МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛЬ-ХОРЕЗМИЙ

ФАКУЛЬТЕТ ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
КАФЕДРА ПРОГРАММНОГО ИНЖИНИРИНГА

Направления программный инжиниринг (Программный инжиниринг)

Допустить к защите
Заведующий кафедрой
Утеулиев Н.У
2019 г. «___»_____

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему: «Решение задачи коммивояжера с помощью генетических алгоритмов в системе MatLab »

Выпускник: **Мамутова А.**
Научный руководитель: **проф.Утеулиев Н.У.**

НУКУС-2019

Оглавление

Введение

Генетические алгоритмы возникли в результате наблюдения и попыток копирования естественных процессов, происходящих в мире живых организмов, в частности, эволюции, связанной с ней селекции (естественного отбора) популяции живых существ. Идея генетических алгоритмов была высказана в конце шестидесятых - начале семидесятых годов XX века. Она была основана на желании составить и реализовать в виде компьютерной программы алгоритм, который будет решать сложные задачи так, как это делает природа - путем эволюции.

Современная библиография по генетическим алгоритмам давно перевалила за 9000 наименований и продолжает непрерывно увеличиваться. Однако, несмотря на такое обилие литературы, довольно трудно точно сформулировать, чем именно они являются - квинтэссенцией эволюционных перестроек в природных популяциях организмов, универсальным средством описания адаптаций в популяциях искусственных объектов, или мощной поисковой процедурой с претензиями на решение задач глобальной оптимизации.

Целью данной работы является изучение генетических алгоритмов как способа оптимизации, их эффективности и трудоемкости. В качестве решаемой задачи была выбрана задача коммивояжера, поскольку она очень хорошо изучена, имеет эффективные способы решения, для того, чтобы сравнить с полученными результатами. Также одной из целей данной работы является изучение распространения генетических алгоритмов на модель с несколькими взаимодействующими популяциями.

Основным инструментом для практического исследования была выбрана среда MATLAB, поскольку она имеет множество встроенных функций и панелей инструментов для решения задач генетического программирования и их параллельного выполнения.

1. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ И ТРАДИЦИОННЫЕ СПОСОБЫ ОПТИМИЗАЦИИ

1.1. Описание генетических алгоритмов

Согласованность и эффективность работы элементов биологических организмов наводит на мысль о возможности использования принципов биологической эволюции с целью оптимизации практически важных для человека систем.

В 1975 г. вышла основополагающая книга Дж. Холланда (Holland) “Адаптация в естественных и искусственных системах”, в которой был предложен генетический алгоритм - алгоритм, основанный на принципах естественного отбора Ч.Дарвина. Генетические алгоритмы относят к области мягких вычислений. Понятие «мягких вычислений» (Лофти Заде, 1994 г.) подразумевает под собой совокупность неточных, приближенных методов решения задач, зачастую не имеющих решение за полиномиальное время. Такие задачи возникают в области биологии, медицины, гуманитарных наук, менеджменте. Методы мягких вычислений хорошо дополняют друг друга, и часто используются совместно. В область мягких вычислений входят такие методы как:

- нечеткая логика;
- нейронные сети;
- вероятностные рассуждения;
- байесовские сети доверия[1];
- эволюционные алгоритмы. [5]

Генетический алгоритм представляет собой метод, отражающей естественную эволюцию методов решения проблем, и в первую очередь задач оптимизации. Генетические алгоритмы - это процедуры поиска, основанные на механизмах естественного отбора и наследования. В них используется эволюционный принцип выживания наиболее приспособленных особей. Они

отличаются от традиционных методов оптимизации несколькими базовыми элементами. В частности, генетические алгоритмы обладают рядом отличительных свойств:

1. кодирование параметров - генетические алгоритмы обрабатывают не значения параметров самой задачи, а их закодированную форму;

---

[1] Байесовские сети доверия - модель вероятностных и причинно-следственных отношений между переменными в статистическом информационном моделировании.

2. операции на популяции - генетические алгоритмы осуществляют поиск решения исходя не из единственной точки (начальное приближение), а из некоторой популяции;
3. использование минимума информации о функции - генетические алгоритмы используют только целевую функцию, а не производные либо иную дополнительную информацию;
4. рандомизация операций - генетические алгоритмы применяют вероятностные, а не детерминированные правила выбора.

Перечисленные свойства приводят в результате к устойчивости генетических алгоритмов. [6]

Сфера применения генетических алгоритмов - это в основном оптимизация многопараметрических функций. Прикладное же применение генетических алгоритмов весьма обширно. Они применяются при разработке программного обеспечения в системах искусственного интеллекта, оптимизации, искусственных нейронных сетях и в других отраслях знаний. Следует отметить, что с их помощью решаются задачи, для которых ранее

использовались нейронные сети. В этом случае генетические алгоритмы выступают просто в роли независимого от нейронных сетей метода, предназначенного для решения той же самой задачи. Примером может служить задача коммивояжера, изначально решавшаяся при помощи сети Хопфилда[1]. Генетические алгоритмы часто используются совместно с нейронными сетями. Они могут поддерживать нейронные сети или наоборот, либо оба метода взаимодействуют в рамках одной гибридной системы, предназначенной для решения конкретной задачи. Генетические алгоритмы так же применяются совместно с нечеткими системами.

Целесообразность использования генетических алгоритмов изложена очень емко и кратко во фразе из статьи К. Де Йонга:

«Решающий аргумент при использовании генетических алгоритмов тесно связан с вопросом о том, какое пространство поиска будет исследовано. Если это пространство легко анализировать и его структура позволяет использовать специализированные методы поиска, то использование генетических алгоритмов менее эффективно с точки зрения затрат вычислительных ресурсов. Если же пространство поиска не поддается анализу иотносительно слабо структурировано, и если возможен эффективный способ представления в генетических алгоритмах этого пространства, то они оказываются удивительно эффективным методом эвристического поиска в больших и сложных областях». [8]

Но не стоит расценивать генетические алгоритмы как своеобразную панацею для задач оптимизации. С большой вероятностью генетические алгоритмы покажут как минимум не лучшие результаты по сравнению со специально разработанными методами для решения специализированных задач. Большой плюс эволюционных вычислений в предоставляемом ими унифицированном подходе к решению самых различных проблем.

Генетические алгоритмы показывают блестящие результаты при

[1] Нейронная сеть Хопфилда — полносвязная нейронная сеть с симметричной матрицей связей. В процессе работы динамика таких сетей сходится к одному из положений равновесия. Эти положения равновесия являются локальными минимумами функционала, называемого энергией сети (в простейшем случае — локальными минимумами отрицательно определённой квадратичной формы на n-мерном кубе). Сеть может быть использована как автоассоциативная память, как фильтр, а также для решения некоторых задач оптимизации.

решении сложных переборных задач (большинство из которых NP-полные, т.е. не решаются полным перебором за полиномиальное время), таких, например, как задача коммивояжера и поиск булевых термов.

1.2. Основные понятия генетических алгоритмов

При описании генетических алгоритмов используются определения, заимствованные из генетики в упрощенном виде и основные понятия линейной алгебры.

Вектор - упорядоченный набор чисел, называемых компонентами вектора.

Булев вектор - вектор, компоненты которого принимают значения из двух элементного (булева) множества значений, как правило {0, 1}.

Популяция - конечное множество особей.

Особь (индивидуум, организм) - набор хромосом с закодированными в них множествами *параметров задачи,* то есть решений, которые иначе называются *точками в пространстве поиска* (searchpoints).

Хромосома (цепочка или кодовая последовательность) - это вектор генов. Хромосома может быть представлена в виде булева вектора, полученного с помощью двоичного кодирования либо кода Грея (см. приложение А). Хромосома обозначается, какправило,

Ген (свойство, знак или детектор) - атомарный элемент генотипа, в частности, хромосомы. Несет в себе наследственную информацию. Обозначается x .

Связь хромосомы и гена изображена на рисунке 1.

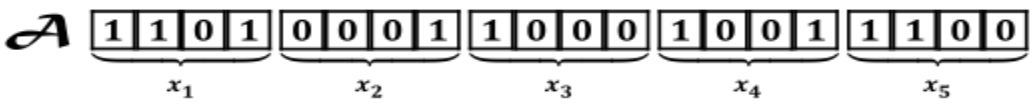


Рисунок 1 - Распределение наследственной информации по длине хромосомы.

Генотип (структура) - набор хромосом данной особи. Следовательно, особями популяции могут быть генотипы либо единичные хромосомы (в самом простом случае, генотип состоит из одной хромосомы).

Фенотип - набор значений, соответствующих данному генотипу, то есть это декодированная структура или множество параметров задачи (решение, точка пространства поиска).

Аллель - значение конкретного гена.

Локус - позиция, указывающая место размещения данного гена в хромосоме. Множество позиций генов - *локи.*

Очень важным понятием в генетических алгоритмах считается *функция приспособленности* (fitnessfunction), иначе называемая *функция оценки.* Эта функция играет важнейшую роль, поскольку позволяет определить степень приспособленности конкретных особей в популяции и выбрать из них наиболее приспособленные (то есть имеющие наибольшие значения функции приспособленности) в соответствии с эволюционным принципом выживания сильнейших. Функция приспособленности так же получила свое название из генетики.

В задачах оптимизации функция приспособленности, как правило, оптимизируется, (точнее сказать, максимизируется) и называется целевой функцией. В задачах максимизации целевая функция преобразуется, и проблема сводится к максимизации. В теории управления функция приспособленности может принимать вид функции погрешности, а в теории игр - стоимостной функции. На каждой итерации генетического алгоритма приспособленность каждой особи данной популяции оценивается при помощи функции приспособленности, и на этой основе создается следующая популяция особей,

составляющих множество потенциальных решений проблемы.

Очередная популяция в генетическом алгоритме называется *поколением,* а к вновь создаваемой популяции особей применяется термин «новое поколение» или «поколение потомков».

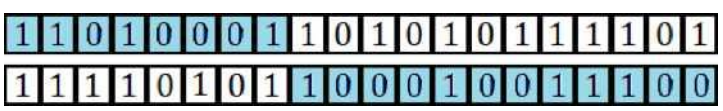
1.3. Генетические операторы

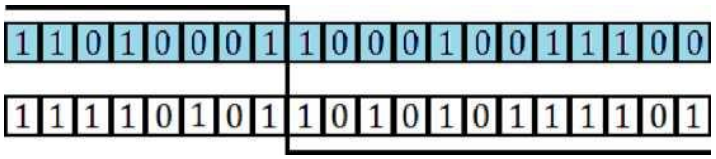
Генетические операторы необходимы для того, чтобы применить принципы наследственности и изменчивости к популяции. Помимо отличительных собенностей, о которых будет рассказано ниже, для всех операторов определено такое свойство как *вероятность*. То есть описываемые операторы не обязательно применяются ко всем скрещиваемым особям, что вносит дополнительный элемент неопределенности в процесс поиска решения. В данном случае, неопределенность не подразумевает негативный фактор, а является своеобразной "степенью свободы" работы генетического алгоритма.

