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 АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА НА БАЗЕ ПОДВИЖНОГО ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Выпускная квалификационная работа

Работу выполнил студент
группы ПМИ-3-18 4 курса
механико-математического
факультета
_____Д.О. Сидоренко
Научный руководитель:
канд. техн. наук,
и. о. зав. каф. МОВС
_____А.Ю. Городилов
“___” _____2022 г.

Пермь 2022

АННОТАЦИЯ

Выпускная квалификационная работа «Разработка алгоритмов решения задачи маршрутизации транспорта на базе подвижного генетического алгоритма» содержит описание известных подходов к решению задач маршрутизации транспорта, описание алгоритмов с использованием подвижного генетического алгоритма для решения задачи маршрутизации транспорта, а также сравнение данных алгоритмов.

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

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

Работа содержит 47 страниц, 2 таблицы, 18 рисунков, 21 информационный источник.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время 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 транспортные расходы – примерно на 25% [1]. Одной из ключевых задач транспортной логистики является задача маршрутизации транспорта, исследованию подходов к решению которой и посвящена данная работа. Актуальность выпускной квалификационной работы объясняется актуальностью задачи маршрутизации транспорта.

Существует множество алгоритмов, на базе которых построены решения задачи маршрутизации транспорта. В частности, задача может быть решена с использованием генетического алгоритма (далее ГА). В данной же работе, будет создано решение на базе модификации генетического алгоритма – подвижного генетического алгоритма (далее ПГА).

Результаты исследований о использовании ПГА [2] [3] свидетельствуют о преимуществах подвижных генетических алгоритмов над классическими на определенных наборах входных данных. Но в рамках всех этих исследований рассматриваются другие классы задач. Для адаптации алгоритма под данный тип задачи необходимо пересмотреть структуру элементов ПГА, предложив новые алгоритмы.

ПГА обладает рядом параметров, которые позволяют управлять ходом движения к оптимальному решению, эти параметры в рамках данной работы будем называть гиперпараметрами. Оптимальные значения гиперпараметров также предстоит найти в рамках данной работы.

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

Для достижения поставленной цели необходимо решить следующие задачи:

1) осуществить анализ научной литературы, связанной с генетическими алгоритмами и подвижными генетическими алгоритмами;
2) осуществить анализ научной литературы, связанной с вариациями и возможными ограничениями для задачи маршрутизации транспорта;
3) осуществить анализ научной литературы, связанной с существующими подходами для решения задачи маршрутизации транспорта;
4) предложить новые алгоритмы на базе подвижного генетического алгоритма для решения задачи маршрутизации транспорта;
5) реализовать на высокоуровневом языке программирования алгоритмы на базе подвижного генетического алгоритма для решения задачи маршрутизации транспорта;
6) выделить гиперпараметры ПГА;
7) разработать инструмент для поиска оптимального набора гиперпараметров ПГА для определенных групп входных данных;
8) сравнить эффективность работы разработанных алгоритмов между собой и с эталонным решением.

В 1 главе данной работы сформулирована постановка задачи маршрутизации транспорта, и описаны возможные ограничения в задаче маршрутизации транспорта. Во 2 главе представлен обзор научной литературы, связанной с описанием классического ГА и ПГА, а также – литературы, связанной с алгоритмами решения задачи маршрутизации транспорта. В 3 главе приведен процесс разработки алгоритмов на базе

подвижного генетического алгоритма. В 4 главе описан процесс генерации входных данных для отладки и тестирования алгоритмов, выделены основные гиперпараметры ПГА и описан разработанный модуль настройки гиперпараметров, представлен поиск оптимальных гиперпараметров, приведены результаты тестирования разработанных алгоритмов вместе с анализом этих результатов.

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 попарные 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 фиксированным и одинаковым 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, и каждое из них после доставки грузов должно вернуться в исходную точку. Возможны 2 критерия оптимальности:

- минимизация суммарного потраченного топлива (равносильна минимизации суммарного расстояния, пройденного всеми транспортными средствами);
- минимизация общего времени доставки всех грузов (равносильна минимизации максимального расстояния, пройденного одним транспортным средством).

