МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯ РФ

Федеральное государственное автономное образовательное
учреждение высшего образования «Пермский
государственный национальный исследовательский
университет»

Кафедра математического
обеспечения вычислительных систем

УДК 519.176+004.421

РАЗРАБОТКА ПОДВИЖНОГО ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА

Курсовая работа по дисциплине
«Моделирование информационных систем»

Работу выполнил студент
группы ПМИ-3 3 курса
механико-математического
факультета
_____Д.О. Сидоренко
Научный руководитель:
канд. техн. наук,
и. о. зав. каф. МОВС
_____А.Ю. Городилов
“___” _____2021 г.

Пермь 2021

АННОТАЦИЯ

Курсовая работа «разработка подвижного генетического алгоритма для решения задачи маршрутизации транспорта» содержит описание подхода с использованием подвижного генетического алгоритма для решения задачи маршрутизации транспорта, а также сравнение данного подхода с другими существующими подходами решения поставленной задачи.

Автор – Д. О. Сидоренко, руководитель – А. Ю. Городилов.

Ключевые слова: маршрутизация транспорта, генетические алгоритмы, хромосомы, эвристические алгоритмы.

Работа содержит 36 страниц, 2 таблицы, 16 рисунков, 13 информационных источников.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время классические генетические алгоритмы утратили свою популярность, на смену пришли их различные модификации. Данная курсовая работа посвящена одной из модификаций – подвижным генетическим алгоритмам.

Основной класс задач, в которых применяется данная модификация, как и классические генетические алгоритмы – это задачи оптимизации. Одной из таких задач является задача маршрутизации транспорта. В данной работе рассматривается классический вариант постановки задачи маршрутизации транспорта в силу того, что упор делается именно на исследование алгоритма.

На российском рынке логистических услуг большая часть от всего объема рынка логистики приходится на транспортную логистику, поскольку на территории России имеется множество автомобильных (841 тыс. км), железных дорог (86 тыс. км) и воздушных путей (800 тыс. км). В связи с этим управление транспортировкой является важнейшим элементом логистики, при этом применение современных подходов позволит снизить общие экономические издержки в среднем на 15–35%, а транспортных расходы – примерно на 25% [1]. Данные расчеты отражают актуальность темы данной курсовой работы и ее практическую применимость.

Стоит также отметить, что статей на тему подвижных генетических алгоритмов не так много, в общем доступе не более 3 публикаций, а публикаций конкретно о применении данного алгоритма для решения задачи маршрутизации транспорта не удалось найти. С другой же стороны, результаты исследований [2] [3] свидетельствуют о преимуществе подвижных генетических алгоритмов над классическими. Таким образом, можно говорить о новизне темы курсовой работы, а также о потенциально высоких результатах разработанного в ходе курсовой алгоритма.

Целью данной работы является разработка подвижного генетического алгоритма для решения задачи маршрутизации транспорта.

Для реализации поставленной цели необходимо решить следующие задачи:

1) осуществить анализ научной литературы, связанной с генетическими алгоритмами и подвижными генетическими алгоритмами,

2) сформулировать постановку задачи маршрутизации транспорта,

3) сформировать целевую функцию для задачи маршрутизации транспорта,

4) определить параметры целевой функции задачи маршрутизации транспорта,

5) выбрать способ кодирования для параметров целевой функции,

6) выделить ключевые составляющие алгоритма,

7) осуществить проектирование алгоритма,

8) реализовать алгоритм,

9) протестировать результаты работы алгоритма,

10) сравнить результаты разработанного алгоритмы с уже существующими аналогами.

В 1 главе данной курсовой работы сформулирована постановка задачи маршрутизации транспорта. Во 2 главе представлен обзор научной литературы, связанной с классическими генетическими алгоритмами и подвижными генетическими алгоритмами. В 3 главе приведены описания двух разработанных подвижных генетических алгоритмов. В 4 главе описывается моделирование и реализация приложения для запуска разработанного алгоритма. В 5 главе проводится тестирование разработанных алгоритмов и анализируются полученные результаты.

1 Постановка задачи

1.1 Описание задачи

Пусть имеется N транспортных средств, находящихся в одной точке (в депо). Имеется M целей, в каждую из которых должен быть доставлен груз. Известна матрица расстояний, в которой указаны все попарны расстояния между целями, а также между целями и депо. Все транспортные средства идентичные и передвигаются с одной фиксированной скоростью, поэтому заданные расстояния соответствуют временам перемещения между целями или между целью и депо. Время для выгрузки товара в каждой из целей считается фиксированной и одинаковой для каждого транспортного средства, кроме того, для выгрузки товара никак не затрачивается топливо, поэтому данным временем можно пренебречь.

Задача заключается в построении оптимальной последовательности целей для каждого транспортного средства, причём каждая цель должна быть посещена хотя бы одним транспортным средством, а транспортные средства после доставки грузов должны вернуться в исходную точку. Критерием оптимальности является минимизация суммарного потраченного топлива.

В данной постановке задаче нет никаких ограничений на транспортные средства, таких как ограничение на грузоподъемность, ограничение на дальность поездки каждого транспортного средства. Данная постановка в силу своей универсальности может быть переработана под любой класс задач маршрутизации транспорта и при этом, изменения в постановке не приведут к значительным изменениям в коде программы: изменится целевая функция и функция подсчета приспособленности каждой особи – данная функция напрямую зависит от целевой; все же остальные элементы программы зависят от параметров целевой функции, которые в рамках всех классов задач остаются неизменными.

. С точки зрения теории алгоритмов, задача является NP–трудной.

1.2 Формулировка в терминах теории графов

Сформулируем условие задачи в терминах теории графов. Пусть

- $V = \{v_0, v_1, \ldots, v_N\}$ – множество вершин, где v_0 – депо, $v_{1..N}$ – цели;
- E – множество рёбер $\{(v_i, v_j) \mid i \neq j\}$;
- C – квадратная матрица расстояний между вершинами, имеющая размерность N+1, где c_{ij} – расстояние между вершинами i и j;
- M – количество имеющихся транспортных средств;
- R_i (i=1..M) – маршрут i-го транспортного средства, представляющий собой последовательность номеров посещённых вершин, причём первое и последнее числа последовательности равны 0, что означает, что маршрут начинается и заканчивается в депо;
- $C(R_i)$ – длина маршрута, равная сумме расстояний между каждой парой соседних вершин в маршруте R_i;

Решением задачи является множество маршрутов R_i (i=1..M), удовлетворяющее условиям:

- Каждая вершина u=1..N, соответствующая цели, входит ровно в один маршрут R_j (доставлять груз несколькими средствами в одну и ту же цель не является оптимальным).

Оптимизируемая величина F вычисляется по следующей формуле ниже :

$$F = \sum_{i=1}^{M} C(R_i). \quad (1)$$

2 Обзор научной литературы

2.1 Генетические алгоритмы

Генетические алгоритмы – это очень популярный способ решения задач оптимизации. В их основе лежит использование эволюционных принципов для поиска оптимального решения [4].

В задачах оптимизации мы имеем некоторую функцию $F(x_1, x_2, \cdots, x_l)$, экстремум (минимум или максимум) которой нужно найти.

При этом функция F называется целевой функцией, а множество $\{x_1, x_2, \cdots, x_l\}$ – параметрами функции.