Оператор кроссинговера (кроссовера, скрещивания, рекомбинации) - оператор, при котором хромосомы обмениваются своими частями. Моделирует процесс скрещивания особей.

Пусть имеются две родительские особи с хромосомами $\mathcal{A}$ и $\mathcal{B}$. Случайным образом определяется точка внутри хромосомы, в которой обе хромосомы делятся на две части и обмениваются ими. Назовем эту точку *точкой кроссинговера* (crossoverpoint), иногда она так же называется точкой разрыва. Описанный процесс изображен на рис. 2.

Родительская пара хромосом





Кроссинговер

Пара хромосом потомков

Рисунок 2 - Одноточечный кроссинговер.

Данный тип кроссинговера называется *одноточечным (single-pointcrossover),* т.к. при нем родительские хромосомы «разрезаются» только в одной случайной точке.

В *двухточечном* (и *многоточечном кроссинговере (multi-pointcrossover)* вообще) *кроссинговере* хромосомы рассматриваются как циклы, которые формируются соединением концов линейной хромосомы. Для замены сегмента одного цикла сегментом другого цикла требуется выбор двух точек разрыва. В таком представлении, одноточечный кроссинговер может быть рассмотрен как двухточечный, но с одной точкой разреза, зафиксированной в начале строки. Следовательно, двухточечный кроссинговер решает ту же самую задачу, но более полно. В настоящий момент исследователи соглашаются, что

двухточечный кроссинговер лучше, чем одноточечный. [7]

Кроме описанных типов кроссинговера так же существует *однородный кроссинговер (uniformcrossover).* Его особенность заключается в том, что значение каждого бита в хромосоме потомка определяется случайным образом из соответствующих битов родителей. Для этого вводится некоторая величина 0 <p<1, и если случайное число больше p, то на *i*-ую позицию первого потомка попадает *i*-ый бит первого родителя, а на *i*-ую позицию второго - *i*-ый бит второго родителя. В противном случае к первому потомку попадает бит второго родителя, а ко второму - первого. Такая операция проводится для всех битов хромосомы. При использовании однородного кроссинговера родителей может быть больше двух. В случае если используется три особи, то такой кроссинговер называется триадным (triadiccrossover).

Вероятность кроссинговера самая высокая среди генетических операторов и равна обычно 60% и больше. [3]

Немаловажным при реализации генетических алгоритмов является стратегия выбора родительской пары:

- Панмиксия (panmixia) - когда обе особи, которые составят родительскую пару, случайным образом выбираются из всей популяции, причем любая особь может стать членом нескольких пар.
- Инбридинг (inbreeding) - первый член пары выбирается равномерно случайно, а вторым с большей вероятностью будет максимально близкая к нему особь. Под «родством» здесь понимается расстояние между членами популяции в смысле геометрического расстояния особей в пространстве параметров (фенотипов), например, для векторов можно использовать евклидово расстояние.
- Аутбридинг (outbreeding) или кроссбридинг - формирует родительские пары из максимально далеких особей.

Оператор мутации (mutation) - случайное изменение одной или нескольких позиций в хромосоме (рис.3). Данный оператор необходим для

"выбивания" популяции из локального экстремума и способствует защите от преждевременной сходимости. [3]

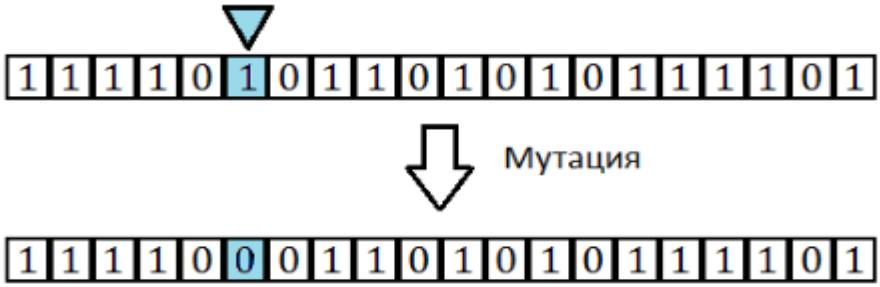


Рисунок 3 - Мутация хромосомы.

Так же как и кроссинговер, мутации могут проводиться не только в одной случайной точке. Можно выбрать для изменения несколько, причем их число может быть случайным. Также можно инвертировать сразу некоторую группу подряд идущих точек. Вероятность мутации значительно меньше вероятности кроссинговера и редко превышает 1%. Так же вероятность может быть функцией от характеристики решаемой задачи, например вероятность мутирования генов можно положить обратно пропорциональной длине хромосомы или размеру популяции.

В зависимости от типа оптимизируемой функции стратегия выбора значения вероятности мутации изменяется. Так, например, мутация с фиксированной вероятностью приводит к хорошим результатам для унимодальных функций. Для мультимодальных применяют самоадаптирующуюся оценку вероятности. Хорошим эмпирическим правилом считается выбор вероятности мутации из соотношения

$$p = \frac{1}{M} \qquad (1)$$

гдеM - число бит в хромосоме.

Оператор инверсии (inversion) - изменение порядка следования генов в хромосоме или ее фрагменте (рис. 4). Данный оператор применяется достаточно редко, но основной его целью является попытка найти порядок генов, который

имеет лучший эволюционный потенциал. Инверсия также значительно расширяет область поиска. Мало того, что генетический алгоритм пытается находить хорошие множества значений генов, он также одновременно пробует находить хорошее упорядочения генов. Это гораздо более трудная задача для решения.

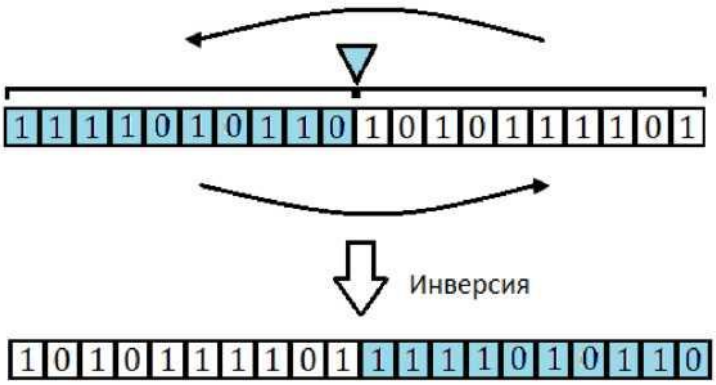


Рисунок 4 - Инверсия в хромосоме.

1.4. Описание работы классического генетического алгоритма

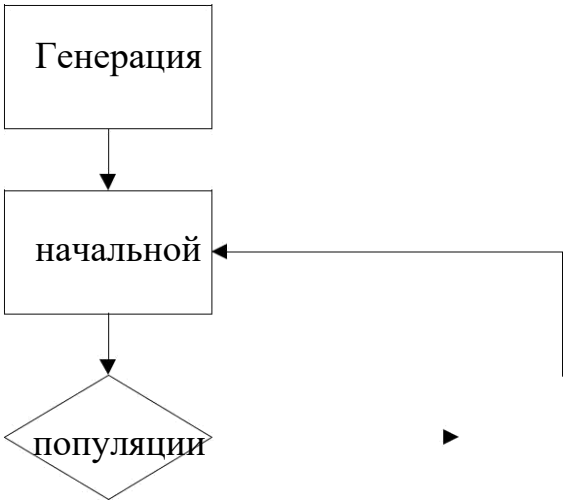
В отличие от эволюции, происходящей в природе, генетический алгоритм только моделирует те процессы в популяции, которые являются существенными для развития.

Наиболее приспособленные особи получают возможность "воспроизводить" потомство с другими особями популяции, что приводит к появлению новых особей, сочетающих в себе некоторые характеристики, наследуемые ими от родителей. Менее приспособленные особи с меньшей вероятностью смогут воспроизвести потомков, так что те свойства, которыми они обладали, будут постепенно исчезать из популяции в процессе эволюции.

Таким образом, воспроизводится вся новая популяция допустимых решений, путем выбора лучших представителей предыдущего поколения, скрещивания их и получения множества новых особей. Это новое поколение будет содержать более высокое соотношение характеристик, которыми обладают хорошие члены предыдущего поколения. В итоге, хорошие

характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных особей приводит к тому, что исследуются наиболее перспективные участки пространства поиска. В конечном итоге, популяция будет сходиться к оптимальному решению задачи.

Имеется много способов реализации идеи биологической эволюции в рамках генетического алгоритма. Схема генетического алгоритма представлена на рис. 5.



Ниже подробнее рассмотрим шаги алгоритма

1. **Генерация начальной популяции**

Из биологии мы знаем, что любой организм может быть представлен своим фенотипом (физическое представление объекта), который фактически определяет, чем является объект в реальном мире, и генотипом, который содержит всю информацию об объекте на уровне хромосомного набора. При

этом каждый ген, то есть элемент информации генотипа, имеет свое отражение в фенотипе. Таким образом, для решения задач нам необходимо представить каждый признак объекта в форме, подходящей для использования в генетическом алгоритме, то есть произвести кодирование решений для того, чтобы генетический алгоритм смог с ними работать.

Все дальнейшее функционирование механизмов генетического алгоритма производится на уровне генотипа, позволяя обойтись без информации о внутренней структуре объекта, что и обуславливает его широкое применение в самых разных задачах.

Генетический алгоритм для представления генотипа объекта применяет битовые строки, и в дальнейшем все его операторы работают только со строками. При этом каждому атрибуту объекта в фенотипе соответствует один ген в генотипе объекта. Ген представляет собой битовую строку, чаще всего фиксированной длины, которая представляет собой значение этого признака.

Для кодирования признаков можно использовать самый простой вариант - битовое значение этого признака. Тогда нам будет весьма просто использовать ген определенной

длины, достаточной для представления всех возможных значений такого признака. Но, к сожалению, такое кодирование не лишено недостатков. Основной недостаток заключается в том, что соседние числа отличаются в значениях нескольких битов, так, например числа 7 и 8 в битовом представлении различаются в 4-х позициях, что значительно увеличивает размер поискового пространства. Одним из решений данной проблемы является использование кода Грея (см. приложение А).

И, наоборот, для того чтобы определить фенотип объекта (то есть значения признаков, описывающих объект) нам необходимо только знать значения генов, соответствующие этим признакам, то есть генотип объекта. Операция определения фенотипа объекта по его генотипу называется операцией декодирования или роста, то есть мы выращиваем фенотип из генотипа.