Данные критерии являются зависимыми друг от друга в том плане, что довольно часто при минимизации одного критерия невозможно добиться оптимального значения другого. Таким образом, на практике выбирается только один из критериев.

На практике наиболее часто данная задача встречается с некоторыми из следующих ограничений:

– товары требуется доставить в момент времени, попадающий во временной промежуток, указанный клиентом;
– цель может быть посещена не всеми транспортными средствами, а только каким–то транспортным средством из подмножества допустимых. Это может быть связано с ценностью груза или с его габаритами;
– ограничение на грузоподъемность одного транспортного средства.

В рамках данной работы будут рассмотрены две вариации задачи маршрутизации транспорта. Обе вариации с точки зрения теории алгоритмов, являются NP–трудными.

В первой постановке задачи имеется всего одно транспортное средство, изначально находящееся в депо. Требуется построить оптимальную последовательность целей для транспортного средства, причём каждая из целей должна быть посещена. Критерием оптимальности является минимизация суммарного пройденного расстояния (суммарного потраченного топлива). В данной постановке задачи нет никаких ограничений на транспортные средства, нет ограничений на грузоподъемность, ограничений на дальность поездки транспортного средства. Данная постановка в силу своей простоты будет рассмотрена первой, а далее на основе ее решения будет предложено решение второй постановки, уже с некоторыми ограничениями.

В рамках второй постановки задачи также рассматривается одно транспортное средство. При этом на транспортное средство накладывается условие на грузоподъемность, что тоже самое что и ограничение на количество посещенных целей без возвращения в депо. Кроме того, у каждой цели есть временное окно, в которое оно должно быть посещено. Требуется построить оптимальную последовательность целей для транспортного средства так, чтобы максимизировать количество посещенных целей, а при

равенстве посещенных целей дополнительным условием является минимизация пройденного расстояния.

Под временным окном для цели понимается промежуток времени, в течение которого может быть доставлен товар в конкретную цель. В рамках второй постановки задачи транспортное средство не ожидает начала временного окна, что означает невозможность посещения цели, если момент приезда транспортного средства не попадает в её временное окно.

Так как транспортное средство движется с одинаковой скоростью и временем выгрузки товара в данной задаче можно пренебречь, то границы временного окна могут быть всегда пересчитаны в пройденное расстояние. В данной работе, чтобы не привязываться к мерам измерения, будем считать, что за одну единицу времени транспортное средство преодолевает одну единицу дистанции.

1.2 Формулировка в терминах теории графов

Сформулируем условие задачи в терминах теории графов. Пусть

- N – количество целей;
- $V = \{v_0, v_1, \ldots, v_N\}$ – множество вершин, где v_0 – депо, $v_{1..N}$ – цели;
- E – множество рёбер $\{(v_i, v_j) \mid i \neq j\}$;
- (tl_i, tr_i), i=1..N – временные окна целей;
- C – квадратная матрица расстояний между вершинами, имеющая размерность N+1, где c_{ij} – расстояние между вершинами i и j;
- M – количество маршрутов;
- R_i (i=1..M) – i-ий маршрут 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;

– $D(R_i)$ – количество посещенных вершин в маршруте R_i;

– *maxSize* – ограничение на количество целей в одном маршруте.

Решением первой постановки задачи является маршрут R, удовлетворяющий условиям:

– Каждая вершина u=1..N, соответствующая цели, входит в маршрут R и только один раз;

– Оптимизируемая величина F вычисляется по следующей формуле:

$$F_1 = C(R). \quad (1)$$

Решением второй постановки задачи является набор маршрутов R_i удовлетворяющий условиям:

– $D(R_i) \leq maxSize$, i=1..M (количество целей в маршруте не может превышать соответствующее максимальное допустимое значение);

– Каждая вершина u=1..N, соответствующая цели, входит максимум в один маршрут R_i (доставлять груз несколько раз в одну и ту же цель не является оптимальным);

– Соблюдается условие временных окон. Формально: пусть $time_{ij}$ – время посещения вершины R_{ij}. Тогда $tl[R_{ij}] <= time_{ij} <= tr[R_{ij}]$.