В терминах классических генетических алгоритмов параметр целевой функции представляется в виде совокупности генов, а вся совокупность параметров – в виде хромосомы. Каждой хромосоме ставится в соответствие особь. Подставив хромосому особи в целевую функцию, можно получить ее значение. То, насколько это значение удовлетворяет поставленным условиям, определяет приспособленность особи – важный параметр, который впоследствии будет использован при отборе особей.

Рассмотрим общую схему ГА [2] [4] [5]. Работа алгоритма начинается с инициализации, на этом этапе создается первое поколение особей. Далее для каждой особи подсчитывается приспособленность. После этого особи сортируются в зависимости от приспособленности каждой особи. Далее происходит создание нового поколения путем использования операторов кроссинговера и мутации. После этого оценивается новое поколение и с помощью критериев остановки принимается решение завершить работу алгоритма или перейти к этапу сортировки новых особей. Блок-схема работы алгоритма приведена на рисунке 1.

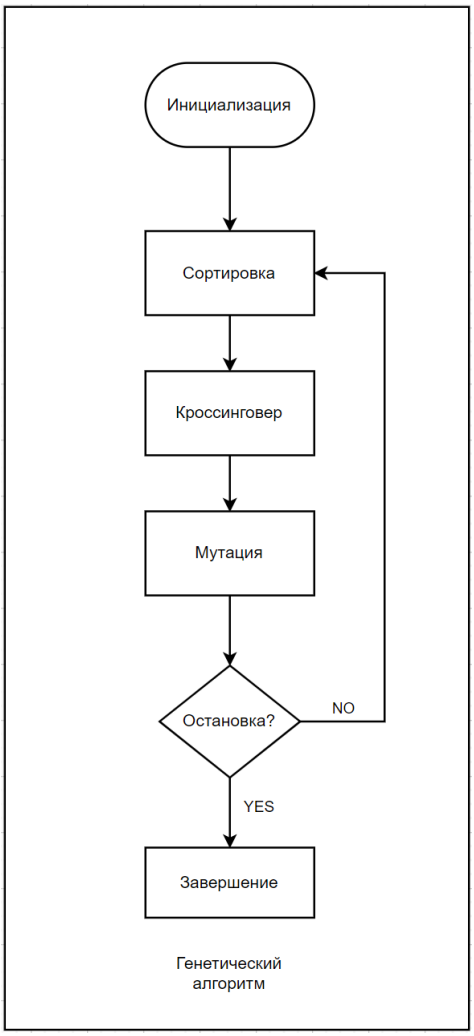

Рисунок 1 – блок-схема работы ГА

Генетические алгоритмы имеют высокую вариативность компонентов. Далее рассмотрим 6 самых важных составляющих.

2.1.1 Представление хромосом

Хромосомы в терминах ГА являются описанием особи в популяции. Представление хромосом определяет структуру алгоритма и то, как будут устроены операторы [6]. Каждая хромосома состоит из последовательности генов из заранее определенного алфавита. В классических ГА обычно используется бинарное кодирование [4], это означает, что каждый ген принимает значение 1 или 0. В то же время часто применяется другой подход, когда в виде генов хранятся реальные значения параметров. Исследования показывают [7], что такой подход представления хромосом в

ГА более эффективен по времени работы и в случае, когда параметры имеют нецелочисленный тип, более точную репликацию (воспроизведение потомства), чем бинарное представление.

То, каким будет представление хромосом, очень сильно зависит от специфики задачи. Выбор остается за разработчиком алгоритма в зависимости от целевой функции, ограничений на параметры функции и других аспектов.

2.1.2 Генетические операторы

Генетические операторы определяют, каким именно будет новое поколение особей, поэтому их выбор в большей степени влияет на то, будет ли новое поколение особей лучше старого и будет ли лучше в принципе.

Существуют 2 основных генетических оператора:

– Оператор кроссинговер:

Кроссинговер принимает 2 хромосомы на входе. Две входные хромосомы обмениваются между собой генами, таким образом создаются одна или две новые хромосом.

Процесс кроссинговера представлен на рисунке 2.

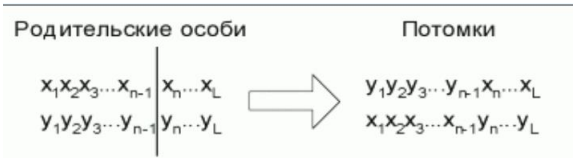

Рисунок 2 – оператор кроссинговер

Данный тип кроссинговера называется одноточечным, так как в нем родительские хромосомы разделяются только в одной случайной точке.

Кроссинговер служит источником появления новых особей.

– Оператор мутации:

Оператор мутации принимает на вход 1 хромосому. Во входной хромосоме случайным образом выбирается ген и инвертируется (в случае двоичной записи генов все очевидно, в других случаях – необходимо

определить функцию инвертирования). Процесс мутации представлен на рисунке 3.

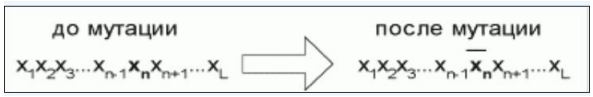

Рисунок 3 – оператор мутации

Как и кроссинговер, мутация может проводится не в одной случайной точке. Можно выбрать некоторое количество точек для инверсии, причем их число также может быть случайным [4].

Вклад оператора мутации заключается в выведении алгоритма из локального оптимума.

2.1.3 Методы отбора нового поколения

ГА представляет из себя итерационный процесс, в котором особи сначала отбираются для скрещивания, потом скрещиваются, затем из их потомков формируется новое поколение и все начинается сначала. Стратегии отбора являются составной частью ГА и определяют "достойных" для скрещивания особей. Ниже рассматриваются несколько наиболее распространенных стратегий [4].

– пропорциональный отбор:

При данном виде отбора сначала подсчитывается приспособленность каждой особи f_i. После этого находится средняя приспособленность в популяции $f_{\text{ср}}$ как среднее арифметическое значений приспособленности всех особей. Затем для каждой особи вычисляется отношение:

$$f_i \Big/ f_{\text{ср}} \qquad (2)$$

Отношение (2) определяет количество скрещиваний, в которых будет принимать участие хромосома с индексом i. Округление значения отношения (2) происходит по следующему правилу: с вероятностью p значение отношения (2) округляется в большую сторону и с вероятностью $(1-p)$ в меньшую, где p – значения дробной части отношения (2).

– турнирный отбор:

При данном виде отбора сначала все особи разделяются на несколько групп. В каждой из групп выбирается некоторое количество наиболее приспособленных особей, которые принимают участие в дальнейшем отборе. Следующие стадии отбора происходит аналогично первой.

2.1.5 Формирование нового поколения и принцип «элитизма»

После скрещивания особей необходимо решить какие из новых особей войдут в следующее поколение, а какие – нет, и что делать с их предками. Есть два варианта. Они отличаются принципом формирования нового поколения. В первом случае, новое поколение будет состоять полностью из особей, получившихся в результате скрещивания или иначе – только из потомков. Во втором случае, новое поколение формируется как из особей потомков, так и из родительских особей. Второй вариант кажется более предпочтительным в силу того, что лучшие родительские особи не заменяются на произвольных потомков, но в таком случае может проявится преждевременная сходимость.