Таким образом, для того чтобы инициализировать начальную популяцию,

необходимо сначала определиться со способами кодирования особей. [3]

1. Оценка популяции

После генерации начальной популяции особей осуществляется ее оценка с помощью операции декодирования и проверяется условие остановки. В случае, когда условие остановки не выполняется, для дальнейшего развития процесса поиска, применяются специализированные операторы генетического алгоритма, одним из которых является оператор селекции.

Оператор селекции является одним из наиболее важных операторов, помогающих поддерживать генетическое разнообразие популяции, которое отвечает за характер поведения популяции: широкий разброс точек в пространстве поиска, сбор точек вокруг некоторой точки и т.д.

Оператор селекции - генетический оператор поиска, посредством которого отбираются индивиды, имеющие разрешение (например, из-за “хорошего” значения целевой функции) на производство потомства. То есть селекция состоит в том, что родителями могут стать только те особи, значение приспособленности которых не меньше пороговой величины, например, среднего значения приспособленности по популяции. Такой подход обеспечивает более быструю сходимость алгоритма.

Однако из-за быстрой сходимости пропорциональный выбор родительской пары не подходит тогда, когда становится задача определения нескольких экстремумов, поскольку для таких задач алгоритм, как правило, быстро сходится к одному из решений. Кроме того, для некоторых многомерных задач со сложным ландшафтом целевой функциибыстрая сходимость может привести к квазиоптимальному решению. То есть возникают проблемы преждевременной сходимости и стагнации.

Пусть *fi* - пригодность i-ой особи тогда средняя пригодность популяции размера N вычисляется по формуле:

$$\bar{f} = \frac{1}{N} \times \sum_{i=1}^{N} f_i \qquad (2)$$

Преждевременная сходимость возникает, когда на ранних поколениях появляется особь с пригодностью ,$f_k \gg \bar{f}$ но$f_k \ll f_{\max}$ □. Гены такой особи довольно быстро распространятся на всю популяцию, и кроссинговер не может более производить новых особей (может только мутация). В таком случае средняя пригодность остается$\bar{f} \ll f_{\max}$ □.

Стагнация возникает, когда ближе к концу работы алгоритма все особи в популяции получают относительно высокую и примерно одинаковую пригодность$\forall k\ f_k \approx f_{\max}$ □, что приводит к очень маленькому селективному давлению, т.е. вероятность выбора лучшего решения немного больше, вероятности выбора худшего.

Эти недостатки могут быть отчасти компенсированы использованием подходящего механизма отбора, который бы тормозил слишком быструю сходимость алгоритма.

При реализации генетических алгоритмов чаще всего используются следующие типы селекции:

- *Селекция по методу рулетки (roulette-wheelselection).* Для скрещивания особи отбираются с помощью запусков рулетки *N* раз. Колесо рулетки содержит по одному сектору для каждой особи из популяции. Размер i-го сектора пропорционален вероятности попадания особи P_i, которая рассчитывается по формуле:

$$p_i = \frac{f_i}{\sum_j^N f_j} \quad (3)$$

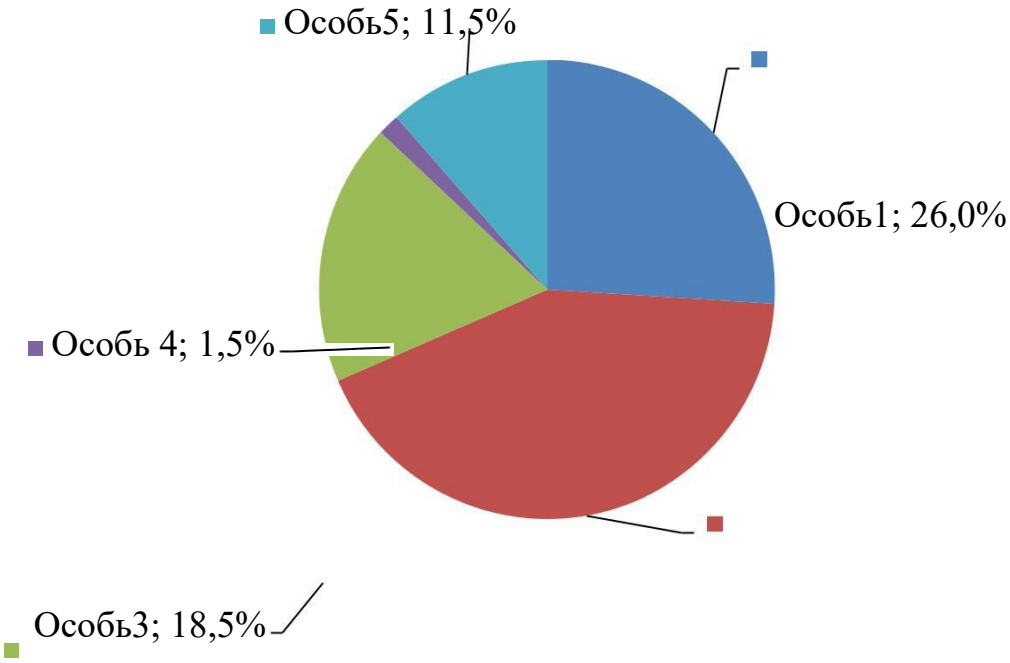
Очевидно, что для всей популяции выполняется условие $\sum_{i=1}^{N} p_i = 1$ При таком отборе члены популяции с более высокой приспособленностью с большей вероятностью будут чаще выбираться, чем особи с низкой приспособленностью (табл. 1 и рис. 6).

Таблица 1 - Метод рулетки. Размер популяции = 5. Суммарная пригодность популяции = 200.

Особь	Пригодность	Вероятность	Доля

		выбора	
Особь$_1$	52	52/200 = 0.26	26%
Особь$_2$	85	85/200 = 0.425	42.5%
Особь$_3$	37	37/200 = 0.185	18.5%
Особь$_4$	3	3/200 = 0.015	1.5%
Особь$_5$	23	23/200 = 0.115	11.5%

Колесо рулетки



Особь2; 42,5%

Рисунок 6 - Оператор селекции типа колеса рулетки с пропорциональными функции
приспособленности секторами

• *Турнирная селекция (tournamentselection)* - из популяции, состоящей из N особей, создается группа из $t\ (t \geq 2)$ особей, выбранных случайным образом (рис. 6). Особь с наибольшей пригодностью в группе отбирается, остальные - отбрасываются. Такая операция повторяется k раз. Затем отобранные особи используются для кроссинговера. Размер группы *t* часто равен 2, в таких случаях говорят о парных (двоичных) турнирах (binarytournament). Число t называется численностью турнира (tournamentsize).

Преимуществом турнирной селекции является то, что она не требует дополнительных вычислений и упорядочивания особей в популяции по возрастанию приспособленности. [3]

- Отбор усечением (truncationselection) – индивиды сортируются

(ранжируются) на основе их пригодности таким образом, чтобы $f_i \geq f_j\, k$ для $i \leq j$, т.е.

первым стоит наиболее приспособленный индивид. Число особей для скрещивания

выбирается в соответствии с порогом $\mathbf{T}\epsilon[\mathbf{0};\mathbf{1}]$. Порог определяет, какая доля особей,

начиная с самой первой (то есть самой приспособленной) будет принимать участие в

отборе. Порог может быть задан и числом больше 1, тогда он будет равен числу особей из

текущей популяции, допущенных к отбору. Среди особей, попавших "под

порог",
случайным образом N раз выбирается самая везучая, среди которых затем выбираются
особи непосредственно для скрещивания

Из-за того, что в этой стратегии используется отсортированная популяция, время её работы может быть большим для популяций большого размера и зависеть также от алгоритма сортировки. [3]

3. Проверка условия остановки

Определение условия остановки генетического алгоритма зависит от его конкретного применения. В оптимизационных задачах, если известно максимальное (или минимальное) значение функции приспособленности, то остановка алгоритма может произойти после достижения ожидаемого оптимального значения, возможно - с заданной точностью. Остановка алгоритма также может произойти в случае, когда его выполнение не приводит к улучшению уже достигнутого значения. Алгоритм может быть остановлен по истечении определенного времени выполнения либо после выполнения заданного количества итераций. Если условие остановки выполнено, то производится переход к завершающему этапу выбора «наилучшей» хромосомы.

4. Генерация новой популяции

Данный шаг является, в своем роде, одним из видов селекции. Здесь происходит применение генетических операторов и отбор индивидов теперь уже из двух популяций (родители и потомки) в новую популяцию, которая будет работать на следующем поколении.

Для того чтобы индивидуальные алгоритмы обладали разными стратегиями оптимизации необходимо обеспечить разнообразие в схемах формирования нового поколения. Существуют различные схемы формирования нового поколения, рассмотрим основные:

• *Стратегия элитарности (elitismstrategy)* - метод основан на построении новой популяции только из лучших особей репродукционной

группы, объединяющей в себе родителей и их потомков. Данный метод хорош с той точки зрения, что исключает «случайное блуждание по пространству поиска», поскольку осуществляется переход в следующее поколение самой лучшей особи (найденной на данном этапе поиска или ранее);

- *Отбор с вытеснением (exclusionselection)* - отбор, построенный на таком

принципе, носит бикритериальный характер - то, будет ли особь из репродукционной

группы заноситься в популяцию нового поколения, определяется не только величиной ееприспособленности, но и тем, есть ли уже в формируемой популяции следующего

поколения особь с аналогичным хромосомным набором. Из всех особей с динаковыми

генотипами предпочтение сначала, конечно же, отдается тем, чья приспособленность выше. Таким образом, достигаются две цели: во-первых, не теряются лучшие найденные решения, обладающие различными хромосомными наборами, а во-вторых, в популяции постоянно поддерживается достаточное генетическое разнообразие [2];

- *Только потомки (детерминизм)* - метод основан на построении новой популяции только из популяции потомков;
- *Случайным образом (стохастика)* - когда особи, которые составят новую популяцию, выбираются случайным образом из репродукционной группы, объединяющей в себе родителей и их потомков.

5. Вывод наилучшей особи

Если условие остановки алгоритма выполнено, то следует вывести результат работы, т.е. представить искомое решение задачи. Лучшим решением считается особь с наибольшим значением функции приспособленности.

