип».
4. Дайте понятие функции приспособленности
5. Перечислите основные шаги алгоритма процесса формирования нового поколения.
6. Перечислите операторы генетического алгоритма и их назначение.
7. Опишите операторы селекции.
8. Как вычислить вероятность каждой особи в методе селекции с помощью рулетки?
9. Какие операторы скрещивания вам известны?
10. Перечислите известные вам операторы мутации.
11. Какие виды кодирования вам известны?
12. Как декодировать строку, записанную в двоичном коде?
13. Как декодировать строку, записанную в коде Грэя?
14. Как связаны вещественные числа с числами в бинарном коде?
15. Какие из операторов генетического алгоритма выполняются с использованием элементов случайности, а какие по строго детерминированным правилами?
16. Перечислите основные отличия генетических алгоритмов от традиционных методов поиска решений.
17. Опишите схему работы генетического алгоритма.
18. Что может являться критерием остановки работы генетического алгоритма?

Литература к теме 2

1 Holland J.H. Adaptation in Natural and Artificial Systems. – The University of Michigan Press, University of Michigan, Ann Arbor, 1975.

2 Редько В.Г. Эволюционная кибернетика. – М.: Наука, 2001. – 159 с.

3 Стецюра Г.Г. Эволюционные методы в задачах управления, выбора, оптимизации // Приборы и системы управления. – 1998. – № 3. – С. 54-62.

4 Курейчик В.М. Генетические алгоритмы. Состояние, проблемы, перспективы // Известия академии наук. Теория и системы управления. – 1999. – № 1. – С. 144-160.

5 Гудман Э.Д. Эволюционные вычисления и генетические алгоритмы // Обозрение прикладной и промышленной математики. – 1996. – Т. 3, вып. 5.

6 Курейчик В.М. Генетические алгоритмы. Обзор и состояние// Новости искусственного интеллекта. – 1998. – № 3.

7 Курейчик В.М., Родзин С.И. Эволюционные алгоритмы: генетическое программирование // Известия академии наук. Теория и системы управления. – 2002. – № 1. – С. 127-137.

8 Куприянов М.С., Матвиенко Н.И. Генетические алгоритмы и их реализации в системах реального времени // Информационные технологии. – 2001. – № 1. – С. 17-21.

9 Подлазова А.В. Генетические алгоритмы на примерах решения задач раскроя// Проблемы управления. –2008, №2. – С.57-63.

10 Липницкий А.А. Применение генетических алгоритмов к задаче о размещении прямоугольников // Кибернетика и системный анализ. – 2002. – № 6. – С. 180—184.

11 Мухачева А.С., Чиглинцев А.В. Генетический алгоритм поиска минимума в задачах гильотинного раскроя // Информационные технологии. – 2001. – № 3. – С. 27-31.

12 Мухачева Э.А., Мухачева А.С., Чиглинцев А.В. Генетический алгоритм блочной структуры в задачах двумерной упаковки // Информационные технологии. – 1999. – № 11. – С. 13-18.

13 Росс Клемент Генетический алгоритм: почему они работают? Когда их применять? // Компьютерра. – 1999. – № 11. – С. 20-23.

14 Hopper H., Turton B.C.H. A review of the application of meta-heuristic algorithms to 2D strip packing problems // Artificial Intelligence Review. – 2001. – N 16. – P. 257-285.

15 Емельянов В.В., Курейчик В.М., Курейчик В.В. Теория и практика эволюционного моделирования. – М.: Физматлитт, 2003. – 432 с.

16 Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы – М.: Горячая линия - Телеком, 2004. – 452 с.

17 Rao S. S. Engineering optimization: theory and practice. 4th ed. New Jersey, Wiley, 2009. – 813 p.

18 Blickle L. Thiele, A Comparison of Selection Schemes used in Genetic Algorithms. TIK-report, Zurich, TIC-Report, Nr. 11, December 1995

19 Goldberg David E. Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, 1989. – 432pp.

20 Herrera F., Lozano M., J.L. Verdegay. Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis // Artificial Intelligence Review, vol. 12 Issue 4, Aug. 1, 1998. – P. 265- 319.

21 Bhunia A. K., Pal.P, Chattopadhyay S., Medya B. K. On Genetic Operators for Unconstrained Optimization Problems // Advanced Modeling and Optimization. – 2010, vol. 12. – Num 2. – P. 291-304.

22 Генетические алгоритмы или как учебник по биологии может помочь в функциональной оптимизации. [Электронный ресурс]. – Режим доступа: http://lazysmart.ru/iskusstvenny-j-intellekst/geneticheskie-algoritmy-ili-kak-uchebn/ (дата обращения 10.11.2023)

23 Панченко Т.В. Генетические алгоритмы: учебно-методическое пособие / под ред. Ю. Ю. Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с.

24 Вороновский Г.К. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергеев.– Х.: ОСНОВА, 1997.– 112 с.

25 Хабаров С. Интеллектуальные системы извлечения новых знаний. Эволюционные вычисления. [Электронный ресурс]. – Режим доступа: http://www.habarov.spb.ru/new_es/exp_sys/ai_10.pdf (дата обращения 10.11.2023)

ТЕМА 3

МНОГОМЕРНАЯ БЕЗУСЛОВНАЯ ОПТИМИЗАЦИЯ ПРИ ПОМОЩИ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

3.1. Постановка задачи

Рассмотрим задачу безусловной минимизации

$$f(x) \to \min \tag{3.1}$$

при условии, что нам известны границы изменения переменной $x: l \le x \le u$.

Вектор параметров целевой функции $f(x),\ x \in D$ имеет компоненты $x_1, x_2, ..., x_n$, которые представляют собой искомые переменные задачи. Для оптимизации многомерных функций удобно использовать непрерывные генетические алгоритмы, то есть те, которые используют вещественное кодирование. При таком способе кодирования хромосомы представляют собой вектор вещественных чисел. Размер хромосомы равен длине вектора решения задачи, таким образом, ген представляет собой компоненту вектора решения задачи. Значения генов принимаются из интервала, на которых определены соответствующие переменные $x_i,\ i = 1, ..., n$, поэтому генетические операторы должны учитывать этот момент.

Использование вещественных чисел позволяет использовать большие домены для переменных, что сложно осуществить при бинарном представлении. Еще одно преимущество в том, что при незначительных изменениях в переменных происходят небольшие изменения значения функции. Использование вещественного кодирования очень близко к естественной формулировке многих задач, что стирает разницу между генотипом и фенотипом. Из-за отсутствия этой разницы пропадает необходимость использовать дополнительные процессы кодирования и декодирования, что увеличивает скорость работы алгоритма [1]. Для вещественного кодирования были изобретены различные способы кроссинговера и мутации. Здесь будут рассмотрены самые распространенные из них, представленные в работах [1, 2].

Генетический алгоритм (ГА) для решения задачи многомерной оптимизации включает следующие компоненты:

- параметры ГА;
- формирование хромосом – возможные решения задачи;
- инициализация хромосом;
- оценка функции приспособленности;

- процесс селекция;
- генетические операторы – скрещивание и мутация.

К параметрам ГА относят следующие: размер популяции, максимальное число популяций, вероятность скрещивания, вероятность мутации.

3.2 Селекция в задаче многомерной оптимизации

Селекция является важным фактором в генетическом алгоритме. Этот процесс случайный и позволяет выбрать лучшие решения. Главная цель оператора селекции выделить решения, которые лучше средних и удалить решения хуже средних из популяции для следующей генерации. Это достигается путем решения следующих задач:

- определение лучших решений в популяции;
- создание нескольких копий хороших решений;
- удаление плохих решений из популяции, так чтобы хорошие решения могли заменить их в популяции.

Имеется несколько схем селекции, такие как, рулетка, ранговая селекция, элитарная селекция, турнирный метод и другие.