Принцип «элитизма», впервые представленный Кеннетом де Йонгом в 1975 году, является дополнением ко многим методам отбора нового поколения. Суть принципа заключается в сохранении некоторого количества лучших особей в каждом поколении [4]. Такие особи могут быть потеряны, если они не будут выбраны для воспроизводства нового поколения или если они будут уничтожены в результате мутаций. Многие исследования [8] [9] [10] установили, что использование принципа “элитизма” значительно улучшает результаты работы генетического алгоритма.

2.2 Подвижные генетические алгоритмы

Подвижный генетический алгоритм (ПГА) – это, по сути, генетический алгоритм с некоторыми фундаментальными отличиями.

Стоит отметить, что не существуют устоявшегося термина для англоязычного варианта ПГА «Fluid genetic algorithm» на русском языке. Подвижность является наиболее точной характеристикой, отражающей суть данного алгоритма.

2.2.1 Отличия от классических ГА

Главное отличие подвижных генетических алгоритмов от классических генетических алгоритмов заключается в том, что хромосомы и особи в терминах ПГА – это разные сущности. Формально и в обычных генетических алгоритмах они являются разными понятиями, но, тем не менее, между ними существует взаимно однозначное отношение. То есть каждая уникальная хромосома ассоциируется только с одной особью и наоборот. В ПГА используется другой подход и с одной хромосомой может ассоциироваться несколько особей [2].

Вторым существенным отличием является то, что в ПГА нет необходимости в операции мутации. Благодаря новой структуре обеспечивается достаточное разнообразие популяции для того, чтобы вероятность попадания в локальный оптимум стала заметно меньше в сравнении с классическими ГА. Собственно говоря, процедура мутации в ПГА выполняется внутренне (автоматически) [2].

2.2.2 Хромосомы в терминах подвижных ГА

Как уже говорилось ранее, в терминах ПГА смысл хромосом меняется. Теперь они соответствуют предрасположенностью хромосомы к превращению в ту или иную особь.

Пример хромосомы приведен на рисунке 4. Значения в каждой ячейке хромосомы представляют из себя вероятность гена стать равным 1 и 0 в противном случае.

0.41	0.26	0.99	0.21	0.63	0.39	0.85

Рисунок 4 – пример ПГА хромосомы

2.2.3 Функция соответствия

Связь между понятиями хромосома и особь в ПГА осуществляется введением функции соответствия. Входным параметром функции является хромосома. На ее основе функция рассчитывает особь.

Способ работы функции заключается в том, что с вероятностью p_i значение в i-ой клетке особи будет равно 1 и с вероятностью $(1 - p_i)$ будет равно 0, где p_i – это вероятность, посчитанная на основе значения i – ой клетки хромосомы.

Например, первая клетка хромосомы на рисунке 4 имеет значение 0.41, следовательно, вероятность того, что первая клетка полученной особи будет равна единице, составляет 41%, а вероятность того, что она будет равна нулю соответственно 59%.

Кроме того, функция соответствия своим наличием позволяет ввести два новых понятия – глобальная скорость обучения и план поколения. План поколения – это хромосома, значения ячеек которой равны среднему арифметическому всех хромосом по каждой ячейке отдельно. Глобальная скорость обучения – гиперпараметр алгоритма, который определяет, насколько конкретная особь будет похожа на всю популяцию.

На основе этих двух параметров, функция соответствия корректирует входную хромосому, делая ее чуть более похожей на всю остальную популяцию. Происходит это путем пересчета значения каждой клетки хромосомы по формуле (3).

$$\begin{cases} \eta_g \times PVB_i + (1 - \eta_g) \times PVC_i < \eta_{DR} & EPV_i = \eta_{DR} \\ \eta_g \times PVB_i + (1 - \eta_g) \times PVC_i > 1 - \eta_{DR} & EPV_i = 1 - \eta_{DR} \\ otherwise & EPV_i = \eta_g \times PVB_i + (1 - \eta_g) \times PVC_i \end{cases} \quad (3)$$

где η_g – глобальная скорость обучения,

PVC_i – значения вероятности, записанное в i – ой ячейке хромосомы,

PVB_i – значения вероятности, записанное в i – ой ячейке плана поколения,

η_{DR} – коэффициент разнообразия,

EPV_i – значение эффективной вероятности (скорректированной вероятности).

Коэффициент разнообразия $\eta_{DR} \in (0;1)$ в формуле (3) удерживает значение эффективной вероятности в отрезке $[\eta_{DR}; 1 - \eta_{DR}]$. Таким образом, каждая ячейка особи имеет вероятность стать равной и нулем, и единицей.

После подсчета EPV_i для каждой клетки хромосомы, на основе эффективной вероятности расчитывается особь, которая является результатом функции.

2.2.4 Модифицированный оператор кроссинговер

Подобно ГА, оператор кроссинговера имеет входными параметрами две хромосомы, но при этом теперь каждая хромосома идет в связке с особью, которая была рассчитана функцией соответствия для этой хромосомы. В процессе работы оператора хромосомы обмениваются частями для образования новой, как и в классических ГА. В результате получается хромосома, которая также идет в связке с особью, полученной из частей особей входных хромосом. Данная особь необходима для того, чтобы после получения новой хромосомы пересчитать ее значения на основе индивидуальной скорости обучения по следующему правилу (4): если значение ячейки особи равно единице, к значению соответствующей ячейки хромосомы добавиться величина скорости обучения, а если значение ячейки равно нулю, то, наоборот, значение уменьшиться на величину скорости обучения.

$$PVC_i = \begin{cases} \max(PVC_i + \eta_{ind}, 1 - \eta_{DR}), & Ind_i = 1 \\ \min(PVC_i - \eta_{ind}, 1 - \eta_{DR}), & Ind_i = 0 \end{cases} \quad (4)$$

где PVC_i – значения вероятности, записанное в i – ой ячейке хромосомы,

η_{ind} –индивидуальная скорость обучения,

η_{DR} – коэффициент разнообразия,

Ind_i – значение соответствующей особи в ячейке с индексом i.

Данное изменение является главным эволюционным механизмом алгоритма: если особь имеет значение в какой – то ячейке равное единице и при этом имеет относительно высокую приспособленность, чтобы принять участие в создании нового поколения, то надо увеличить вероятность появления единицы в данной ячейке у потомков. Аналогичные рассуждения, если значение в ячейке равно нулю.

Результатом работы алгоритма является пересчитанная на основе индивидуальной скорости обучения хромосома.

На рисунке 5 приведён пример оператора кроссинговера. Индивидуальная скорость обучения для примера равна 0.05.