Используя генетические операторы, схема генетического алгоритма будет

выглядеть следующим образом [2];

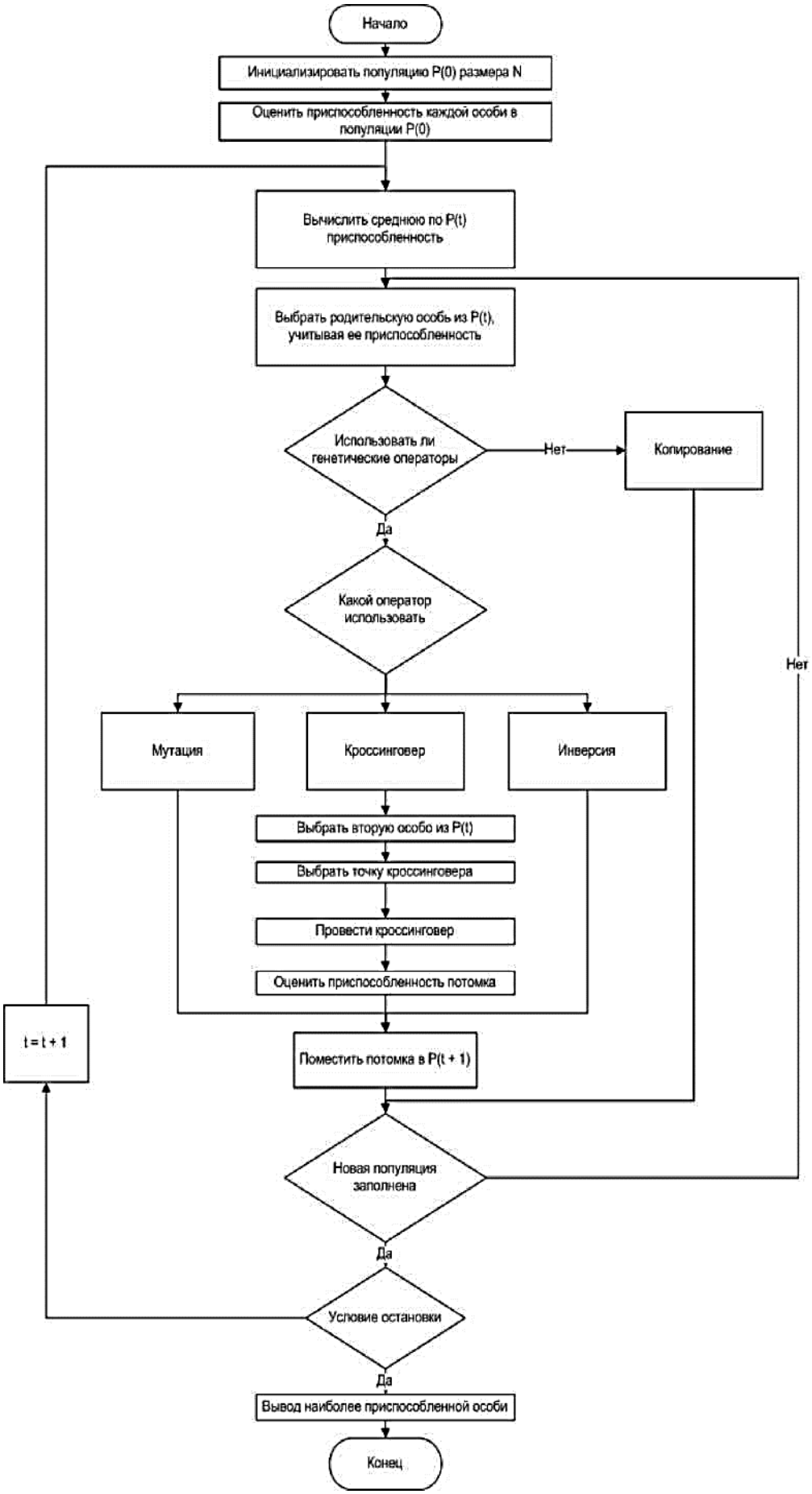


Рисунок 7 - Расширенная схема генетического алгоритма

2. ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

2.1. Постановка задачи безусловной оптимизации

Любая задача безусловной оптимизации выглядит следующим образом:

$$max(min) f(\bar{x}), \text{где } x = (x^1, x^2, \dots, x^n) x_i \in [a, b] i = \overline{1, n} \quad (4)$$

где - максимизируемая (минимизируемая) целевая функция, имеющая

один глобальный экстремум. Предполагается, что о функции известно лишь то, что она определена в любой точке области поиска. Любая дополнительная информация о характере функции и ее свойствах (дифференцируемость, непрерывность, свойства Липшица и т.д.) предполагается неизвестной и не учитывается в процессе поиска.

Под решением задачи (4) будем понимать вектор $\bar{x} = (x^1, x^2, .., x_n)$. Оптимальным решением задачи (4) будем считать вектор *x,* при котором целевая функция *f (x)* принимает максимальное (минимальное) значение.

Практически всегда оптимизируемая функция обладает каким-либо свойством (свойствами): многоэкстремальность, алгоритмическое задание, сложная конфигурация допустимой области, наличие нескольких типов переменных. Это приводит к необходимости применения специализированных методов, к которым и относятся эволюционные и генетические алгоритмы, хорошо зарекомендовавшие себя в ситуациях, когда применение стандартных методов оптимизации крайне затруднено.

2.2. Ограничения при реализации генетических алгоритмов

Суть использования генетических алгоритмов держится на трех принципах: кодирование, оценивание и воспроизводство, но с практической точки зрения имеют смысл иные качества, которые, однако, нисколько не отменяют основные параллели с эволюционными механизмами.

Под *кодированием* понимается способ представления данных в генетическом виде. Здесь важно, чтобы была возможность получить решение в хромосоме, а также, чтобы в генотипе мог быть записан любой корректный вариант, более или менее претендующий на то, чтобы оказаться ответом на поставленную задачу. В большинстве случаев проблем с этим не возникает, однако для одной и той же задачи может существовать несколько способов генетического представления параметров, которые могут существенно влиять на скорость генетического поиска и качество решения.

Оценивание является еще одним важным принципом. Смысл оценивания заключается в том, чтобы различать особей в зависимости от того, насколько "успешны" соответствующие им закодированные решения. При этом не должно возникать коллизий, когда две практически равноценные особи имеют существенно различные значения приспособленностей, и, наоборот, когда качественно разные особи оцениваются одинаково.

Основная цель *воспроизводства* - получение новых вариантов решений-кандидатов из уже существующих. Здесь очень желательно, чтобы при скрещивании родительских особей получались корректные в рамках поставленной задачи потомки. В ряде случаев это условие требует использования "нестандартных" генетических операторов и/или специфичного кодирования. К примеру при решении задачи коммивояжера в маршруте не должна два раза и чаще встречаться одна и та же вершина, что часто получается в результате применения традиционно используемых операторов одно- и двухточечного и однородного кроссинговера. Поэтому для данной

проблемы разработаны специальные операторы скрещивания и мутации.

Так же важным параметром генетических алгоритмов является размер популяции. При практической реализации возможны две крайности:

- слишком малый размер популяции (<10). Данный выбор в большинстве случаев годится только для очень простых задач. В противном случае будет наблюдаться быстрое вырождение популяции;
- слишком большой размер популяции (>1000). Как и полагается, решение скорее всего, будет найдено за меньшее число поколений, однако часто ценой лишних вычислительных затрат. В некоторых случаях, когда просто надо найти решение, это не критично. Однако бывает так, что необходимо продемонстрировать преимущества (если есть) генетического подхода для решения выбранной проблемы перед уже методами и алгоритмами.

Исходя из данных рекомендаций, оптимальным размером популяции является 2030 особей, однако в некоторых задачах требуется 50-100 особей. Исследования показывают, что размер популяции во многом зависит от размера хромосом. Так, для алгоритма с 32-битовыми хромосомами размер популяции будет больше, чем для алгоритма с 16-битовыми. [7]

Так как у генетических алгоритмов есть не оцениваемая численно характеристика, описывающая их поисковые способности, они зависят от всего, но в большей степени от стратегий селекции и генетических операторов. Отсюда:

- использование более агрессивных вариантов отбора вкупе с достаточно большой вероятностью мутации во многих случаях позволяет добиться более хороших результатов, по сравнению с каноническим генетическим алгоритмом. Агрессивными стратегиями отбора можно считать отбор усечением с достаточно большим порогом (т.е. когда к воспроизводству допускается меньшее количество особей), а также турнирный отбор с

размером турнира 4 и больше;

- популяция большего размера работает стабильнее и зачастую лучше. Если же необходимо уложиться в некоторое количество вычислений целевой функции, то лучше поискать оптимальный размер, при котором и решение может быть найдено, и вычислительные затраты вполне приемлемые;
- двухточечный и однородный операторы кроссинговера, как правило, работают лучше, чем одноточечный;
- планомерное выслеживание и ликвидация диверсионных элементов в лице дубликатов в популяции повышают качество результатов и полезно против преждевременной сходимости;
- применение стратегии элитарности - позволяет гарантированно оставить в популяции наилучших особей;
- большая вероятность мутации в некоторых случаях способна улучшить работу алгоритма (особенно для малых популяций), но нежелательна, в силу внесения большой хаотичности в эволюционный процесс, что может отрицательно сказаться на стабильности работы алгоритма. [3]

2.3. Решение задачи коммивояжера с помощью генетического алгоритма

Рассмотрим достоинства и недостатки стандартных и генетических методов на примере классической *задачи коммивояжера* (TSP - travelingsalesmanproblem), которая является одной из самых известных задач дискретной оптимизации. Она формулируется следующим образом: дан

полный взвешенный граф $G(X, V)$ порядкаn, где - множество вершин; $V \subseteq X \times X$ - множество ребер. В данном графе нужно найти Гамильтонов

цикл, имеющий наименьший суммарный вес входящих в него ребер.

Илиформально:

$$\begin{cases} Q(x) = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij} \rightarrow min \\ \sum_{i=1}^{N} x_{ij} = 1, \qquad \forall j = \overline{1,N} \\ \sum_{j=1}^{N} x_{ij} = 1, \qquad \forall j = \overline{1,N} \\ x_{ij} \in \{0,1\} \end{cases} \quad (5)$$

Где c_{ij} - вес ребра (i,j).

$$x_{ij} = \begin{cases} 1, \text{если есть переход из } \mathbf{i} \text{ в } j \\ \mathbf{0}, \text{если перехода из } \mathbf{i} \text{ в } \mathbf{j} \text{ нет} \end{cases} \quad (6)$$

Очевидно, что решением задачи является перестановка из N вершин, количество возможных перестановок равно $N!$, однако количество различных решений задачи с

учетом направления обхода и сдвига начальной вершины будет $\frac{(N-1)!}{2}$.

Данная задача относится к классу NP-полных задач, то есть время работы алгоритма, решающего задачу коммивояжера, существенно зависит от размера входных данных, то есть от количества городов.

Все эффективные (сокращающие полный перебор) методы решения задачи коммивояжёра — методы эвристические. В большинстве эвристических методов находится не самый эффективный маршрут, а приближённое решение. Зачастую востребованы так называемые any-time алгоритмы, то есть постепенно улучшающие некоторое текущее приближенное решение.

На практике применяются различные модификации более эффективных методов: метод ветвей и границ и метод генетических алгоритмов, а также алгоритм муравьиной колонии[2].