Ранговая селекция. В этой схеме селекции порядок ранжирования индивидуумов относительно значений их функции пригодности определяет вероятность в текущей популяции. Популяция сортируется от лучших к худшим особям. Вероятность селекции каждой особи определяется в соответствии с рангом. Известны линейные и нелинейные ранговые методы. В линейном ранговом методе каждому индивидууму присваивается ранг в порядке убывания функции пригодности и вероятность i-го индивидуума в популяции вычисляется по формуле

$$p_i = \frac{F_i}{\sum_{j=1}^{K} F_j},\ F_i = \frac{1}{K}\left(f_{\max} - (f_{\max} - f_{\min})\frac{i-1}{K-1}\right), \qquad (3.2)$$

где K – размер популяции; $f_{\max}$, $f_{\min}$ – максимальное и минимальное значения целевой функции хромосомы (особи) в популяции.

Если значения целевой функции принимают отрицательные значения, то вероятности можно вычислять по формулам

$$p_i = \frac{1/(1+w_i)}{\sum_{j=1}^{K}\left(1/(1+w_j)\right)},\ i=1,...,K;\ w_i = \frac{f(x_i)}{\sum_{j=1}^{K} f(x_j)},\ i=1,...,K. \qquad (3.3)$$

В нелинейном ранговом методе вероятность i-го индивидуума определяется как

$$p_i = c\cdot(1-c)^{i-1},\ c\in[0,1]. \qquad (3.4)$$

Турнирная селекция. В этом методе селекции выбирается группа хромосом (индивидуумов)

из популяции случайным образом, затем наилучшие хромосомы в этой группе берутся в качестве родителей для создания следующего поколения с помощью генетических операций, таких как кроссинговер и мутация. Этот процесс будет повторяться до тех пор, пока не будет достигнута необходимая численность популяции хромосом. Параметром турнирного метода является размер турнира. Количество хромосом в турнире принимает значение от 2 до размера турнира.

Турнирный метод выбора основывается на следующих предположениях.

Для задачи безусловной оптимизации:

- Среди группы выбранных хромосом выбирается хромосома с лучшим значением пригодности.

Для задач условной оптимизации:

- Когда удовлетворяют ограничениям задачи все хромосомы (индивидуумы), выбирается индивидуум с лучшим значением функции пригодности.
- Когда одна хромосома (индивидуальная) удовлетворяют ограничениям задачи, а другие – не удовлетворяет, выбирается первая.
- Когда все хромосомы (индивидуумы) не удовлетворяют ограничениям- неравенствам задачи, тогда выбирается хромосома с меньшим нарушением ограничения.
- Если все хромосомы (индивидуумы) недопустимы в задаче с ограничениями-равенствами, то выбирается любая хромосома.

3.3 Кроссинговер в задаче многомерной оптимизации

После селекции выбранные хромосомы принимают участие в скрещивании для получения лучших потомков с целью улучшения популяции. Эта операция проводится на двух или более родительских хромосомах (решениях) одновременно и создает потомство из родительских генов. В этой операции число хромосом, которое будет принимать участие в скрещивании, равно $p_{kros} \cdot K$. Здесь p_{cros} – вероятность скрещивания, K – размер популяции, [·] – целая часть числа.

Процесс улучшения решения состоит из:

- выбора двух или более случайных хромосом из текущей популяции;
- создание потомков путем оператора скрещивания;

При описании оператора кроссинговера для обозначения родительских особей, выбранных на этапе селекции, приняты следующие обозначения:

$$P_1(x^1) = (x_1^1, x_2^1, ..., x_n^1),$$
$$P_2(x^2) = (x_1^2, x_2^2, ..., x_n^2) \quad (3.5)$$

где n – размерность задачи.

Для обозначения потомков особей P_1 и P_2 приняты обозначения:

$$C_1(y^1) = (y_1^1, y_2^1, ..., y_n^1),$$
$$C_2(y^2) = (y_1^2, y_2^2, ..., y_n^2). \quad (3.6)$$

Рассмотрим различные операторы скрещивания.

Простой арифметический кроссинговер. Пусть на t-ой генерации для скрещивания случайным образом выбраны две родительские особи

$$P_1^t(x^1) = (x_1^1, x_2^1, ..., x_n^1),$$
$$P_2^t(x^2) = (x_1^2, x_2^2, ..., x_n^2)$$

Тогда потомки

$$C_1^{t+1}(y^1) = (y_1^1, y_2^1, ..., y_n^1),$$
$$C_2^{t+1}(y^2) = (y_1^2, y_2^2, ..., y_n^2)$$

образуются следующим образом

$$y_i^1 = a \cdot x_i^1 + (1-a) \cdot x_i^2,\ i = 1, ..., n;$$
$$y_i^2 = a \cdot x_i^2 + (1-a) \cdot x_i^1.\ i = 1, ..., n. \quad (3.7)$$

В качестве величины a можно взять

$$a = f_{\max} / (f_{\max} + f_{\min}),$$

где $f_{\max} = \max_{1 \le j \le K}(f_j)$; $f_{\min} = \min_{1 \le j \le K}(f_j)$, f_j – значение целевой функции j-го решения.

Здесь оператор скрещивания использует простой статический параметр $a \in [0, 1]$. Если a является константой, то кроссинговер называют *равномерным арифметическим кроссинговером*. В противном случае его называют *неравномерным арифметическим кроссинговером.*

Кроссинговер смешивания. Создается случайный потомок в пределах гиперпрямоугольника, определяемом точками родителей. Пусть выбраны родительские хромосомы x_i^1 и x_i^2. Тогда i-я хромосома потомка в однопараметрическом операторе смешения определяется случайно из интервала $\left[x_i^1 - \alpha \cdot D_i,\ x_i^2 + \alpha \cdot D_i \right]$, где $D_i = x_i^2 - x_i^1$ при этом предполагается $x_i^1 < x_i^2$. Таким образом, результирующие потомки определяются

следующим образом

$$y_i^1 = (1-\mu_i)x_i^1 + \mu_i x_i^2;\ y_i^2 = (1-\mu_i)x_i^2 + \mu_i x_i^1, \tag{3.8}$$

где $\mu_i = (1+2\alpha)r_i - \alpha$, r_i – случайное число из интервала [0,1]. Наибольший эффект кроссинговер смешивания дает при $\alpha = 0,5$. При $\alpha = 0$ оператор смешивания дает случайное решение из интервала $[x_i^1, x_i^2]$.

В двухпараметрическом операторе смешения потомок создается путем выбора случайного решения из интервала $\left[x_i^1 - \alpha \cdot D_i,\ x_i^2 + \beta \cdot D_i\right]$, где α и β два положительных вещественных числа.

Кроссинговер Лапласа. Оператор скрещивания основан на распределении вероятностей Лапласа. Плотность $G(x)$ и функция распределения $F(x)$ вероятностей Лапласа определяются формулами

$$G(x) = \frac{1}{2b}\exp\left(-\frac{|x-a|}{b}\right),\ -\infty < x < \infty,$$

$$F(x) = \begin{cases} \frac{1}{2}\exp\left(\frac{(x-a)}{b}\right),\ x \le a \\ 1 - \frac{1}{2}\exp\left(\frac{(x-a)}{b}\right),\ x > a \end{cases},$$

Здесь $a \in R,\ b > 0$ – параметры распределения.

Случайно выбранные две родительские хромосомы

$$P_1(x^1) = (x_1^1, x_2^1, ..., x_n^1),$$
$$P_2(x^2) = (x_1^2, x_2^2, ..., x_n^2)$$

производят два потомка

$$C_1(y^1) = (y_1^1, y_2^1, ..., y_n^1),$$
$$C_2(y^2) = (y_1^2, y_2^2, ..., y_n^2)$$

с компонентами $y_i^1 = x_i^1 + \alpha \cdot \left|x_i^1 - x_i^2\right|,\ y_i^2 = x_i^2 + \alpha \cdot \left|x_i^1 - x_i^2\right|,\ i = 1, ..., n$. Здесь α – случайное число из распределения Лапласа, которое можно вычислить с помощью равномерного датчика по формуле $\alpha = a + b\ln(2 \cdot r)$, где r – случайное число из интервала (0,1].