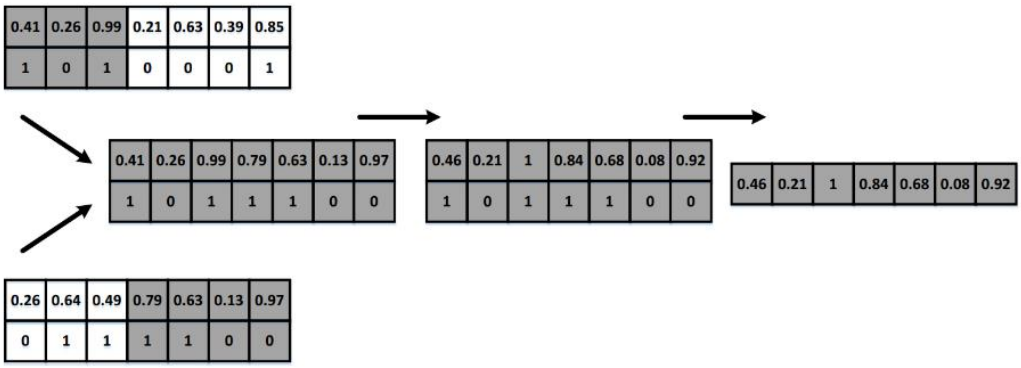

Рисунок 5 – оператор кроссинговер

2.2.5. Итерации эволюции

Подобно ГА, ПГА работает по принципу эволюции, с каждой итерацией приближаясь к оптимальному ответу. Единственное отличие блок-схемы ПГА от блок-схемы ГА, представленных на рис. 6, заключается в том, что ПГА не нуждается в мутации для выхода из локальных оптимумов. Более того, инициализация двух алгоритмов разная. Все хромосомы ПГА изначально будут одинаковыми, и все их клетки будут иметь значение 0.5. План первого поколения будет таким же, как и все хромосомы. Однако, если ПГА запускается для решения проблемы с некоторым пониманием того, каким должен быть оптимальный ответ, можно задать план первого

поколения по–другому, чтобы дать алгоритму лучший старт и более быструю сходимость к глобальному оптимуму [2] [3].

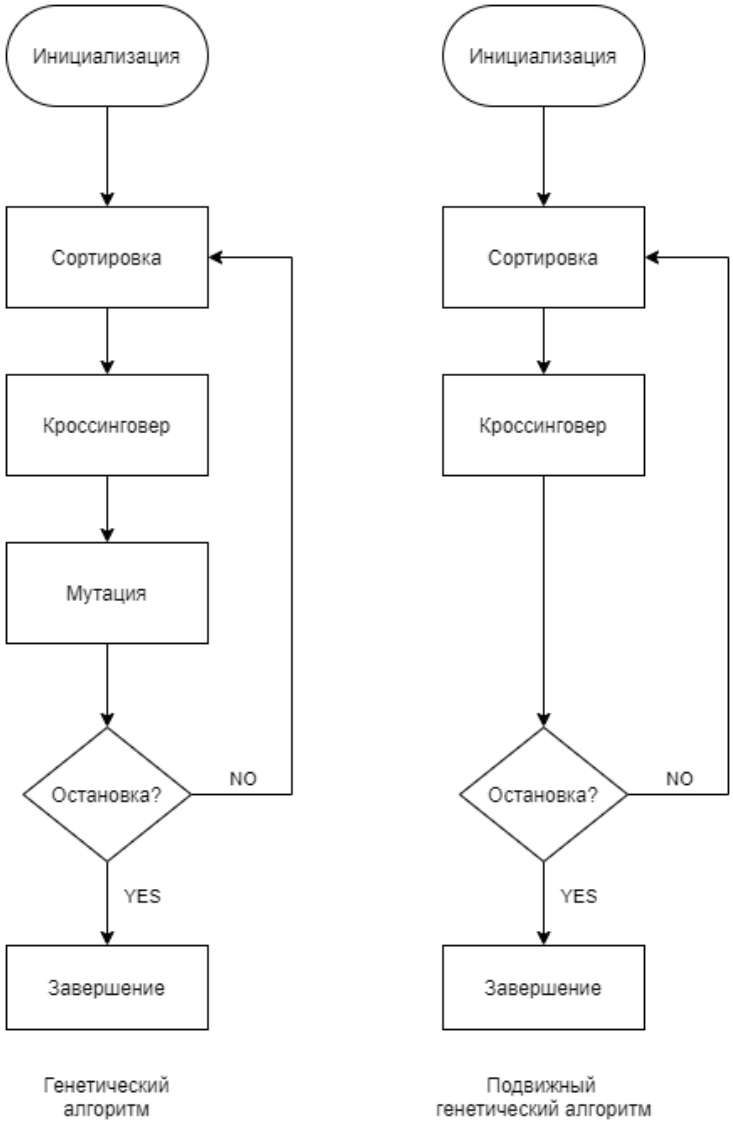

Рисунок 6 – блок-схемы ГА и ПГА

3 Разработка алгоритма

На данном этапе необходимо адаптировать описание подвижного генетического алгоритма под задачу маршрутизации транспорта.

За основу алгоритма был взят подход к классическим ГА, описанный Кристианом Принсом [11] и примененный в выпускной квалификационной работе Александром Цаплиным [12] в 2020 году. Все решения кодируются одной хромосомой, представляющей из себя последовательность вершин в порядке посещения без каких-либо разделителей. Такое хранение является возможным, поскольку по любой такой последовательности всегда можно вычислить оптимальное разбиение на пути, запустив, например, алгоритм Дейкстры на ориентированном ациклическом графе, где вершины – это элементы последовательности, а рёбра соединяют пары вершин меньших индексов с большими. Поскольку есть последовательность вершин в виде перестановки, то алгоритм Дейкстры вырождается в задачу линейного динамического программирования.

Таким образом, особи будут представлять из себя перестановку вершин, по которой однозначно находится оптимальное значение целевой функции за полиномиальное время. Хромосомы же будут предрасположенностью к превращению в ту или иную перестановку.

Приспособленность особи будет зависеть от суммарной дальности проезда. В генетических алгоритмах и не только принято, что чем выше приспособленность особи, тем она лучше [4]. Если значению приспособленности особи присвоить значение суммарной дальности оптимального маршрута, построенного по этой особи, это будет некорректно, так как в данной постановке задачи предпочтительными являются особи с меньшей суммарной дальностью. Таким образом, целесообразно приспособленностью особи считать обратное значение суммарной дальности оптимального маршрута, построенного по этой особи (5) :

$$Fit_i = {}^{1}/_{F_i} \quad (5)$$

где Fit_i – значение приспособленности i – ой особи,

F_i - суммарная дальность оптимального маршрута для i – ой особи.

От того как будет кодироваться последовательность вершин очень сильно зависит структура алгоритма. Далее рассмотрим 2 возможных подхода, ключевым отличием которых и является представление перестановки вершин. Оба подхода были разработаны, реализованы и протестированы в ходе данной курсовой работы.

3.1 Подход с использованием кода перестановки

В первом подходе перестановка представляется в виде кода. Как получить код перестановки удобно рассмотреть на примере. Посчитаем код для перестановки {4, 5, 1, 3, 2}. Будем итерироваться по перестановке последовательно слева направо и для каждого элемента выписывать его позицию в отсортированном массиве из элементов, чьи позиции в перестановке не меньше текущего: 4 в массиве {1, 2, 3, 4, 5} имеет четвертую позицию; 5 в массиве {1, 2, 3, 5} – также четвертую позицию, 1 в массиве {1, 2, 3} – первую позицию, и так далее для всех элементов перестановки. В итоге, для исходной перестановки получим последовательность {4, 4, 1, 2, 1} – это и будет кодом перестановки. У данного кодирования есть 2 явных преимущества. Первое – это простота кодирования и раскодирования.

Второе и главное преимущество заключается в отсутствии необходимости модифицировать оператор кроссинговер: 2 любых кода перестановки можно разделить в любой точке, соединить начало одного кода с концом другого, и в результате всегда получится корректный код перестановки. Перестановка же не имеет такого свойства. Если особь никак не кодировать, то в результате оператора кроссинговера, в котором особи обмениваются своими частями, может получится последовательность из повторяющихся вершин, при этом некоторые вершины могут отсутствовать.