Для того чтобы задачу можно было решить с помощью генетических алгоритмов, нужно выяснить, что именно является решением этой задачи,

[1]Алгоритм муравьиной колонии - один из эффективных полиномиальных алгоритмов для нахождения приближённых решений задачи коммивояжёра, а также аналогичных задач поиска маршрутов на графах. Суть подхода заключается в анализе и использовании модели поведения муравьёв, ищущих пути от колонии к источнику питания и представляет собой мета эвристическую оптимизацию.

закодировать решение в виде хромосомы и составить функцию приспособленности для таких хромосом. Только после этого можно решать эту задачу средствами генетических алгоритмов.

Выясним, что можно считать решением задачи коммивояжера. Очевидно, что каким-либо решением будет любой маршрут между городами, удовлетворяющий следующим условиям: он пересекает все без исключений города и не один не пересекает больше одного раза. Закодировать такой маршрут можно в виде последовательности номеров городов, начиная с самого первого, в конце последовательности номерпредпоследнего города, так как маршрут замкнут и последним будет город, с которого он начинался. Очевидно, что в этой последовательности не будет повторяющихся значений. Пусть для простоты примера количество городов N = 8, тогда одной из возможных последовательностей городов будет путь, изображенный на рис. 8.

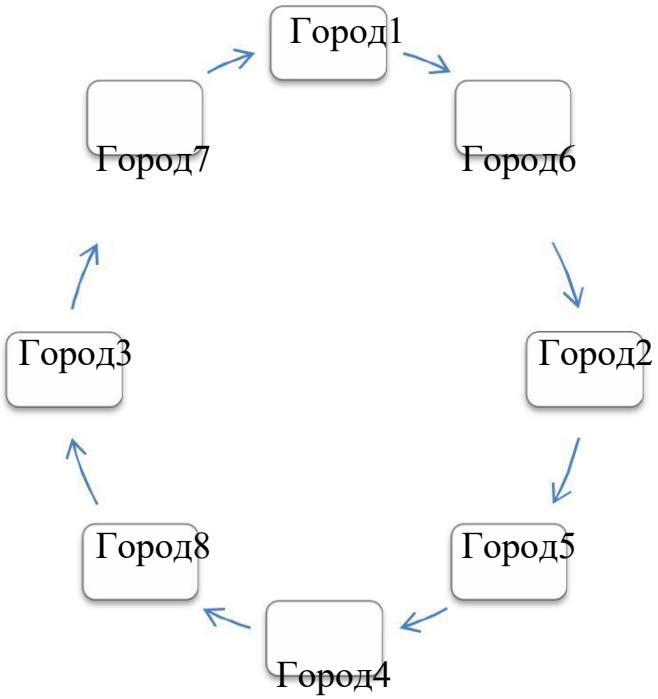


Рисунок 8 - Пример маршрута коммивояжера при обходе 8 городов

Закодируем города числами от 1 до 8. Тогда тот же самый путь примет вид:

1|6|2|5|4|8|3|7.

Теперь нам нужно представить решение в виде хромосомы. Выше мы уже

закодировали решение в виде последовательности номеров городов, теперь осталось перекодировать ее хромосому. Для определенности будем считать, что мы кодируем в хромосому в виде битового вектора. Очевидно, что длина гена в битах в хромосоме будет равна:

$$L = \log_2 N \quad (7)$$

Для нашего примера $L = \log_2 \square 8 = 3$, то есть для кодирования одного гена понадобиться 3 бита. Кодируем последовательность с помощью двоичного кодирования (табл. 2):

Таблица 2 - Кодирование последовательности городов с помощью двоичного кодирования.

000	101	001	100	011	111	010	110
1	6	2	5	4	8	3	7

Однако представив решение таким образом, мы не учли несколько существенных факторов: при случайной генерации начальной популяции может возникнуть хромосома, в которой будут повторяющиеся значения генов: 000 000 010 011 100 101 110 010.

1. хромосомы с повторяющимися генами может дать кроссинговер или мутация.

Есть несколько способов решения этого недостатка кодирования, но все они ведут к излишнему потреблению вычислительных ресурсов, т.к. надо дополнительно проверять хромосомы. Один из способов - проверять на повторяющиеся значения внутри функции приспособленности, и, встретив такие, заменять их на те значения, которых нет в хромосоме. Второй способ - ничего не проверять, а присвоить таким хромосомам очень низкое значение функции приспособленности, но в этом случае генетический алгоритм начинает крайне неэффективно работать.

Вообще говоря, для генетических алгоритмов очень важен вопрос кодирования решений в последовательность генов. От того, насколько оно

удачно, зависит качество работы алгоритма. Самое главное, и обязательное, требование к кодированию - хромосома должна однозначно представлять некоторое решение, чтобы не было возможности трактовать одну и ту же хромосому по-разному. Желательно, чтобы хромосомы занимали как можно меньше бит, были короче. Так же важным условием является простота кодирования. От этого зависит скорость работы.

После кодирования запускается генетический алгоритм с желаемыми параметрами.

2.4 Реализация генетического алгоритма для решения задачи коммивояжера с использованием пакета MATLAB 7.5

Пакет MALAB 7.5 предоставляет встроенную панель инструментов GenethicAlgorithmandDirectSearchToolbox™, которая предназначена для расширения функциональных возможностей пакета генетическими алгоритмами. Работать с данной панелью инструментов можно двумя способами: с консоли или вызвав панель GenethicAlgorithmTool с помощью команды gatool. По сути, оба способа являются идентичными с той разницей, что используя панель GenethicAlgorithmTool, любые параметры генетического алгоритма настраиваются с использованием графической оболочки.

Рассмотрим встроенные функции для работы с генетическими алгоритмами [9]:

- [x fval reason output population scores] = ga(@fitnessfun, nvars, options) - функциядлянахожденияминимумацелевойфункции;

  Входные параметры:

  - @fitnessfun - указатель функции в М-файле, по которой производится расчет функции приспособленности; onvars - число независимых переменных для функции приспособленности; ooptions - настраиваемые параметры генетического алгоритма;

  Выходные параметры:

  - x - конечная точка расчета;
  - fval - значение функции приспособленности в точке x; oreason - причина остановки алгоритма;
  - output - структура с информацией о эффективности работы алгоритма для каждого выполненного поколения;
  - population - состояние последнего семейства; oscores - конечное состояние;

- val = gaoptimget(options, 'Name') - возвращает параметры используемого генетического алгоритма;

  Входные параметры:

  - options - структура с параметрами генетического алгоритма; o 'Name' - имя параметра, значение которого нужно извлечь;

  Выходные параметры:

  - val - значение запрашиваемого параметра;

- options = gaoptimset('paraml', value1, 'param2', value2, . . .) - устанавливает параметры генетического алгоритма; Входные параметры:

  - o'paraml' - имя параметра; ovalue1- устанавливаемое значение;

  Выходные параметры:

  - options- структура с параметрами генетического алгоритма.

Для решения задачи коммивояжера будем генерировать города случайным образом

в отрезке [0, 1], при этом расстояния между городами подчиняются правилу треугольника.

Используя встроенные функции MATLAB, реализуем генетический алгоритм со следующими характеристиками и ограничениями:

1. Хромосома представлена в виде бинарного вектора;
2. Размер популяции - 50 особей;
3. Для расчета функции приспособленности используется стандартная функция travelling_salesman_fitness, принимающая в качестве аргумента хромосому (то есть одна из возможных перестановок городов) и возвращающая значение функции приспособленности для нее;
4. Кроссинговер: применяемая стратегия - генерация перестановок (используется стандартная функция crossover_permuation), вероятность кроссинговера
   - 0.8;
5. Мутации: стратегия - так же применяется стандартная функция mutate_permutation, которая с определенной вероятностью генерирует перестановки в векторе пути, вероятность мутации - 0.2. Величина мутации такая большая, так как используется относительно небольшая популяция и велика вероятность попадания в локальный минимум;
6. Применяется стратегия элитарности; из каждой предыдущей популяции остается 2 наилучшие особи;
7. Критерием остановки выполнения алгоритма является сохранение значений функции приспособленности на протяжении 50 поколений (для количества городов >100), 1000 поколений (для 100 - 400 городов), 7500 поколений (>400 городов);

Сравним эффективность решения задачи коммивояжера с помощью генетического алгоритма с эффективными методами решения на одинаковых наборах данных. Для решения задачи коммивояжера с помощью минимального остова без(с) оптимизацией была использована программа, написанная на C++. В качестве алгоритма построения минимального остова был использован алгоритм Прима. Далее к решению, полученному с помощью минимального остова, применялась оптимизация: для участка пути длинной, где k - шаг оптимизации, генерировались всевозможные перестановки, то есть вокруг города i генерировались перестановки в участке пути, в который входят города $i-k \ldots i \ldots i + k$. Количество перестановок тогда будет равно $(2k - 1)!$.

Таблица 4 – Сравнение времени решения задачи коммивояжера с помощью генетических алгоритмов с эффективными методами решения на одинаковых наборах данных (время в мс).

Количество городов	Точное решение	Генетический алгоритм	Решение с помощью минимального	Оптимизация решения с помощью минимального остова (k = 3)	Оптимизация решения с помощью минимального остова (k = 5)
5	1,4274	1,4274	1,4608	1,4608	-
10	3,1710	3,1710	3,5153	3,1710	3,1710
15	3,2265	3,2266	3,3323	3,2265	3,2265
20	-	3,9634	5,5095	4,7750	4,7750
50	-	5,8946	7,9498	6,6888	6,2239
100	-	8,7381	10,790	10,051	9,3428
200	-	11,9872	15,1465	13,7863	12,8208
400	-	16,3385	20,6697	18,8074	17,4822
800	-	23,4532	29,5846	27,0346	25,5666

1600	-	33,2398	41,2549	37,8895	35,2624
3200	-	-	58,0455	53,1781	49,6798
6400	-	-	81,5697	75,82	72,8593
10000	-	-	101,539	93,6233	89,4984

Таблица 4 – Сравнение времени решения задачи коммивояжера с помощью генетических алгоритмов с эффективными методами решения на одинаковых наборах данных (время в мс).

Количество городов	Точное решение	Генетический алгоритм	Решение с помощью минимального остова	Оптимизация решения с помощью минимального остова (k =	Оптимизация решения с помощью минимального остова (k
1	2	3	4	5	6
5	0	82	0	0	-
10	15	1061	0	0	50
15	12750	1532	0	0	421
20	-	2593	0	0	968

Таблица 4 - Сравнение времени решения задачи коммивояжера с помощью генетических алгоритмов с эффективными методами решения на одинаковых наборах данных (время в мс, продолжение).