3.4. Мутация в задаче многомерной оптимизации

Основная задача оператора мутации - ввести генетическое разнообразие популяции. Этот оператор используется для улучшения тонкой настройки системы. Это реализуется в хромосоме с наименьшей вероятностью (плохой хромосоме). Общий алгоритм мутации состоит в следующем:

- 1. Рассчитать целую часть числа $p_m \cdot K$, где p_m – вероятность мутации, K – размер популяции.
- 2. Случайно выбрать хромосому из популяции и затем случайно выбрать ген этой хромосомы.
- 3. Произвести новый ген, соответствующий выбранному гену, посредством операции мутации.
- 4. Повторить шаги (1) - (3) *K* раз.

Рассмотрим некоторые часто используемые операции мутации, такие как неравномерная мутация и экспоненциальная мутация.

Неравномерная мутация. Пусть выбрана хромосома $P(x) = (x_1, x_2, ..., x_n)$ и $x_k \in [l_k, u_k]$, l_k, u_k – нижняя и верхняя границы интервала соответственно для компоненты x_k. Мутация на t-й итерации происходит по следующим формулам:

$$y_k = \begin{cases} x_k + \Delta(t, u_k - x_k), \ \textit{если} \ r_k \le 0,5, \\ x_k - \Delta(t, x_k - l_k), \ \textit{если} \ r_k > 0,5. \end{cases} \tag{3.9}$$

Здесь r_k – случайное число из интервала [0,1], $\Delta(t, z)$ – функция, определяемая одной из формул

$$\Delta(t, z) = z \cdot \left(1 - r^{(1-t/T)^b}\right) \tag{3.10}$$

или

$$\Delta(t, z) = z \cdot r \cdot \left(1 - (1 - t/T)^b\right), \tag{3.11}$$

где t – номер текущего поколения, T – максимальное число поколений; b – системный параметр, определяющий степень неравномерности; r – случайное число из интервала [0,1].

Экспоненциальная мутация. Экспоненциальная мутация или мутация MPT была предложена для решения оптимизационных задач в аэродинамике. Мутация k-го гена происходит следующим образом:

$$y_k = (1 - g) \cdot l_k + g \cdot u_k, \tag{3.12}$$

где

$$g = \begin{cases} g - g \cdot \left(\dfrac{g - r}{g} \right)^b, & \text{если } r < g, \\ g, & \text{если } r = g, \\ g + (1 - g) \cdot \left(\dfrac{r - g}{1 - g} \right)^b, & \text{если } r > g. \end{cases} \qquad (3.13)$$

Здесь $g = \dfrac{x_k - l_k}{u_k - x_k}$; r – случайное число из интервала [0,1].

Вопросы для самопроверки

1. Какие компоненты включает вещественный генетический алгоритм (ГА) для решения задачи многомерной оптимизации?
2. С помощью какой операции оставляют в популяции решения выше средних и удаляют решения хуже средних из популяции и какие задачи для этого надо решить?
3. Как рассчитывается вероятность особи в линейном и нелинейном ранговом методе?
4. Перечислите предположения, на которых основан турнирный метод выбора решений в задаче безусловной оптимизации.
5. Перечислите предположения, на которых основан турнирный метод выбора решений в задаче условной оптимизации.
6. Как происходит скрещивание с помощью простого арифметического кроссинговера?
7. Чем отличается равномерный арифметический кроссинговер от неравномерного?
8. Как происходит скрещивание с помощью кроссинговера Лапласа?
9. Перечислите основные шаги алгоритма мутации в вещественных генетических алгоритмах.
10. Как происходит неравномерная мутация в вещественных генетических алгоритмах?
11. Как происходит экспоненциальная мутация в вещественных генетических алгоритмах?
12. В чем заключается задача коммивояжера?
13. Что является оптимальным решением задачи коммивояжера?
14. Приведите детерминированный алгоритм решения задачи коммивояжера.
15. Что является в задаче коммивояжера геном, генотипом и особью?
16. Как осуществляется генерация начальной популяции в задаче коммивояжера?
17. Опишите операторы скрещивания в задаче коммивояжера.
18. Опишите операторы мутации в решении задачи коммивояжера
19. Каковы особенности применения генетических алгоритмов формирование системы прогнозирующих правил в деятельности страховых компаний

Литература к теме 3

1. Herrera F., Lozano M., J.L. Verdegay. Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis // Artificial Intelligence Review, vol. 12 Issue 4, Aug. 1, 1998. – P. 265- 319.

2. Bhunia A. K., Pal.P, Chattopadhyay S., Medya B. K. On Genetic Operators for Unconstrained Optimization Problems // Advanced Modeling and Optimization. – 2010, vol. 12. – Num 2. – P. 291-304.
3. Otman Abdoun, Jaafar Abouchabaka, Chakir Tajani. Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem // International Journal of Emerging Sciences. – 2012, vol. 2. Num 1.–P. 61-77
4. Larrañaga P., Kuijpers C.M.H., Murga R.H. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. // Artificial Intelligence Review 13. – Kluwer Academic Publishers: Printed in the Netherlands. –1999. – P. 129-170.
5. Примеры практического применения генетических алгоритмов. [Электронный ресурс]. – Режим доступа: https://studopedia.org/11-55215.html (дата обращения 10.11.2023)
6. Социальная экономика [Электронный ресурс]. – Режим доступа: https://utmagazine.ru/posts/9255-socialnaya-ekonomika. (дата обращения 10.11.2023)
7. Белобородова Н.А. Модели и методы прогнозирования развития муниципального образования в нефтегазовом районе России // Нефтегазовое дело, 2010, №2. – с. 77–92.
8. Горбунов М.А, Медведев А.В., Семенкин Е.С. Применение генетических алгоритмов для решения задачи оценки эффективности региональной политики промышленности // Вестник Сибирского государственного университета им. акад. М.Ф. Решетнева, 2011, № 4. – С. 8–12.
9. Zitzler E., Thiele L. Muptiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach // IEEE Trans. on Evolutionary Computation. – San Diego, 1999. – pp. 257–271.
10. Примеры практического применения генетических алгоритмов. [Электронный ресурс]. – Режим доступа: https://studopedia.org/11-55215.html (дата обращения 10.11.2023)

Тема 4.
ПРИМЕРЫ ЗАДАЧ, РЕШАЕМЫХ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

4.1 Задача коммивояжера

4.1.1 Постановка задачи [1,2]

Задача заключается в том, что коммивояжер должен посетить каждый город на некоторой территории ровно один раз, а затем вернуться к исходной точке. Учитывая стоимость (расстояния, время поездки) проезда между всеми городами, необходимо так спланировать свой маршрут, чтобы стоимость (длина пути всего тура, время поездки) была минимальна. Поиск маршрута состоит в наборе перестановок из n городов. Оптимальным решением является перестановка, которая дает минимальную стоимость тура. Размерность пространства поиска равна (n-1)! для асимметричной задачи и (n-1)!/2 – для симметричной задачи. Другими словами, размерность тура N определяется множеством точек $v=\{v_1,v_2,...,v_n\}$, где v_i – i-й город, заданный координатами x_i, y_i. Длина пути одного из маршрутов (одной из перестановок) определяется формулой

$$f=\sum_{i=1}^{n-1}\sqrt{(x_i-x_{i+1})^2+(y_i-y_{i+1})^2}+\sqrt{(x_n-x_1)^2+(y_n-y_1)^2}\,. \qquad (4.1)$$

Решением задачи является способ планирования $T=\big(T[1],T[2],...,T[n],T[1]\big)$, где $T[i]$ является перестановкой на множестве {1, 2,..., N }.

Введем матрицу расстояний между городами v_i и v_j

$$d(i,j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}\,, \qquad (4.2)$$

тогда формула (4.1) примет вид

$$f(T)=\sum_{i=1}^{n-1}d(T[i],T[i+1])+d(T[n],T[1])\,. \qquad (4.3)$$

Задача коммивояжера заключается в минимизации функции $f(T)$ по аргументу T, где $T=\big(T[1],T[2],...,T[n],T[1]\big)$.