Напомним, что в ПГА в операторе кроссинговера входные хромосомы идут в связке с особями, и частями обмениваются как хромосомы, так и особи. Поэтому обмен частями должен быть возможен и для хромосом, и для особей.

Кроме того, элементы кода перестановки обладают следующим свойством:

$$MaxVal[i] \leq n - i + 1, 1 \leq i \leq n \qquad (6)$$

где n – размер перестановки, i – позиция в коде перестановки, а MaxVal[i] – максимальное значение элемента, стоящего на позиции i.

Хромосома будет представлять из себя набор генов, где ген на позиции i представляется в виде набора вероятностей p_{ij} – вероятность того, что ген с индексом i будет равен значению $(j + 1)$. Свойство (6) позволяет хранить хромосому в виде треугольной матрицы. Пример хромосомы приведен на рисунке 7.

	Гены хромосомы				
Значение гена	X1	X2	X3	X4	X5
1	0.1	0.7	0.3	0.1	1.0
2	0.3	0.1	0.3	0.9	-
3	0.4	0.1	0.4	-	-
4	0.15	0.1	-	-	-
5	0.05	-	-	-	-

Рисунок 7 – пример хромосомы для первого подхода

Механизмом эволюции данного подхода будет пересчет вероятностей появления последовательностей вершин по принципу, описанному в разделе «модифицированный оператор кроссинговер».

Рассмотрим пример, допустим после обмена частями хромосом и особей получилась особь с кодом перестановки $\{a_1, a_2, a_3\}$ и новой

хромосомой. Тогда вероятность первого гена в новой хромосоме иметь значение a_1 увеличится на значение индивидуальной скорости обучения, а вероятности других значений уменьшатся на одинаковую величину так, чтобы в сумме все вероятности были равны 1. Аналогично новая хромосома будет пересчитана для значений a2 и a3. На рисунке 8 приведен пример изменения хромосомы для перестановки {5, 1, 3, 2, 1}, индивидуальная скорость обучения для примера равна 0.06, а до изменения хромосома соответствует хромосоме на рисунке 7.

Значение гена	Гены хромосомы				
	X1	X2	X3	X4	X5
1	0.1 - 0.015	0.7 + 0.06	0.3 - 0.03	0.1 - 0.06	1.0
2	0.3 - 0.015	0.1 - 0.02	0.3 - 0.03	0.9 + 0.06	-
3	0.4 - 0.015	0.1 - 0.02	0.4 + 0.06	-	-
4	0.15 - 0.015	0.1 - 0.02	-	-	-
5	0.05 + 0.06	-	-	-	-

Рисунок 8 – Изменение вероятностей на основе индивидуальной скорости обучения

3.2 Подход без кодирования перестановки

Второй подход построен на том, что особь будет явно представлять перестановку вершин. Гены хромосомы в таком случае будут набором из N вероятностей, где N – количество целей. Таким образом, каждый ген может иметь значение от 1 до N. Пример хромосомы приведен на рисунке 9.

Значение гена	Гены хромосомы				
	X1	X2	X3	X4	X5
1	0.1	0.3	0.15	0.69	0.08
2	0.3	0.13	0.15	0.11	0.32
3	0.4	0.18	0.4	0.09	0.18
4	0.15	0.09	0.2	0.07	0.23
5	0.05	0.4	0.1	0.04	0.19

Рисунок 9 – пример хромосомы для второго подхода

При таком представлении возникает ряд проблем. Во-первых, при использовании первого подхода не было необходимости следить за корректностью перестановки, сейчас такая необходимость возникает. Данная проблема будет решена в функции соответствия. При расчете особи по хромосоме функция соответствия будет хранить список вершин, которые уже использованы, тогда значение текущего гена не может быть равно ни одному значению из этого списка. Из доступных вершин на основе их вероятностей, пересчитанных пропорционально так, чтобы сумма была равна 1, функцией соответствия будет выбрано значение для текущего гена. Таким образом, будет получена корректная перестановка или, другими словами, посчитана корректная особь.

Далее все особи отсортируются в порядке увеличения приспособленности. Для лучших особей будет пересчитаны их хромосомы по принципу того, как это происходит в кроссинговере для ПГА. Для худших особей также будут пересчитаны их хромосомы по аналогии, с той лишь разницей, что мы будем уменьшать вероятность появления худших особей из их хромосом. На рисунке 10 представлена иллюстрация изменения хромосомы, приведённой на рисунке 9, при условии, что по ней была

посчитана особь {3, 1, 4, 2, 5} с низкой относительно других особей приспособленностью, индивидуальная скорость обучения равна = 0.04

Значение гена	Гены хромосомы				
	X1	X2	X3	X4	X5
1	0.1 + 0.01	0.3 - 0.04	0.15 + 0.01	0.69 + 0.01	0.08 + 0.01
2	0.3 + 0.01	0.13 + 0.01	0.15 + 0.01	0.11 - 0.04	0.32 + 0.01
3	0.4 - 0.04	0.18 + 0.01	0.4 + 0.01	0.09 + 0.01	0.18 + 0.01
4	0.15 + 0.01	0.09 + 0.01	0.2 - 0.04	0.07 + 0.01	0.23 + 0.01
5	0.05 + 0.01	0.4 + 0.01	0.1 + 0.01	0.04 + 0.01	0.19 - 0.04

Рисунок 10 – изменение вероятностей для особи с низкой приспособленностью

4 Моделирование системы

4.1 Этап проектирования

Для возможности взаимодействия пользователя с программой необходимо создать консольное приложение. На рисунке 11 приведена диаграмма активностей, описывающая наиболее вероятный вариант использования приложения.