1	2	3	4	5	6
50	-	3812	0	15	1921
100	-	6734	0	15	6890
200	-	25719	0	46	13761
400	-	145414	31	93	21906
800	-	480854	78	156	49218
1600	-	1465773	343	617	93233
3200	-	-	1343	2089	189639
6400	-	-	6343	7456	386234
10000	-	-	26781	30421	586580

Анализируя табл. 3 и табл. 4, очевидно, что генетический алгоритм находит более точное решение по сравнению с решением задачи с помощью минимального остова даже с оптимизацией. Однако при решении задачи коммивояжера с помощью генетического алгоритма возникает ряд проблем - нужно менять условие остановки алгоритма в зависимости от числа городов, поскольку генетические алгоритмы имеют относительно плохую сходимость при большой длине хромосомы; так же генетические алгоритмы имеют не очень хорошие показатели по времени (рис. 9).

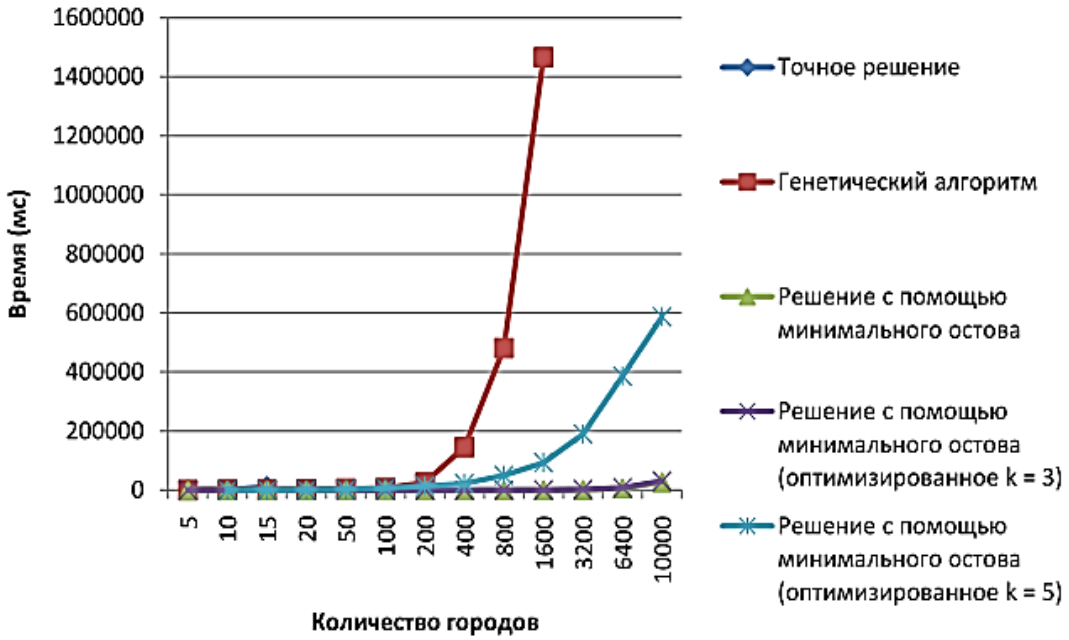


Рисунок 9 - График зависимости времени от числа городов для разных способов решениязадачикоммивояжера.

Согласно исследованиям генетические алгоритмы имеют трудоемкость в среднем $O(n^m)$, где n - длина хромосомы (то есть количество городов), m - отражает влияние размера популяции и вероятностей генетических операторов. Узким местом генетических алгоритмов является многократное вычисление значения функции приспособленности. Количество вычислений равняется:

$$K = N * M \quad (8)$$

где N - размер популяции, M - количество поколений.

Попробуем взять в качестве исходных данных для решения задачи коммивояжера генетическим алгоритмом полученные решения с помощью минимального остова с и без оптимизации (табл. 5 и табл. 6). При больших количествах городов также приходится менять условие остановки алгоритма.

Количество городов	Исходное полученное решение	Решение с помощью минимального остова	Оптимизация решения с помощью минимального	Оптимизация решения с помощью минимального
5	1,4274	1,4608	1,4608	-
10	3,1710	3,1710	3,1710	3,1710
15	3,2266	3,2265	3,2265	3,2265
20	3,9634	3,4328	3,4328	3,4328
50	5,8946	5,8946	5,8946	5,8946
100	8,7381	7,8422	7,5621	7,5621
200	11,9872	11,3452	10,0232	10,0232
400	16,3385	15,5464	14,6119	14,6119
800	23,4532	21,6932	19,7432	19,7432
1600	33,2398	35,0656	30,6541	30,6541

3200	-	45,8116	41,3276	41,3276
6400	-	-	65,3527	65,3527
10000	-	-	76,3488	76,3488

Таблица 5 - Сравнение точности решения задачи коммивояжера с помощью

Количество городов	Исходное время	Решение с помощью минимального остова	Оптимизация решения с помощью минимального остова (k = 3)	Оптимизация решения с помощью минимального остова (k = 5)

генетических алгоритмов с эффективными методами решения. Набор данных - решение задачи, полученное с помощью эффективных способов.

5	82	0	0	-
10	1061	0	0	0
15	1532	0	0	0
20	2593	15	15	15
50	3812	93	76	76
100	6734	532	129	114
200	25719	1630	467	321
400	145414	5907	1945	1095
800	480854	37890	37436	35340
1600	1465773	108423	96790	93827
3200	-	245906	254242	252352
6400	-	1035621	676432	650936
10000	-	-	1356214	1295308

Таблица 6 - Сравнение времени решения задачи коммивояжера с помощью генетических алгоритмов с эффективными методами решения (время в мс). Набор данных - решение задачи, полученное с помощьюэффективных способов.

Очевидно, что решение имеет лучшее качество, но количество шагов для оптимизации, взятых для решения задачи с помощью минимального остова, никак не влияет на него. Из этого можно сделать вывод, что генетические алгоритмы можно применять после эффективных способов решения для увеличения точности. Время выполнения так же значительно сократилось (рис. 10).

Рисунок 10 - График зависимости времени от числа городов для решения задачи коммивояжера.

Набор данных - решение задачи, полученное с помощью эффективных способов.

На основе полученных результатов можно сделать следующий вывод: пусть есть задача, для которой может быть получено некоторое приближенное решение, тогда для того, чтобы улучшить полученное решение, его можно подать на вход генетическому алгоритму. Решение, полученное таким способом, будет более точным.

В силу особенностей своей реализации, генетические алгоритмы хорошо поддаются распараллеливанию, исходя из этого, исследуем их эффективность в таком случае.

3. РАЗВИТИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ В СТОРОНУ МОДЕЛИ С НЕСКОЛЬКИМИ ВЗАИМОДЕЙСТВУЮЩИМИ ПОПУЛЯЦИЯМИ

3.1. Модель миграции генетических алгоритмов

Генетические алгоритмы применяются и при параллельных вычислениях (parallelimplementations).

Первым подходом является распараллеливание отдельных шагов алгоритма: селекции и репликации, мутации, вычисление функции приспособленности. При этом популяция разделяется на блоки, над каждым из которых работает отдельный поток. Однако, такой подход не очень удобен в программировании.

Наиболее часто применимой является **модель миграции** (Migration). Модель миграции представляет популяцию как множество подпопуляций. Каждая подпопуляция обрабатывается отдельным процессором. Эти подпопуляции развиваются независимо друг от друга в течение одинакового количества поколений *T* (время изоляции). По истечении времени изоляции происходит обмен особями между популяциями (миграция, "воровство невест"). Количество особей, подвергшихся обмену (вероятность миграции), метод отбора особей для миграции и схема миграции определяет частоту возникновения генетического многообразия в подпопуляциях и обмен информацией между подпопуляциями.

Отбор особей для миграции может проходить следующим образом:

- случайная однообразная выборка из числа особей;
- пропорциональный отбор: для миграции берутся наиболее пригодные особи.

Отдельные подпопуляции в параллельных генетических алгоритмах можно условно принять за вершины некоторого графа. В связи с этим можно рассматривать топологию графа миграционного генетического алгоритма. Наиболее распространенной топологией миграции является полный граф (рис. 11), при которой особи из любой подпопуляции могут мигрировать в любую другую подпопуляцию. Для каждой подпопуляции полное число потенциальных "иммигрантов" строится на основе всех подпопуляций. Мигрирующая особь

случайным образом выбирается из общего числа.

При использовании в неограниченной миграции пропорционального отбора сначала формируется массив из наиболее пригодных особей, отобранных по всем подпопуляциям. Случайным образом из этого массива выбирается особь, и ею заменяют наименее пригодную особь в подпопуляции 1. Аналогичные действия проделываем с

остальными подпопуляциями. Но при таком походе возможны коллизии: что какая-то популяция получит дубликат своей "хорошей" особи.

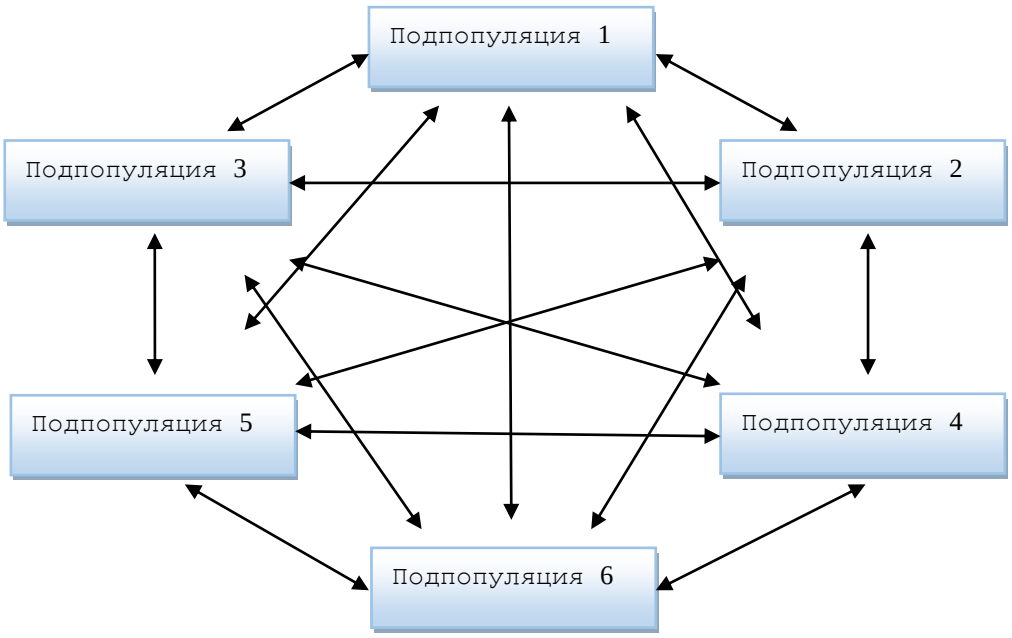


Рисунок 11 – Миграция с топологией полной сети

Другая основная миграционная схема имеет топологию кольца (рис. 12). Здесь особи передаются между соседними (по направлению обхода) популяциями. Таким образом, особи из одной подпопуляции могут мигрировать только в одну - соседнюю популяцию.