Детерминированные алгоритмы

Задача коммивояжера может быть редуцирована к задаче целочисленного линейного программирования. Пусть города пронумерованы числами от 1 до n. Для каждой пары городов зададим расстояние (или стоимость переезда) c_{ij}, $i,j=1,...,n$. Введем бинарные

переменные x_{ij}

$$x_{ij} = \begin{cases} 1, \ \textit{если осуществлен переезд из пункта } i \textit{ в пункт } j \\ 0, \ \textit{иначе} \end{cases} \tag{4.4}$$

Тогда имеем следующую оптимизационную задачу. Найти переменные x_{ij}, $i, j = 1,...n$, обращающие в минимум функцию

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{i-1} c_{ij} x_{ij} \tag{4.5}$$

и удовлетворяющие ограничениям

$$\begin{aligned} &\forall i \in N = \{1,2,...,n)\} \quad \sum_{j \neq i}^{n-1} x_{ij} = 2, \\ &\textit{for each } S \subset N \quad \sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 2. \end{aligned} \tag{4.6}$$

Первое ограничение означает, что каждая вершина должна сообщаться через пару ребер с остальными вершинам, то есть, через входное и выходное ребро. Второе ограничение связано с устранением подмаршрутов, т.е. каждое множество вершин $S \subset N$ является либо пустым, либо содержит все вершины, сочетающееся с остальными вершинам через минимум два ребра.

Для решения задачи можно использовать точные методы, такие как комбинаторный метод ветвей и границ и метод отсекающих плоскостей (метод Гомори). Однако с ростом размерности задачи экспоненциально возрастает требуемый объем памяти и времени счета. Поэтому для задач большой размерности точные методы не используются.

В течение последних десятилетий появилось несколько приближенных алгоритмов для аппроксимации оптимального решения: метод ближайшего соседа, жадный алгоритм, метод включения ближайшего города, метод имитации отжига и др.

4.1.2. Решение задачи коммивояжера с помощью генетического алгоритма

Задача коммивояжера относится к категории NP-полных задач, т.е. задач решаемых методом полного перебора всех вариантов.

Формализация задачи

Ген – это число, характеризующее номер посещаемого города.

Генотип – строка из чисел длиной N, описывающая порядок посещения городов.

Особь – конкретная строка из чисел (допустимый вариант решения задачи).

Представление тура

Существуют различные представления тура: бинарное представление, представление путей и представление смежности. В данной работе будет рассмотрено представление путей. Представление путей представляет собой упорядоченный список городов, которые необходимо посетить, причем если город i является j-ым элементом списка, то город i необходимо посетить j-м по счету. Например, тур 3-2-4-1-7-5-8-6 представляется просто как (3 2 4 1 7 5 8 6) [2].

Генерация начальной популяции

Начальная популяция обусловливает скорость и сходимость алгоритма.

Для этого можно применить несколько методов для генерации начальной совокупности:

- Случайная генерация начальной популяции.
- Генерация первой особи случайным образом, затем она будет мутирована N-1 раз с помощью оператора мутации.
- Генерация первой особи с использованием эвристического механизма. Преемник первого города расположен на расстоянии, меньшем по сравнению с другими. Затем мы используем оператор мутации на маршруте, полученном для того, чтобы сгенерировать (N-2) других индивидуумов, которые будут составлять начальную популяцию.

Операторы скрещивания

Одними из удачных операторов кроссинговера для решения ЗК признаны [1,2] Partially-Mapped Crossover (PMX) и Order Crossover (OX), или, как их ещё называют, частично соответствующий кроссинговер и упорядоченный кроссинговер соответственно. Операторы PMX и OX представляет собой обмен частей туров двух родительских особей. Для начала случайно выбираются две секущие точки для выделения частей туров из родительских особей (так называемый двухточечный кроссинговер). При таком обмене часто могут возникать частичные туры, например, тур $T = (5\ 7\ 1\ 2\ 5\ 4\ 3\ 6)$ содержит частичный тур: коммивояжер, выйдя из города 5, вернется в город 5, не посетив города 4, 3 и 6. Для того, чтобы не возникало частичных туров, в этих операторах существуют

механизмы создания правильных туров. Алгоритм создания потомков с помощью оператора PMX представлен ниже и проиллюстрирован на рисунках 4.1 и 4.2.

Шаг 1. Случайно выбрать две секущие точки для выделения части тура из родителя 1.

Шаг 2. Скопировать все города из родителя 2 в потомка.

Шаг 3. Для каждого города C из передаваемой части тура родителя 1:

Шаг 3.1. Найти город C в потомке. На его место поставить тот город, который заменяется городом C.

Шаг 3.2. Копировать город C из родителя 1 в потомка на ту же позицию.

Шаг 4. Для создания второго потомка поменять местами родителей и перейти к шагу 1.

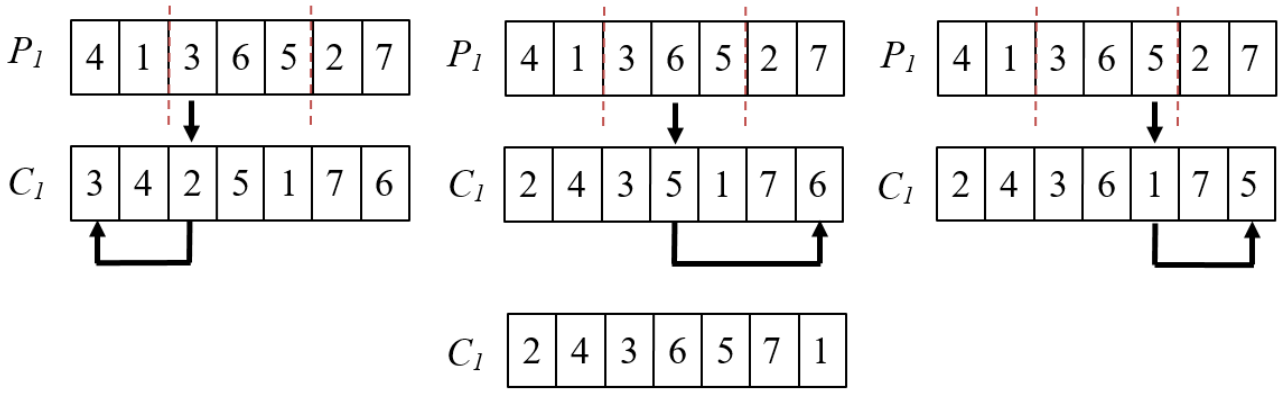


Рис. 4.1. Формирование первого потомка с помощью PMX

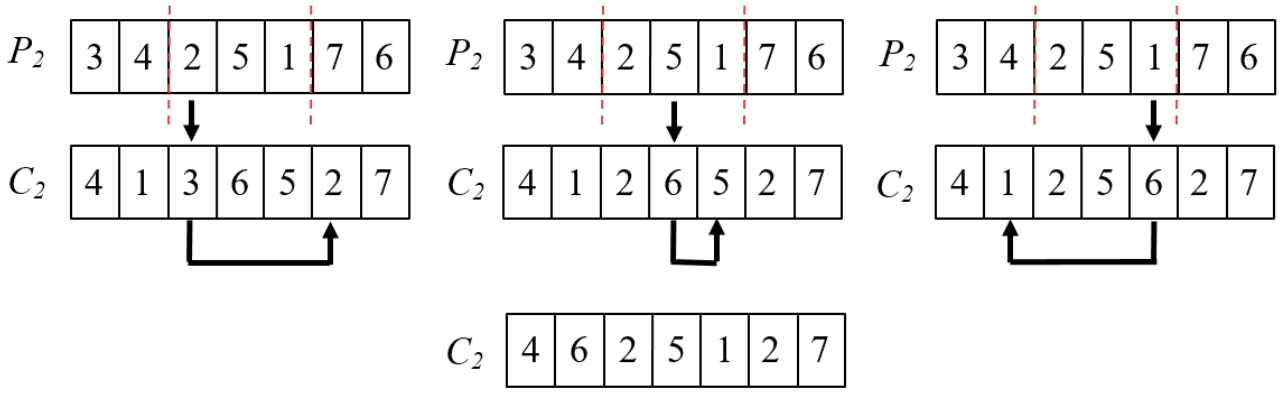


Рис. 4.2. Формирование второго потомка с помощью PMX

Особенность OX состоит в том, что он сохраняет порядок следования городов, а не их расположение в туре. Он строит потомка с использованием части тура одного из родителей и сохраняет порядок следования городов, представленный в другом родителе. Алгоритм создания потомков с помощью OX представлен ниже и проиллюстрирован на рисунке 4.3.