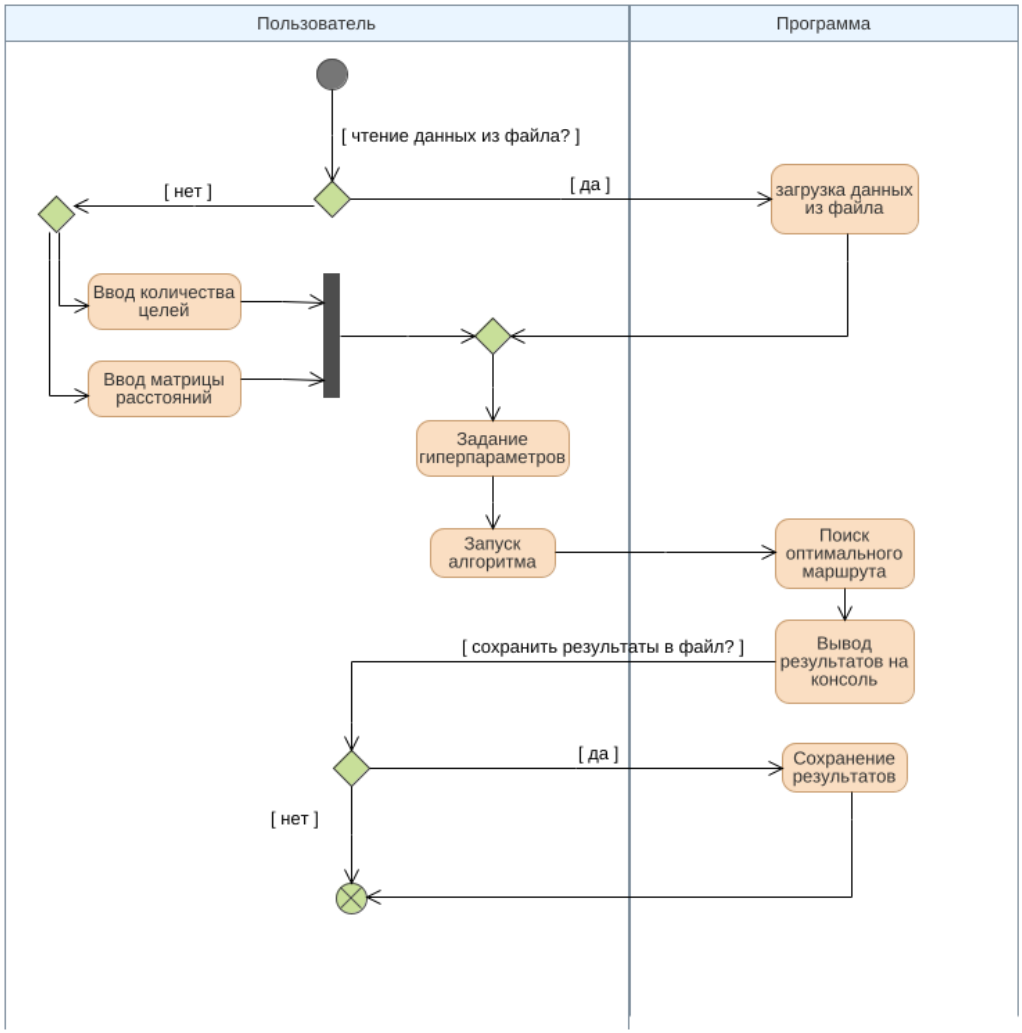

Рисунок 11 – диаграмма активностей

Также на этапе проектирования приложения была построена диаграмма классов, изображённая на рисунке 12. На данной диаграмме представлена структура классов, используемых для решения вычислительной задачи.

Для хранения всех входных данных используется класс InputData. В классы Individual, Chromosome, Gene описывающие такие сущности как особь, хромосома и ген соответственно, вынесены все методы по работе с данными сущностями.

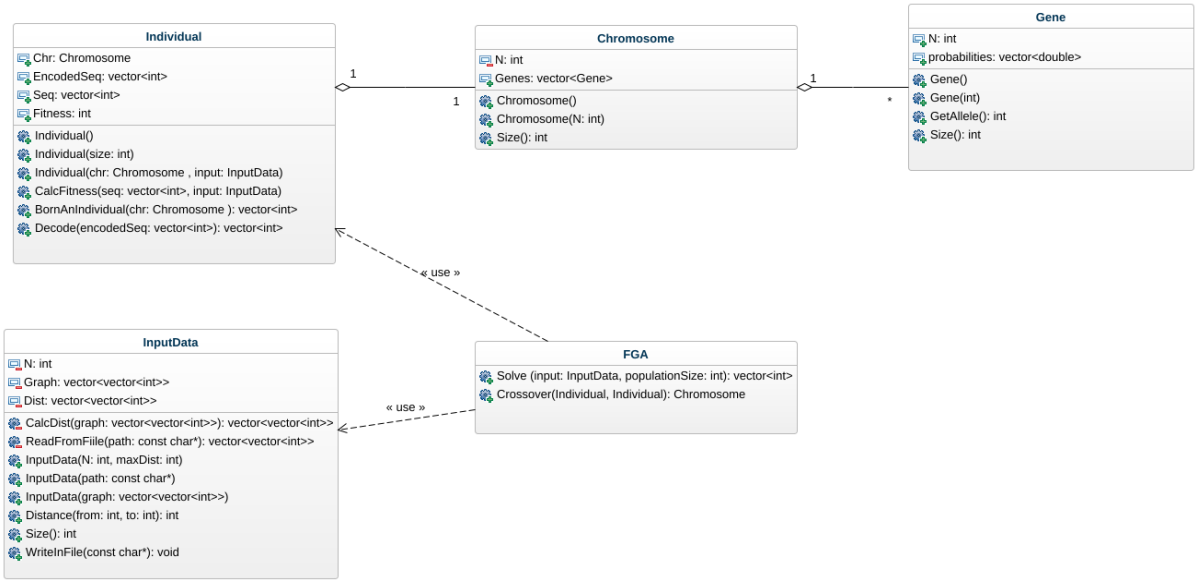

Рисунок 12 – диаграмма классов

4.2 Этап реализации

Для реализации данной программы был выбран язык C++, поскольку он имеет высокую производительность и необходимую функциональность для реализации алгоритма. Всё взаимодействие с пользователем реализовано через консольное приложение. В качестве среды разработки была выбрана среда Visual Studio 2019.

5 Тестирование и результаты

Для проверки корректности алгоритма был реализован алгоритм полного перебора. В силу того, что алгоритм, перебирающий все решения имеет экспоненциальное время работы, то для наборов данных, где n превышает 12 за эталон взят генетический алгоритм из ВКР Александра Цаплина [12].

Тестирование проводилось на 7 тестах, со значениями гиперпараметров:

- Индивидуальная скорость обучения = 0,05
- Глобальная скорость обучения = 0,05
- Коэффициент разнообразия = 0,005

Значения гиперпараметров были подобраны экспериментальным путем. Время работы ГА на каждом тесте было ограничено в 10 секунд, для ПГА минимальное количество итерация для завершения было равно 1000, в случае достижения тысячи итераций время работа ограничивалось десятью секундами. Число особей в популяции после отбора для ПГА было равно 60. Результаты сравнения результатов первого подхода ПГА, точного алгоритма и ГА приведены в таблице 1.

Таблица 1 – Сравнение первого подхода ПГА с другими алгоритмами

Количество целей	Точный	ГА		Первый подход ПГА	
	Сумм. длина	Сумм. длина	Количество итераций	Сумм. длина	Количество итераций
5	2395	2395	4508046	2395	1164
10	2354	2354	3627050	2354	1000
20	-	1416	2835683	4782	1000
40	-	2136	1528882	13753	1000
60	-	2039	872248	19488	1000
80	-	2152	546916	27304	1000
100	-	2125	353726	36993	1000

Из результатов тестирования видно, что ПГА при количестве целей большем двадцати начинает существенно проигрывать в плане оптимальности решения. Далее был проведен ряд тестов, в которых анализировалось поведение ПГА по ходу итераций. На рисунке 13 приведена зависимость между значением приспособленности лучшей особи итерации от номера итерации.