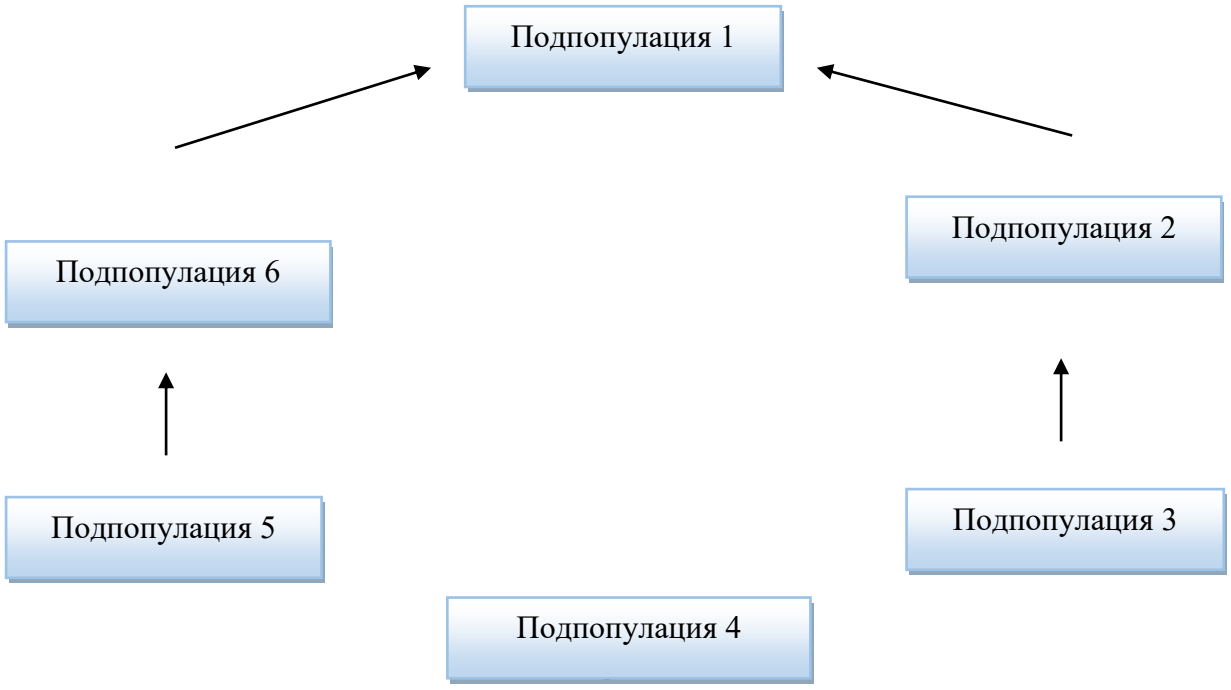


Рисунок 12 - Миграция с топологией кольца

Существует стратегия миграции, объединяющая первую и вторую модель, так же она позволяет разрешить коллизии первой модели. Как и при топологии кольца, миграция осуществляется только между ближайшими соседями, однако миграция в этой модели так же возможна между "крайними" подпопуляциями (тороидальные краевые миграции). [7]

3.2. Островная модель генетических алгоритмов

Островная модель является наиболее распространенной моделью параллельного генетического алгоритма. Ее суть заключается в том, что популяция, как правило, состоящая из очень большого числа особей, разбивается на одинаковые по размеру подпопуляции. каждаяподпопуляция обрабатывается отбельным процессором с помощью одной из разновидностей непараллельного генетического алгоритма. Изредка, например, через каждые пять поколений, подпопуляции будут обмениваться несколькими особями. Такие миграции позволяют подпопуляциям совместно использовать генетический материал.

Пусть выполняются 16 независимых генетических алгоритмов, используя подпопуляции из 100 особей в каждой. Если миграции нет, то происходит 16 независимых поисков решения. Все поиски ведутся на различных начальных популяциях и сходятся к определенным особям. Исследования подтверждают, что генетический дрейф склонен приводить подпопуляции к различным доминирующим особям. Это связано с тем, что, во- первых, количество островов, принимающих доминирующих "эмигрантов" с острова ограничено (2-5 островов).

Во-вторых, обмен особями односторонен. Поэтому в большой популяции появляться группы островов с различными доминирующими особями. Если популяция имеет небольшой размер, то возможно быстрое мигрирование ложных доминирующих особей.

Например, истинное решение находится только на одном острове, а несколько ложных доминант - на других островах. Тогда при миграции количество ложных особей на островах возрастет (на каждый остров миграции происходят с не менее двух островов), генетическим алгоритмом верное решение будет разрушено. Тем самым в маленькой популяции при генетическом дрейфе возможно появление ошибочных доминирующих особей и схождение алгоритма к ложному оптимуму.

Введение миграции в островной модели позволяет находить различные особи- доминанты в подпопуляциях, что способствует поддержанию многообразия в популяции. Каждую подпопуляцию можно принять за остров. Во время миграции подпопуляции обмениваются своим генетическим материалом. При частом мигрирование большого количества особей происходит перемешивание генетического материала. Тем самым устраняются локальные различия между островами. Очень редки миграции не позволяют предотвратить преждевременную сходимость алгоритма на маленьких популяциях. В рассматриваемой модели с каждого острова миграции могут происходить лишь на определенное расстояние: 2-5 остовов в зависимости от количества популяций. Таким образом, каждый остров оказывается почти изолированным. Количество островов, на которые могут мигрировать особи одной подпопуляции, называют *расстоянием изоляции.* Следует заметить, что в такой модели взаимомиграции исключены (рис. 13), следовательно, не будут возникать коллизии.

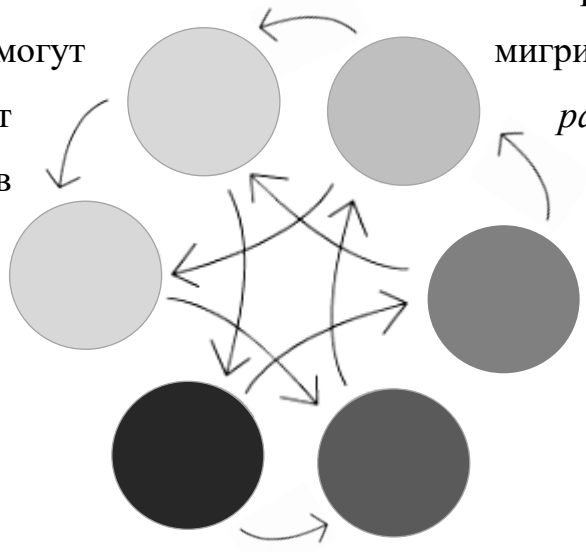


Рисунок 13 - Островная модель генетических алгоритмов

Основным преимуществом островной модели генетических алгоритмов является то, что используя такую модель, можно моделировать различные комбинации способов отбора и формирования следующего поколения или сделать так, что в разных популяциях будут использоваться разные комбинации операторов генетического алгоритма.

3.3. Применение модели генетических алгоритмов с несколькими взаимодействующими популяциями для решения задачи коммивояжера

Для параллельного выполнения генетического алгоритма используем встроенные функции среды MATLAB, позволяющие распараллелить выполнение. Все эти функции входят в панель инструментов ParallelComputingToolbox™ [9]. Данная панель инструментов позволяет использовать два подхода для решения параллельных задач. Первый подход основан на непосредственно процедуре отправки задания jodmanager (планировщик), в инструкциях (m-файле) которой описана последовательность команд, которая будет выполняться рабочими процессами. В этом m-файле помимо основных команд MATLAB могут быть использованы функции MPI для коммуникации между рабочими процессами. Второй подход для решения параллельных задач основан на режиме **pmode**. С помощью этого режима непосредственно из командного окна MATLAB

становится возможным обращение к процессам, просмотр их локальных переменных, обмен данными между ними.

Для включения режима **pmode** пользователю нужно ввести:

>>pmodestartconfnumlabs

гдеconf - имя конкретной конфигурации планировщика, numlabs - количество рабочих процессов, которые должны быть запущены. Для распараллеливания вычислений и создания нескольких популяций воспользуемся функциями, позволяющими моделировать жизнь сразу нескольких взаимодействующих между собой популяций.

Сравним на тех же самых наборах данных эффективность решения задачи коммивояжера с различным числом взаимодействующих популяций.

Для простоты возьмем 4 подпопуляции. Обмен будет происходить каждые 20 поколений, согласно стратегии элитарности будут мигрировать 2 лучшие особи.

Таблица 7 - Сравнение эффективности решения задачи коммивояжера с помощью генетических алгоритмов (с миграциями и без) с эффективными методами решения.

Количество городов	Генетический алгоритм без миграций	Генетический алгоритм с миграциями
5	1,4274	1,4275
10	3,1710	3,17103
15	3,2266	3,2266
20	3,9634	3,9856
50	5,8946	5,8946
100	8,7381	8,7842
200	11,9872	12,0034
400	16,3385	16,3456
800	23,4532	23,5132
1600	33,2398	33,6429
3200	-	47,4307

Анализируя табл. 7 легко увидеть, что точность вычислений несколько ухудшается. Это обусловлено обменом генетической информации между популяциями: появление ложных доминирующих особей в подпопуляциях и вследствие этого попадание в локальные минимумы функции. Сравним так же время выполнения (табл. 8 и рис. 1 4). Как видно из полученных данных, время выполнения уменьшилось, но по-прежнему трудоемкость осталась экспоненциальной.

Таблица 8 - Сравнение времени решения задачи коммивояжера с помощью генетических алгоритмов (с миграциями и без) с эффективными методами решения.