Шаг 1. Выбрать две секущие точки для выделения части тура из родителя 1.

Шаг 2: Выделить часть тура из родителя 1 и передать потомку. Все города из переданной части тура отметить как использованные в родителе 2.

Шаг 3: Начиная с правой стороны от переданной части тура, помещать неиспользованные города родителя 2 в потомок.

Шаг 4: Для создания второго потомка поменять местами родителей и перейти к шагу 1.

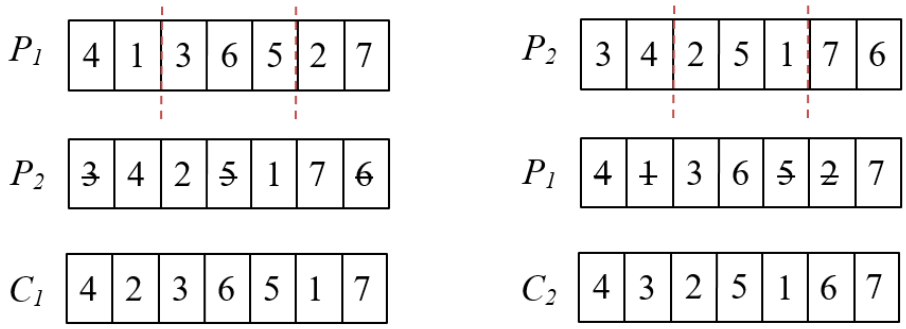


Рис. 4.3. Формирование потомков с помощью OX

Представленные выше операторы кроссинговера не используют информацию о расстояниях между городами. Эвристический кроссинговер – пример создания потомков с использованием информации о длине пути между городами. Алгоритм создания потомков с помощью эвристического кроссинговера представлен ниже.

Шаг 1. Выбрать начальным городом тура потомка любой из начальных городов туров родительских особей.

Шаг 2. Взять следующий город из родительских особей, оценить расстояние до них и передать потомку тот город, расстояние до которого будет меньше

Шаг 3. Если выбранный город образует частичный тур, то проверить город другого родителя, а если и он не создает правильный тур, то выбрать случайный город, не образующий цикл.

Шаг 4. Повторять шаги 2 и 3 пока все города не войдут в тур потомка.

Операторы мутации в решении задачи коммивояжера

В работах [1,2] предложены различные операторы мутации, ориентированные на решение задачи коммивояжера. Простая мутация перестановки (SIM) выбирает случайно два города в списке городов и переставляет их, т.е. меняет местами. Оператор перестановки представлен на рисунке 4.4.

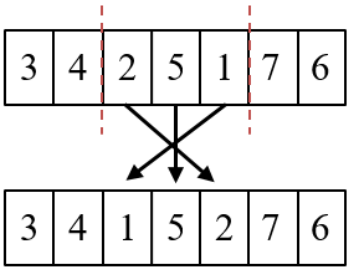


Рис. 4.4. Оператор мутации SIM

При мутации перестановки относительно центра (CIM) хромосома случайным образом разбивается на две части. В каждой части хромосомы гены перестанавливаются в обратном порядке. Оператор представлен на рисунке 4.5.

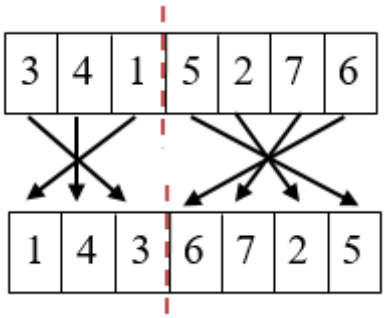


Рис. 4.5. Оператор мутации CIM

Мутация вставки (IM) случайно выбирает ген в хромосоме и переставляет его в другое случайно выбранное место. Оператор представлен на рисунке 4.6.

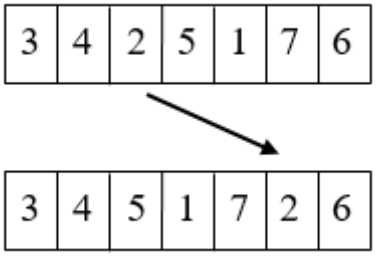


Рис. 4.6. Оператор мутации IM

Мутация перестановки (SM) случайно выбирает два гена и меняет их местами. Оператор представлен на рисунке 4.7.

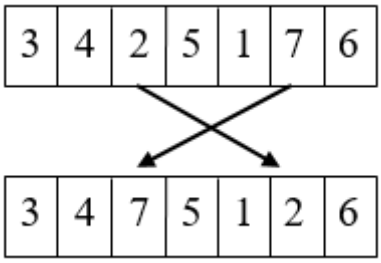
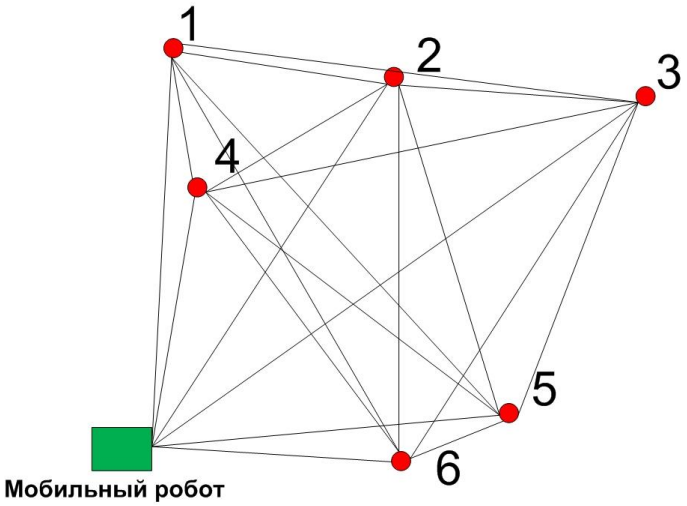


Рис. 4.7. Оператор мутации SM

Рассмотрим пример из темы 2 (пример 2.1.). Начальные решения приведены на рис 4.9. (см. рис. 2.3 и 2.4)

◄ **Пример 4.1** Допустим, роботу необходимо объехать шесть контрольных точек за наименьшее время. Расстояние от каждой точки до каждой задано в виде матрицы расстояний.



	1	2	3	4	5	6
1		6	24	4	40	38
2	6		7	5	30	32
3	24	7		34	27	31
4	4	5	34		23	25
5	40	30	27	23		4
6	38	32	31	25	4	

Матрица расстояний

Рис. 4.8 Задача о поиске кратчайшего пути

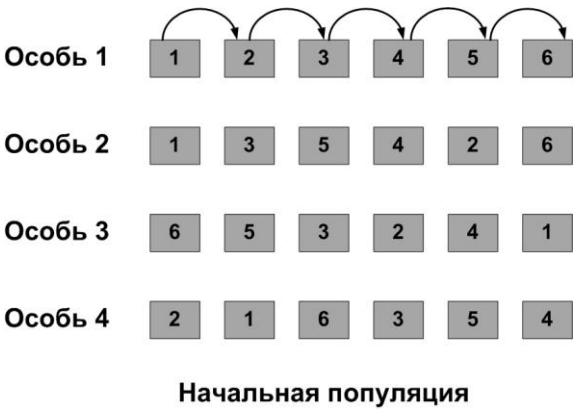


Рис. 4.9. Начальная популяция (начальные пробные решения)

Выпишем их в таблицу 4.1.