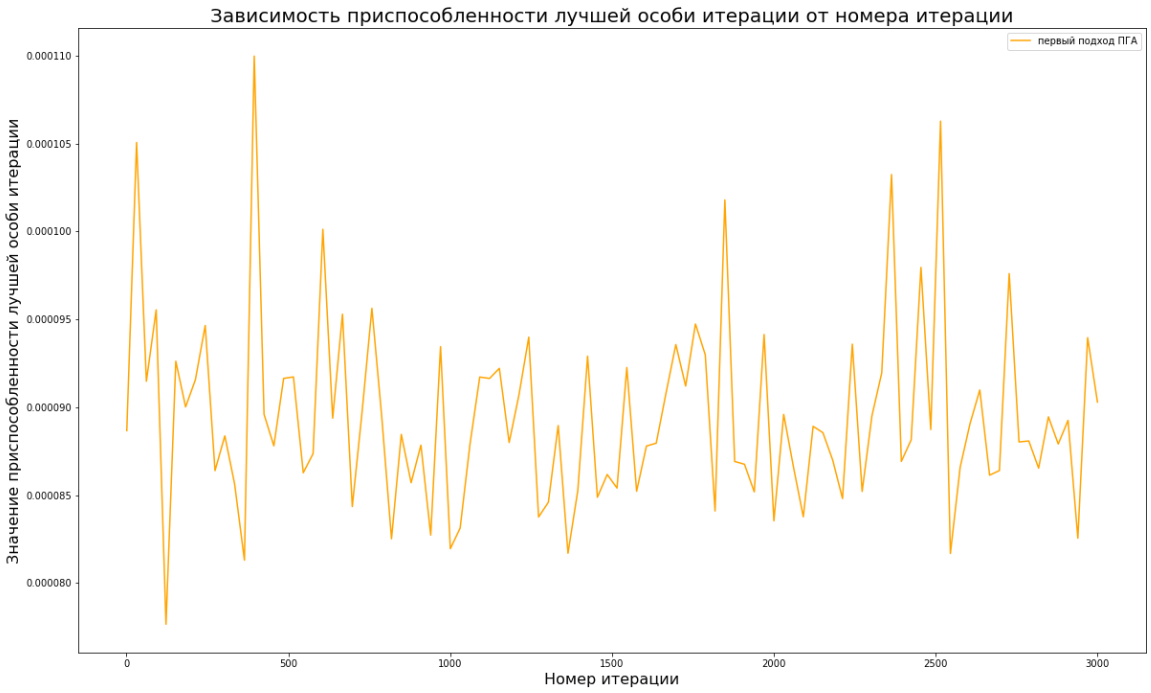

Рисунок 13 – зависимость приспособленности лучшей особи итерации от номера итерации для первого подхода

Рисунок 13 показывает, что зависимости между значением приспособленности лучшей особи итерации от номера итерации почти нет, хотя это зависимость должна быть, в этом заключается суть алгоритма – с увеличением номера итерации, приспособленность лучшей особи должна становится лучше. Кроме того, можно заметить сильные перепады графика, после итераций с высокой приспособленностью лучшей особи сразу наступают итераций с очень низкой приспособленностью.

Показателем сходимости алгоритма к локальному оптимуму является увеличение общей приспособленности, которая вычисляется как среднее

арифметическое приспособленностей всех особей популяции. На рисунке 14 приведена зависимость общей приспособленности от номера итерации для первого подхода.

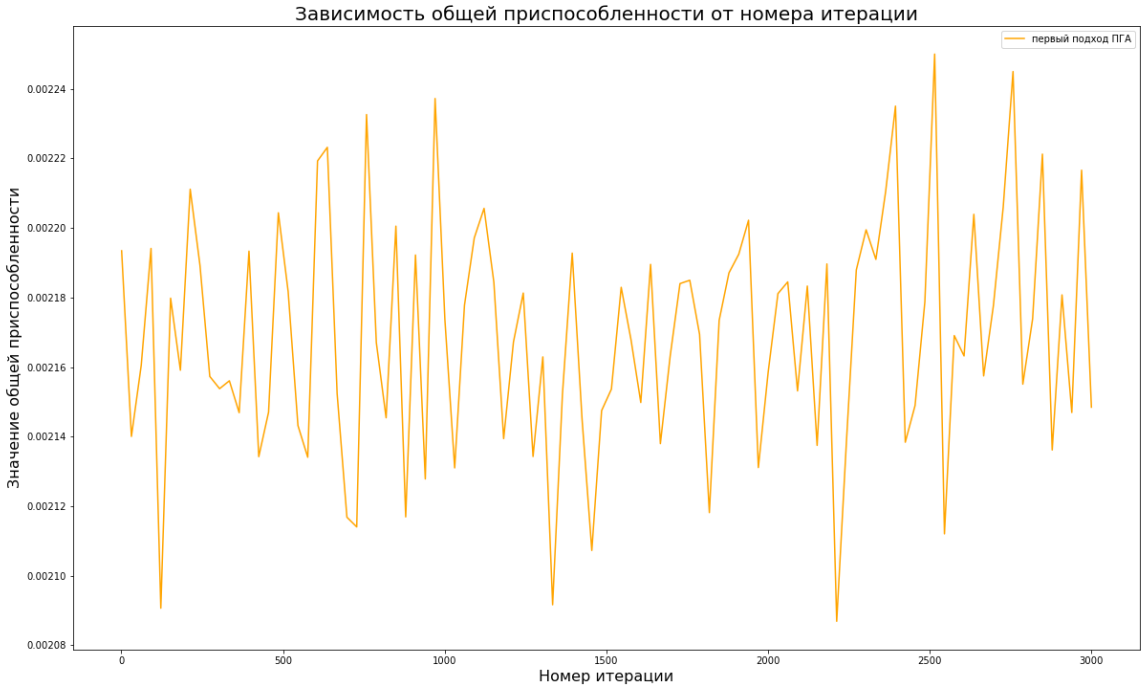

Рисунок 14 – зависимость общей приспособленности от номера итерации для первого подхода

На рисунке 14 видны сильные колебания, алгоритм «стоит на месте», и общая приспособленность не улучшается. Это является красноречивым показателем того, что алгоритм не движется к оптимальному значению целевой функции.

По результатам анализа сходимости графиков стало очевидно, что алгоритм не сходится даже к локальному оптимуму. Была проанализирована структура алгоритма и найден существенный недочет. Рассмотрим пример из двух перестановок {4, 2, 1, 3} и {1, 3, 2, 4}. Код первой перестановки равен {4, 2, 1, 1}, второй – {1, 2, 1, 1}. Несмотря на то, что перестановки не совпадают полностью, у них почти одинаковые коды. Продолжая рассуждения, если предположить, что одна из перестановок имеет высокую приспособленность, а другая низкую, увеличением вероятности появления

первой перестановки, мы также увеличиваем вероятность появления второй, что является ошибкой и не может приводить к верному ответу. Совпадение результатов на маленьком наборе данных является следствием небольшого числа возможных вариантов и большим количеством итераций, другими словами, вероятность попасть в один из 5 возможных вариантов имея при этом 1000 попыток очень велика, но это не является сходимостью к оптимуму.

Для второго подхода точно также как и для первого было проведено тестирование на 7 тестах с аналогичными условиями тестирования. Результаты тестирования приведены в таблице 2.

Таблица 2 – Сравнение второго подхода ПГА с другими алгоритмами

Количество целей	Точный	ГА		ПГА	
	Сумм. длина	Сумм. длина	Количество итераций	Сумм. длина	Количество итераций
5	2395	2395	4508046	2395	3404
10	2354	2354	3627050	2354	1000
20	-	1342	2651739	3172	1000
40	-	2028	1412832	10552	1000
60	-	1939	813539	17833	1000
80	-	1816	522085	25363	1000
100	-	2485	347037	33556	1000

Результаты второго подхода ПГА оказались на 10% лучше первого подхода, но все еще сохранилась существенная разница в сравнении с классическим ГА.

Для второго подхода аналогично первому были построены графики зависимости приспособленности лучшей особи итерации и зависимости

общей приспособленности популяции от номера итерации, графики представлены на рисунках 15 и 16 соответственно.