Количество городов	Генетический алгоритм без миграций	Генетический алгоритм с миграциями
5	82	22
10	1061	274
15	1532	382
20	2593	665
50	3812	954
100	6734	1785
200	25719	6241
400	145414	36749
800	480854	134628
1600	1465773	390526
3200	-	1268415

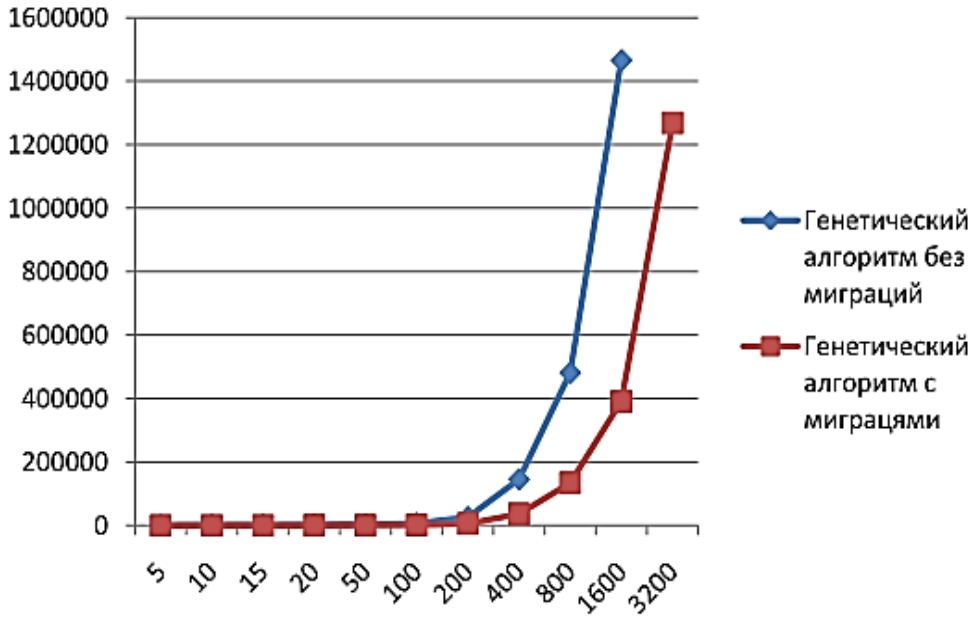


Рисунок 14 - Сравнение зависимость времени выполнения генетического алгоритма с и без миграций.

Для следующего эксперимента возьмем в качестве исходных данных для решения задачи коммивояжера генетическим алгоритмом с взаимодействующими популяциями решения, полученные с помощью минимального остова с и без оптимизации (табл. 9 и табл. 10). Настойки генетического алгоритма оставим прежними.

Таблица 9 - Сравнение точности решения задачи коммивояжера с помощью генетических алгоритмов с несколькими взаимодействующими популяциями с эффективными методами решения. Набор данных - решение задачи, полученное с помощью эффективных способов.

Количество городов	Исходное полученное решение	Решение с помощью минимального остова	Оптимизация решения с помощью минимального остова (k = 3)	Оптимизация решения с помощью минимального остова (k = 5)
5	1,4275	1,4275	1,4275	-
10	3,17103	3,1710	3,1710	3,1710
15	3,2266	3,2265	3,2265	3,2265
20	3,9856	3,4328	3,4328	3,4328
50	5,8946	5,8946	5,8946	5,8946
100	8,7842	7,5376	7,2316	7,2316
200	12,0034	10,1352	9,6252	9,6257
400	16,3456	14,6237	13,4672	13,4672
800	23,5132	19,7432	16,9965	16,9123
1600	33,6429	30,6541	27,4412	27,4412
3200	47,4307	45,8116	41,3276	41,3276
6400	-	62,4731	55,9865	55,9865
10000	-	70,7319	67,0552	67,0552

Таблица 10 - Сравнение времени решения задачи коммивояжера с помощью генетических алгоритмов с несколькими взаимодействующими популяциями с эффективными методами решения (время в мс). Набор данных - решение задачи, полученное с помощью эффективных способов.

Количество городов	Исходное	Решение с помощью минимального остова	Оптимизация решения с помощью минимального Остова (k=3)	Оптимизация решения с помощью минимального остова (k=5)
5	22	0	0	-
10	274	0	0	0
15	382	0	0	0

Таблица 10 - Сравнение времени решения задачи коммивояжера с помощью генетических алгоритмов с несколькими взаимодействующими популяциями с эффективными методами решения (время в мс). Набор данных - решение задачи, полученное с помощью эффективных способов (продолжение).

20	665	15	15	15
50	954	56	56	56
100	1785	193	190	184
200	6241	1534	1315	1141
400	36749	9907	5945	5895
800	134628	25767	21284	20940
1600	390526	55135	66790	63827
3200	1268415	144673	134263	127152
6400	-	594679	375421	351367
10000	-	1332589	931789	904234

Решение так же имеет лучшее качество, время выполнения так же значительно сократилось (рис. 15).

Рисунок 15 - График зависимости времени от числа городов для решения задачи коммивояжера. Набор данных - решение задачи, полученное с помощью эффективных способов.

Сравним так же для 400 городов время выполнения алгоритмов с миграциями и с варьирующимся количеством подпопуляций (рис. 16). Как видно из графика, ощутимый прирост дает уде разделение одной популяции на две, но затем эффективность от увеличения числа подпопуляций падает.

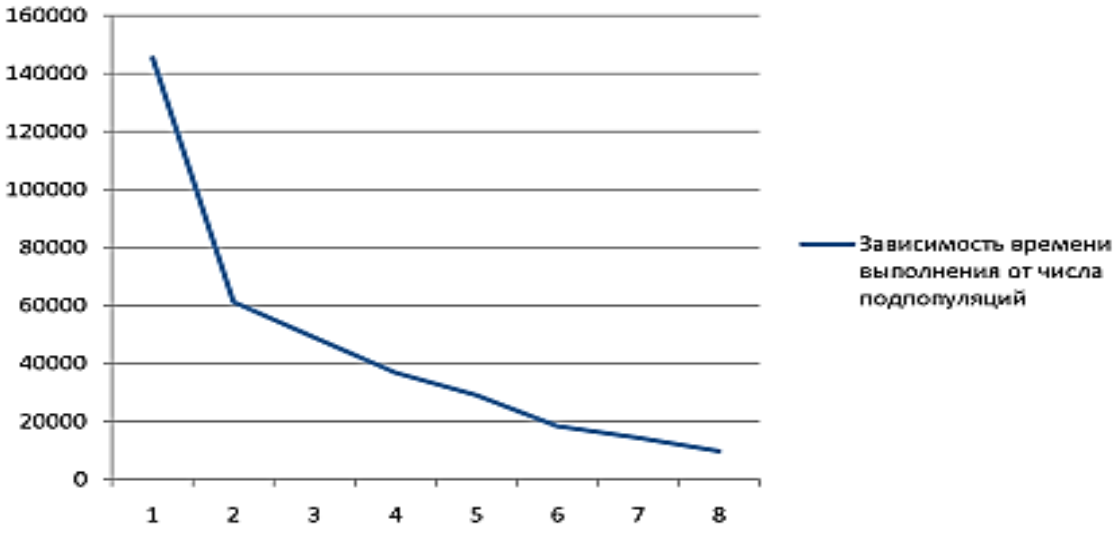


Рисунок 16 - Зависимость времени выполнения генетического алгоритма от числа подпопуляций.

Заключение

В ходе проделанной работы были получены следующие результаты:

1. генетические алгоритмы являются эффективным средством оптимизации, но не самым быстрым;
2. они очень чувствительны к параметрам, а следовательно, сложны в реализации, так как для решения одной задачи нужно перебрать несколько комбинаций параметров, возможно так и не найдя наиболее подходящей;
3. генетические алгоритмы отлично поддаются распараллеливанию, для этого существует ряд моделей, например, островная модель;
4. при распараллеливании генетических алгоритмов их время выполнения снижается, но только лишь на константу. Серьезный прирост дает использование двух, трех, четырех взаимодействующих подпопуляций, однако, с ростом числа подпопуляций эффективность добавление каждой последующей снижается.

На основании проведенного анализа можно сделать выводы о том, что в настоящее время генетические алгоритмы являются мощным вычислительным средством в разнообразных оптимизационных задачах. Но, несмотря на все свои достоинства, генетические алгоритмы имеют множество модификаций и сильно зависят от параметров. Зачастую небольшое изменение одного из них может привести к неожиданному улучшению или ухудшению результата. Следует помнить, что применение генетических алгоритмов полезно лишь в тех случаях, когда для данной задачи нет подходящего специального алгоритма решения.

При использовании генетических алгоритмов рекомендуется, если есть техническая возможность, использовать их параллельную модель, поскольку это даст сокращение времени выполнения.

Список использованных источников и литературы

1. Батищев Д. И., Неймарк Е. А., Старостин Н. В. Применение генетических алгоритмов к решению задач дискретной оптимизации / Д. И. Батищев, Е. А. Неймарк, Н. В. Старостин. - Н. Новгород: 2007. - 85 с.
2. Вороновский Г. К., и др. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г. К. Вороновский. - Х.: Основа, 1997. - 112 с.
3. Генетические алгоритмы [Электронный ресурс] // Генетические алгоритмы и не только. - Электрон. дан. - [б.м.], 2003-2007. - URL: http://qai.narod.ru/GA/ (Дата обращения: 01.06.2010).
4. Генетические алгоритмы: почему они работают? Когда их применять? [Электронный ресурс] // КомптьютерраOnline. - Электрон. дан. - [б.м.], 1997 - 2010. - URL: http://www.computerra.ru/offline/1999/289/2523/ (Дата обращения: 01.06.2010).
5. Мягкие вычисления [Электронный ресурс] // Википедия свободная энциклопедия. - Электрон. дан. - [б.м.], 2001-2010.-URL: http://ru.wikipedia.org/wiki/Мягкие вычисления (Дата обращения: 01.06.2010).
6. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М.

Пилиньский, Л. Рутковский; пер. с польск. И. Д. Рудинского. - М: Горячая линия, 2006. - 452 с.

7. Панченко Т. В. Генетические алгоритмы: учебно-методическое пособие / Т.В. Панченко. - Астрахань: Изд. дом «Астраханский университет», 2007. - 87 с.

8. De Jong, K.A. Introduction to the second special issue on genetic algorithms. / K.A. De Jong. - Machine Learning, 5(4). - p. 351-353.

9. Parallel Computing Toolbox User’s Guide. - MathWorks Inc., 2010. - 665 p.

Приложение А. Кодирование Г рея

Код Грея - непозиционный код с одним набором символов (0 и 1) для каждого разряда. Таким образом, в отличие от римской системы счисления число в коде Грея не является суммой цифр. Чтобы показать соответствие последовательности чисел коду Грея можно воспользоваться таблицей (табл. 11), но есть и наглядное правило построения этой последовательности.

Младший разряд в последовательности чисел в коде Грея принимает значения 0 или 1, затем следующий старший разряд становится единичным и младший разряд принимает свои значения уже в обратном порядке (1, 0). Этим объясняется название кода

- отраженный. Соответственно, два младших разряда принимают значения 00, 01, 11, 10, а затем, при единичном следующем разряде, те же значения в расположены в обратном порядке.

Таблица 11 - Числа в коде Грея и в двоичном коде

Число	Двоичный код	Код Грея
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Алгоритм перевода чисел в коде Грея в позиционный код прост: каждый

разряд в позиционном коде равен сумме по модулю 2 этого и всех более старших разрядов в коде Грея. Старшие разряды, соответственно, совпадают. Перевод из позиционного кода в код Грея так же прост: каждый разряд в коде Грея равен сумме по модулю 2 этого следующего старшего разряда в позиционном коде.