Таблица 4.1 Начальные решения

№ решения							Длина пути	Вероятность выбора
1	1	2	3	4	5	6	74	0,26
2	1	3	5	4	2	6	111	0,23
3	6	5	3	2	4	1	47	0,29
4	2	1	6	3	5	4	125	0,22

Далее используем рулетку для отбора лучших решений. Рассчитаем вероятности выбора каждого решения по формуле

$$p_s(x) = \left(1 - \frac{f_s(x)}{\sum_{i=1}^{K} f_i(x)}\right) \Big/ \sum_{s=1}^{K} \left(1 - \frac{f_s(x)}{\sum_{i=1}^{K} f_i(x)}\right). \qquad (4.7)$$

Здесь K – количество начальных решений (размер начальной популяции).

Результаты занесем в таблицу 4.1.

С помощью равномерного датчика генерируем 4 случайных числа из интервала [0,1]. Пусть эти числа будут равны 0,75; 0,5; 0, 20; 0,15. Тогда пул родителей составят номера решений: 3, 3, 1, 1. Для скрещивания выберем 1-е и 3-е решения, так как 3-е с 3-м и 1-е с 1-м не приведут к улучшению решений.

Для скрещивания используем оператор PMX. Выберем две секущие точки, например, 2 и 4 и сформируем потомки (см. табл. 4.2)

Таблица 4.2 Кроссинговер PMX.

1 родитель							2 родитель					
1	2	3	4	5	6		6	5	3	2	4	1
1 потомок							2 потомок					
6	5	3	2	4	1		1	2	3	4	5	6
			→						←			
6	5	3	4	2	1		1	4	3	2	5	6

Таким образом, мы получили два допустимых решения (два потомка):

(6 5 3 4 2 1) и (1 4 3 2 5 6). Длина пути для первого потомка составит 76, для второго – 79 (см. рис. 4.8).

Дальше применим оператор мутации SIM (простой перестановки) к каждому из потомков. Пусть это будут 3-й и 5-й гены для 1-го потомка и 1-й и 6-й – для 2-го потомка.

Таблица 4.3 Оператор мутации SIM

1 потомок до мутации							2 потомок до мутации					
6	5	3	4	2	1		1	4	3	2	5	6
1 потомок после мутации							2 потомок после мутации					
6	5	2	4	3	1		6	4	3	2	5	1

Длина пути для первого потомка после мутации составит 97, для второго – 136 (см. рис. 4.8). Таким образом, мутация ухудшила решения.

4.2. Использование генетических алгоритмов для решения социально экономических задач

Социальная экономика – это отрасль научного знания, деятельность и сфера экономики, основной целью которой является достижение личностного развития, а также способствование прогрессу человечества [4].

Как правило, с социальной экономикой связывают дополнительно два понятия:

1. Социальное предпринимательство – использование стартапов и других средств предпринимательства для разработки, финансирования и реализации решений социальных, культурных или экологических проблем [4].

2. Социальное инвестирование – практика распределения ресурсов с целью достижения положительного социального эффекта. Социальное инвестирование может относиться как к разделу отказа от вложения в определенные виды активов (алкогольная, табачная, оружейная продукция), так и к инвестициям в развитие социально-полезных активов, регионов или внутрикорпоративное развитие (расход на повышение квалификации сотрудников). О последних типах далее и пойдет речь [4].

Генетические алгоритмы позволяют осуществлять поиск оптимальных решений для целого ряда практических задач. Также заметим, что широта сферы применения ГА обусловлена, прежде всего, универсальностью, а также способностью одновременно оптимизировать решение задачи по нескольким критериям [4].

Анализ результатов использования ГА позволяет выделить следующие условия, при

выполнении которых задача решается эффективно:

1) большое пространство поиска, ландшафт которого является негладким (содержит несколько экстремумов);

2) сложно формализуемая функция степени оценки качества решения;

3) многокритериальность поиска;

4) поиск по заданным критериям приемлемого, а не единственного оптимального решения.

4.2.1 Применение генетических алгоритмов для решения задачи прогнозирования развития города

Задача – спрогнозировать показатели, соответствующие развитию города на основе независимых переменных, отражающих производство этого города.

В данном примере будет рассмотрен город Уфа. Пусть имеется некий набор параметров (табл. 4.1). Согласно постановке задачи Y_1, Y_2, Y_3 – показатели, характеризующие рост доходов в бюджет города, сокращение безработицы, рост заработной платы. Показатель Y_1 представлен в денежном выражении; Y_2 – число зарегистрированных безработных, чел.; Y_3 – представлен также в денежном выражении. Данные показатели рассматриваются как зависимые переменные от тех переменных, которые обозначены через X_i и характеризуют производство основных видов продукции в натуральном выражении.

Таблица 4.1 – Зависимые и независимые показатели города

Показатель	Наименование переменной
Доход в бюджет	Y_1
Безработица	Y_2
Заработная плата	Y_3
Добыча нефти	X_1
Производство автомобильного бензина	X_2
Производство дизельного топлива	X_3
Первичная переработка нефти	X_4
Производство мазута	X_5

Производство пиломатериалов	X_6
Производство стеновых материалов	X_7
Производство электроэнергии	X_8
Производство теплоэнергии	X_9
Производство мяса и мясопродуктов АПК	X_{10}
Производство хлеба и хлебобулочных изделий АПК	X_{11}
Производство молока и молочных продуктов АПК	X_{12}
Производство скота и птицы (сельское хозяйство)	X_{13}
Производство молока (сельское хозяйство)	X_{14}
Капитальные вложения в производственную сферу	X_{15}
Производство услуг	X_{16}

Решение задачи проводится в несколько этапов:

Этап 1 – предобработка данных. Аналитическое выявление тех показателей X_i, от которых зависит Y_i (например, в периоде 2005-2006 гг. показатель Y_1 представлен переменными X_7, X_8, X_{13}, X_{16}) и составление целевой функции φ.

Этап 2 – прогнозирование с помощью ГА. Использование генетического алгоритма для выявления наилучшего показателя Y_i. Алгоритм работы представлен на рис. 4.10.

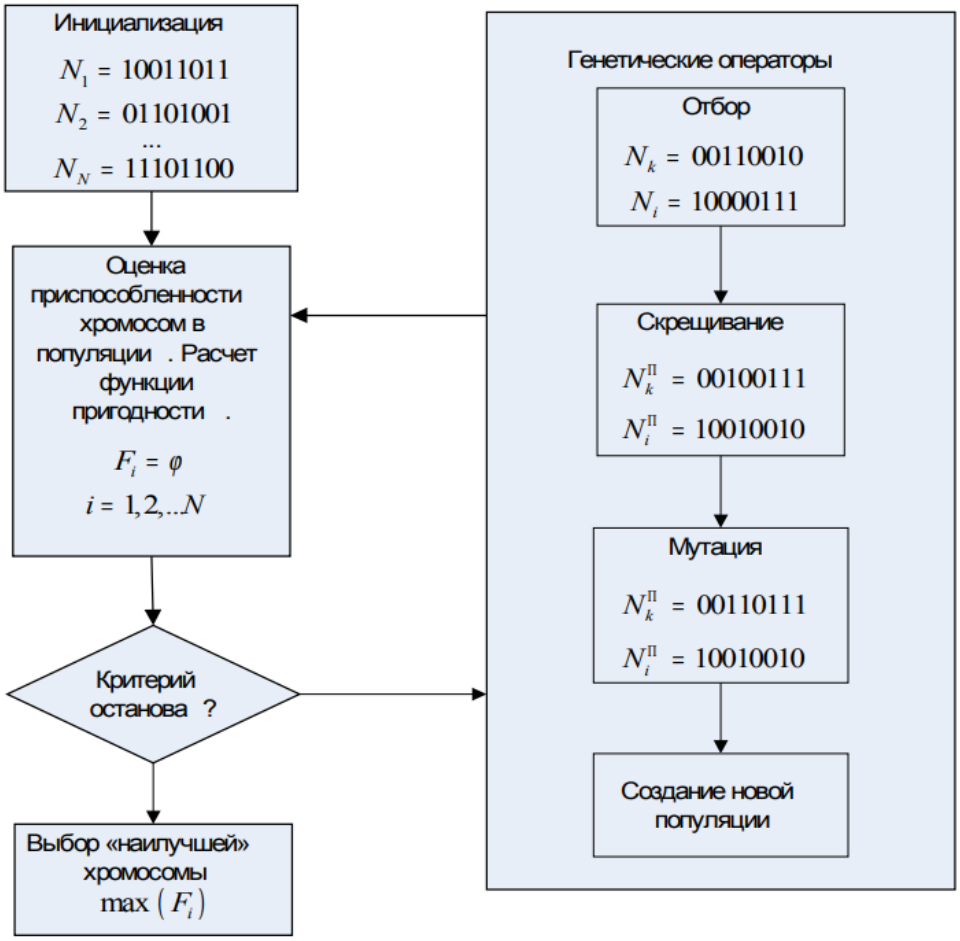


Рис. 4.10 – Алгоритм работы ГА в подборе наилучших параметров X_i

Кодирование переменных выполняется так: пусть длина каждого гена 16 бит, и для целевой функции перебираются параметры X_1 и X_2, тогда получим хромосомную нить из 32 нулей и единиц, и тогда N равен:

$$\underbrace{1000101101110110}_{X_1}\underbrace{0011000111111011}_{X_2}$$

Вероятность отбора родителя пропорциональная функции пригодности:

$$P_i^{отбора} = \frac{F_i}{\sum_{i=1}^{n} F_i}.$$

Далее работает оператор кроссинговера в случайной точке, например:

$$N_1 = 1000111011001001010.010110001101$$
$$N_2 = 0111001010100011101.111111000110$$

Скрещенные особи:

$$N_1^{\Pi} = 1000111011001001010.111111000110$$
$$N_2^{\Pi} = 0111001010100011101.010110001101$$

Мутация:

$$N_1^{\text{П}} = 1000111\boxed{0}111001001010.111111000110$$

Мутированная особь:

$$N_1^{\text{П}} = 1000111111001001010.111111000110$$

4.2.2. Применение генетических алгоритмов для решения задачи оценки эффективности региональной промышленной политики

Данная задача формулируется как задача социально-экономического развития региона. Пусть планирование экономической деятельности в виде совокупности инвестиционных проектов (ИП) в регионе осуществляется экономическими агентами, заинтересованными в эффективном функционировании региональной экономики. К таким агентам следует отнести производственный сектор, население региона, управляющий их взаимодействием региональный (налоговый) центр и потенциальный инвестор – физическое или юридическое лицо любой формы собственности, готовое вложить в развитие региона свои свободные денежные средства (капитал) [8].

Модель. Пусть имеется некая сложная модель с ограничениями, описывающая инвестиции:

$$J_{inv} = \frac{\sum_{k=1}^{n} \sigma_k x_k + \gamma \sum_{k=1}^{n} x_{n+k}}{1+r} - x_{2n+1} - x_{2n+2} - \varepsilon_2 x_{2n+4} \to \max,$$

$$J_{cons} = \frac{\beta \sum_{k=1}^{n} x_{n+k}}{1+r} - \varepsilon_1 x_{2n+3} \to \max,$$

$$J_{tax} = \frac{-\sum_{k=1}^{n} \sigma_k x_k + \left(\alpha_3 - \beta\left(\alpha_3 - \alpha_4\right)\right) \sum_{k=1}^{n} x_{n+k}}{1+r} -$$

$$-\left(1-\varepsilon_1\right) x_{2n+3} - \left(1-\varepsilon_2\right) x_{2n+4} \to \max,$$

$$\sum_{k=1}^{n} \gamma_k x_k + \gamma \sum_{k=1}^{n} x_{n+k} + x_{2n+1} + x_{2n+2} + x_{2n+4} \geq 0,$$

$$-\sum_{k=1}^{n} \theta_k x_k + \rho \sum_{k=1}^{n} x_{n+k} \geq 0,$$

$$\beta \sum_{k=1}^{n} x_{n+k} + x_{2n+1} + M_0 - \sum_{k=1}^{n} q_k \geq 0,$$

$$-\sum_{k=1}^{n}\sigma_k x_k + \left(\alpha_3 - \beta\left(\alpha_3 - \alpha_4\right)\right)\sum_{k=1}^{n} x_{n+k} +$$

$$+\left(1-\varepsilon_1\right)x_{2n+3} + \left(1-\varepsilon_2\right)x_{2n+4} + N_0 \geq 0,$$

$$x_{n+k} \leq q_k,\ x_{n+k} \leq \delta_k x_k\ \left(k = 1,...,n\right),\ x_{2n+k} \leq I_0,,$$

$$x_{2n+2} \leq K_0,\ x_{2n+3} + x_{2n+4} \leq L_0,\ x_k \geq 0\left(k = 1,...,2n+4\right),,$$

$$W = \left(1-\varepsilon_3\right)\left(1-\beta\right)R - \left(1-\varepsilon_3\right)\left(Am + N_2\right) =$$

$$= \left(1-\varepsilon_3\right)\left[-\sum_{k=1}^{n}\theta_k x_k + \left(1-\beta\right)\sum_{k=1}^{n} x_{n+k}\right] \geq 0.$$

Обозначения:

$J_{inv},\ J_{const},\ J_{tax}$ – критерии инвестора, потребителя и управляющего центра;

$\delta_k = P_k V_k / c_k$ – фондоотдача основных производственных фондов (ОПФ) k-го направления деятельности;

c_k – стоимость ОПФ k-го типа;

V_k – проектная производительность ОПФ k-го типа;

P_k – стоимость единицы продукции k-го типа;

$x_k = c_k m_k, k = 1,...,n$ –стоимость ОПФ, приобретаемых в ИП;

m_k – количество приобретаемых ОПФ k-го типа;

$x_{n+k} = P_k m_k y_k\ \ (k = 1,...,n)$ – выручка от реализации продукции k-того типа;

y_k – объем выпуска по k-му виду продукции;

$R = \sum_{k=1}^{n} P_k m_k x_k = \sum_{k=1}^{n} x_{n+k}$ – выручка от реализации по всем видам продукции;

q_k – прогнозный спрос на продукцию k-го типа;

Am – амортизационные отчисления;

$Am(T) = T\sum_{k=1}^{n}\frac{c_k m_k}{T_k} = \sum_{k=1}^{n}\frac{T}{T_k}x_k$ – сумма амортизационных отчислений по всем видам ОПФ;

T_k – срок службы ОПФ k-го типа;

T – срок действия ИП;

K_0 – начальный собственный капитал предприятия;

M_0 – начальный собственный средства потребителей;

N_0 – начальные собственные средства налогового центра;

$N_2 = \alpha_2 S^0$ – налог на имущество;

$S^0 = \sum_{k=1}^{n}\left(1-\frac{T}{T_k}\right)c_k m_k = \sum_{k=1}^{n}\left(1-\frac{T}{T_k}\right)x_k$ – остаточная стоимость ОПФ;

L_0 – максимальная сумма дотаций за весь период действия ИП;

I – внешние инвестиции;

$\overline{I}$ – внутренние инвестиции;

I_0 – максимальная сумма внешних инвестиций;

W – прибыль;

α_1, α_2, α_3, α_4 – ставки налогов на добавленную стоимость (НДС), на имущество (НИ), на прибыль (НП), на доходы физических лиц (НДФЛ);

ε_1, ε_2 – проценты возврата дотаций производителям и потребителям;

β – часть выручки, поступающая в фонд оплаты труда;

r – ставка дисконтирования, учитывающая инфляцию, требования инвестора по доходности и различные риски проекта;

$x_{2n+1} = I$ – внешние (возвратные) инвестиции (инвестора);

$x_{2n+2} = \overline{I}$ – внутренние инвестиции производителя;

$x_{2n+3} = Dot_1$ – дотации потребителям;

$x_{2n+4} = Dot_2$ – дотации производителям;

$$\gamma_k = \left(2\alpha_2 + \alpha_3 - \alpha_2\alpha_3\right)\frac{T}{T_k} + \alpha_2\alpha_3 - 2\alpha_2 - 1;$$

$$\sigma_k = \left(\alpha_2 + \alpha_3 - \alpha_2\alpha_3\right)\frac{T}{T_k} + \alpha_2\left(1-\alpha_3\right);$$

$$\theta_k = \left(1-\alpha_2\right)\frac{T}{T_k} - \alpha_2, k = 1,...,n;$$

$$\rho = 1-\alpha_1-\beta, \gamma = \left(1-\alpha_3\right)\rho;$$

$$\tau = \rho\alpha_3 + \beta\alpha_4.$$

Задача – найти такой общий объем инвестиций, объем инвестиций в приобретение ОПФ (основные производственные фонды) указанных видов (отраслей) экономической деятельности, а также планируемую выручку от реализации продукции, чтобы чистый приведенный доход (NPV) инвестиционного проекта по развитию указанных видов экономической деятельности в регионе был максимальным.