Они свидетельствуют о движении алгоритма к локальному оптимуму, но происходит это слишком медленно. И правда, тысяча итераций эволюции в сравнении с пятьюстами итерациями эволюции ГА кажутся разумным объяснением сильной разницы оптимальных решений, найденных алгоритмами.

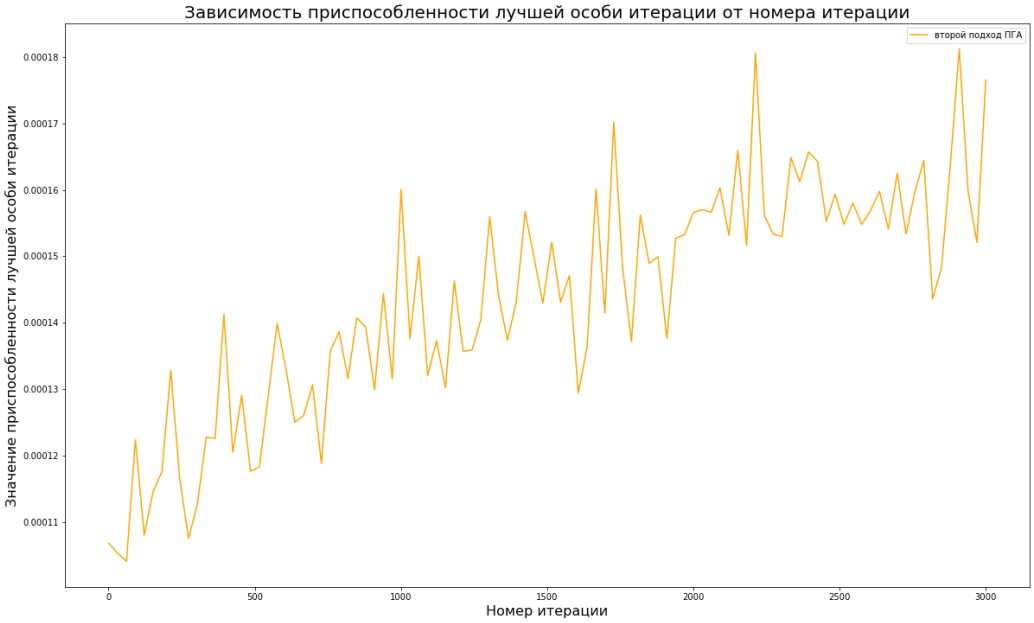

Рисунок 15 – зависимость приспособленности лучшей особи итерации от номера итерации для второго подхода

Рисунок 16 – зависимость общей приспособленности от номера итерации для второго подхода

На графиках четко прослеживается увеличение и приспособленности лучшей особи, и улучшение общей приспособленности популяции в зависимости от числа итераций. Это означает, что алгоритм движется в направлении оптимального решения. На графиках нет заметных перепадов, что свидетельствует о том, что пересчет вероятностей производится правильно.

В отличие от первого подхода, в данном подходе к представлению последовательности вершин не было найдено существенных недостатков, кроме больших временных затрат на совершение одной итерации эволюции (1 итерация при количестве целей порядка 100 выполняется примерно полсекунды).

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы были решены 9 задач:

1) проанализирована научная литература, связанная с генетическими алгоритмами и подвижными генетическими алгоритмами;

2) сформулированы постановка задачи маршрутизации транспорта и ее целевая функция;

3) определены параметры целевой функции задачи маршрутизации транспорта в вид перестановки целей;

4) выбрано несколько способов представления параметров целевой функции: с использованием кода перестановки и явное хранение перестановки;

5) осуществлено проектирование алгоритма;

6) реализован алгоритм на языке C++;

7) реализован точный алгоритм и адаптирован генетический алгоритм для сравнения результатов разработанного алгоритма;

8) протестированы результаты работы для двух подходов алгоритма;

9) сформулированы выводы по итогам тестирования алгоритма:

– первый подход имеет явные недостатки и не движется к оптимальному решению,

– второй подход не имеет явных недостатков, движется к оптимальному решению, но делает это слишком медленно в силу сложности своей структуры.

Таким образом, цель работы, заключающаяся в разработке подвижного генетического алгоритма для решения задачи маршрутизации транспорта, была достигнута.

Тем не менее, разработанный подвижный генетический алгоритм заметно уступает классическому ГА в поиске оптимального решения. Это можно объяснить популярностью использования ГА для решения задачи маршрутизации транспорта и наличием оптимизаций, которые сильно

влияют на эффективность результатов генетического алгоритма. В будущем предполагается попытаться применить уже существующие оптимизации генетических алгоритмов для ПГА. Кроме того, подходы к представлению перестановки вершин, описанные в данной курсовой работе, не являются оптимальными в силу того, что занимают много памяти и требуют большого количества временных ресурсов, что не позволяет выполнить достаточное количество итераций эволюции.

Созданное приложение находится в открытом доступе. Исходный код можно скачать из репозитория ресурса Github [13].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гончарова Ю.А. Задачи маршрутизации при транспортировке: обзор моделей, методов и алгоритмов // Логистика и управление цепями поставок, 2019, № 4, 74–88.

2. *Jafari-Marandi R., Smith B. K.* Fluid Genetic Algorithm (FGA) // Journal of Computational Design and Engineering, V. 4, 2017, P. 158–167.

3. *Hong H.* Flood susceptibility assessment in Hengfeng area coupling adaptive neuro-fuzzy inference system with genetic algorithm and differential evolution // Science of the Total Environment, V. 621, 2018, P. 1124–1141.

4. *Mitchell M.* An Introduction to Genetic Algorithms // Fifth printing, 1999.

5. *McCall J.* Genetic algorithms for modelling and optimization // Journal of Computational and Applied Mathematics, V. 184, 2005, P. 205–222.

6. *Garzelli A., Capobianco L., Nencini F.* Fusion of multispectral and panchromatic images as an optimisation problem // Algorithms and Application, 2008, P. 223 – 250.

7. *Michalewicz Z.* Genetic Algorithms + Data Structures. Evolution Programs, Springer-Verlag, New York, 1994.

8. *Bhandari D., Murthy C. A., Pal S. K.* Genetic algorithm with elitist model and its convergence // International Journal of Pattern Recognition and Artificial Intelligence, V. 10, 1996, P. 731–747.

9. *Jayaram M. A., Nataraja M. C., Ravikumar C. N.* Elitist Genetic Algorithm Models: Optimization of High Performance Concrete Mixes // Materials and Manufacturing Processes, V. 24, 2009, P. 225–229.

10. *Du H., Wang Z., Zhan W., Guo J.* Elitism and Distance Strategy for Selection of Evolutionary Algorithms. IEEE Access, V. 6, 2018, P. 44531–44541.

11. *Prins C.* A simple and effective evolutionary algorithm for the vehicle routing problem // Computers & Operations Research 31, 2004. P. 1985-2002.

12. Capacitated Vehicle Routing Problem (3rd year course work) [Электронный ресурс] [Режим доступа: https://github.com/Alexandr-TS/CVRP].

13. Using FGA to solve Vehicle Routing Problem (3rd year course work) [Электронный ресурс] [Режим доступа: https://github.com/DimaSidorenko/VPR.Fluid_Genetic_algorithm].