С использованием региональной экономической статистики были рассмотрены четыре отрасли экономической деятельности Красноярского края [8]:

1) добыча полезных ископаемых;
2) обрабатывающие производства;
3) производство и распределение электроэнергии, газа и воды;
4) строительство.

Данная задача была решена [9] симплекс-методом и ГА.

Таблица 4.5 – Результаты расчетов симплекс-методом и модифицированным алгоритмом SPEA для линейных ограничений

x	Симплекс-метод	ГА (SPEA)
x_1	72 848,31	73 007,44
x_2	84 774,68	85 331,24
x_3	129 242,2	135 649,19
x_4	57 115,08	58 170,4
x_5	28 046,6	28 016,63
x_6	438 878,5	438 763,51
x_7	26 623,9	27 934,7099
x_8	24 616,6	24 575,99
x_9	0	0
x_{10}	0	0
x_{11}	0	0
x_{12}	0	0
Значение критерия	326	326

	669,590 6	208,370 8

На линейных ограничениях ГА и симплекс-метод выдают примерно одинаковые результаты.

Симплекс-метод применим только в случае присутствия линейных ограничений. Если ограничение вида

$$x_{n+k} \le q_k, k = 1,...,n$$

заменяется на

$$x_{n+k} \le \frac{k_i}{1 + b_i e^{\alpha_i / (\beta_i x_{n+k})}},$$

то задача этим методом уже нерешаема, и необходимо применять другие методы, одним из которых и зарекомендовал себя генетический алгоритм SPEA [9].

4.2.3. Формирование системы прогнозирующих правил в деятельности страховых компаний [10]

Генетические алгоритмы могут использоваться для нахождения оптимального набора правил, позволяющих прогнозировать страховые риски с учетом ряда определяющих его факторов. Для решения этой задачи необходимо иметь базу данных, содержащую фактические значения переменных, влияющих на страховой риск.

Рассмотрим пример использования генетического алгоритма. Для оптимизации экспертных правил в сфере страхования. Допустим, что компания, занимающаяся страхованием автомобилей, использует базу данных, которая помимо прочих включает следующие факторы: максимальную скорость автомобиля (км/час), возраст автомобиля (лет), возраст водителя (лет) и риск, определенный экспертно по некоторой шкале на основе анализа обращений клиентов о выплате компенсации по страховым случаям. Правила, задающие оценку страхового риска, сконструированы в виде: ЕСЛИ максимальная скорость автомобиля лежит в диапазоне [Ai]И возрастной диапазон автомобиля [Bi] И возраст водителя находится в диапазоне [Ci], ТО страховой риск имеет значение [Di].

Для конкретной выборки из БД это правило может иметь следующий вид:

ЕСЛИ максимальная скорость [91 − 100 км/час] И возраст автомобиля [11 − 15 лет] И возраст водителя [31 − 40 лет], ТО риск [3]. Здесь уровень риска отображается на интервал

[1, 5], при этом высокие значения соответствуют большим страховым рискам.

Подобные правила, основанные на фактических значениях переменных, случайным образом выбранных из БД, составляют исходную популяцию. Для каждой из переменных, входящих в популяцию, предварительно задается диапазон состояний. Например, переменная «возраст автомобиля», может иметь пять возможных состояний: 1 − 5, 6 − 10, 11 − 15, 16 − 20, 21 − 25 лет. Далее сформированная популяция обрабатывается генетическими операторами с учетом специфики рассматриваемой задачи. Целевая функция должна показывать, насколько точно сгенерированные правила описывают реальные страховые случаи, хранящиеся в БД. Например, если какое-то правило описывает 4 случая из 5, то значение целевой функции будет 4/5, или 80%.

Новые члены популяции образуются в результате скрещивания и мутации начального набора правил. В данном случае при скрещивании двух правил происходит обмен парами «атрибут − значение» на участке строки после точки кроссинговера.

Справка. В базах данных **атрибут** представляет собой свойство сущности. Так, атрибутами сущности «Ассортимент товара» являются его «Код», «Модель», «Название», «Поставщик», «Стоимость» и другие.

В результате образуются два новых правила, жизнеспособность которых оценивается по тому, насколько удачно они описывают страховые случаи, которые имели место в прошлом. Мутация правил обеспечивает необходимое разнообразие признаков и заключается в изменении значений атрибутов с заданной вероятностью. Таким образом, первоначально сформированный набор правил преобразуется случайно направленным способом в другой набор, который лучше остальных описывает накопленную статистику страховых случаев. Результирующая система правил в дальнейшем используется для прогнозирования страховых рисков.

Следует отметить, что подобный подход к формированию системы правил может приводить к некорректным правилам продукций. В то же время он освобождает разработчиков и экспертов от трудоемкой работы по формулированию и оценке правил, так как некорректные результаты отбрасываются при сопоставлении сгенерированных продукций с реальными страховыми ситуациями. Привлечение прошлого опыта для оценки пригодности прогнозирующих правил не позволяет предвидеть новые ситуации, которые не имели места в прошлом. Поэтому при решении задач описанным способом очень важно следить за своевременным пополнением и модификацией информации в БД,

которая отражает появление новых фактов, атрибутов и тенденций.

Вопросы для самопроверки

1. В чем заключается задача коммивояжера?
2. Опишите детерминированный алгоритм решения задачи коммивояжёра.
3. Опишите генетический алгоритм решения задачи коммивояжёра.
4. Какие операторы скрещивания применяются в генетическом алгоритме?
5. Опишите алгоритм создания потомков с помощью оператора PMX.
6. Опишите алгоритм создания потомков с помощью оператора OX.
7. Опишите алгоритм создания потомков с помощью эвристического оператора скрещивания.
8. Какие операторы мутации используются в решении задачи коммивояжёра?
9. Что такое социальная экономика?
10. Перечислите условия, при которых социально-экономические задачи решаются эффективно.
11. Опишите алгоритм решения задачи прогнозирования развития города.
12. В каком случае оправдано применение генетического алгоритма для решения оценки эффективности региональной промышленной политики?
13. Каковы особенности применения генетических алгоритмов формирование системы прогнозирующих правил в деятельности страховых компаний?

Литература к теме 4

1. Otman Abdoun, Jaafar Abouchabaka, Chakir Tajani. Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem // International Journal of Emerging Sciences. – 2012, vol. 2. Num 1.–P. 61-77
2. Larrañaga P., Kuijpers C.M.H., Murga R.H. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. // Artificial Intelligence Review 13. – Kluwer Academic Publishers: Printed in the Netherlands. –1999. – P. 129-170.
3. Примеры практического применения генетических алгоритмов. [Электронный ресурс]. – Режим доступа: https://studopedia.org/11-55215.html (дата обращения 10.11.2023)
4. Белобородова Н.А. Модели и методы прогнозирования развития муниципального образования в нефтегазовом районе России // Нефтегазовое дело, 2010, №2. – с. 77–92.
5. Горбунов М.А, Медведев А.В., Семенкин Е.С. Применение генетических алгоритмов для решения задачи оценки эффективности региональной политики промышленности // Вестник Сибирского государственного университета им. акад. М.Ф. Решетнева, 2011, № 4. – С. 8–12.

6. Zitzler E., Thiele L. Muptiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach // IEEE Trans. on Evolutionary Computation. – San Diego, 1999. – pp. 257–271.