– Основная оптимизируемая величина F_2 вычисляется по формуле:

$$F_2 = \sum_{i=1}^{M} D(R_i). \quad (2)$$

– Дополнительная оптимизируемая величина F_2' вычисляется по формуле:

$$F_2' = \sum_{i=1}^{M} C(R_i). \quad (3)$$

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.

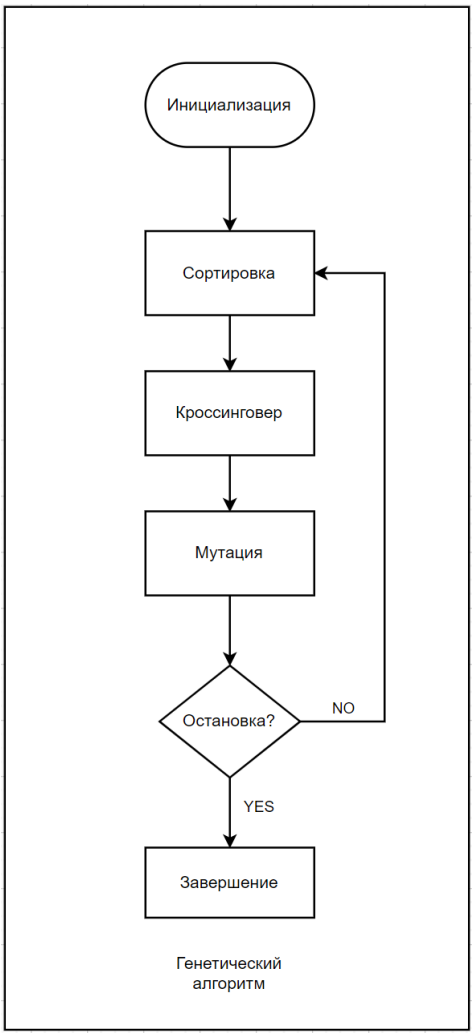

Рисунок 1 – блок-схема работы ГА

Генетические алгоритмы имеют высокую вариативность компонентов. Далее рассмотрим шесть 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.

Существуют 2 основных генетических оператора:

– оператор кроссинговер:

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

Процесс кроссинговера представлен на рисунке 2.

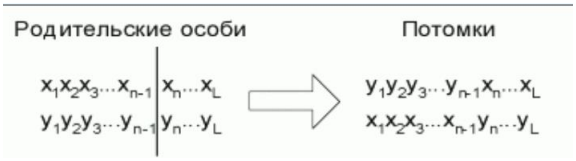

Рисунок 2 – оператор кроссинговер

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 одну 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.

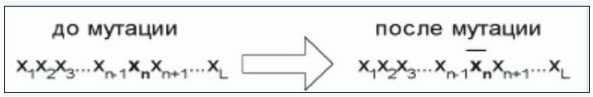

Рисунок 3 – оператор мутации

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 (4):

$$f_i \Big/ f_{\text{ср}} \tag{4}$$

Отношение (4) определяет количество скрещиваний, в которых будет принимать участие хромосома с индексом i. Округление значения отношения (4) происходит по следующему правилу: с вероятностью p значение отношения (4) округляется в большую сторону и с вероятностью $(1-p)$ в меньшую, где p – значения дробной части отношения (4).

– 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 происходят аналогично первой.

2.1.4 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]. 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 предрасположенности хромосомы к превращению в ту или иную особь.

Пример хромосомы приведен на рисунке 4. Значение p в каждой ячейке хромосомы означает, что соответствующий ген с вероятностью p станет равным 1 и с вероятностью (1 - p) станет равным 0.

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.

Принцип работы функции можно разбить на два этапа.

На первом этапе, происходит расчет эффективной вероятности на основе трех параметров:

– 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;
– коэффициент разнообразия – гиперпараметр алгоритма, который удерживает значение эффективной вероятности в интервале от (0;1). Таким образом, каждая ячейка особи может стать равна и нулю, и единице с определенной вероятностью.

Расчет значения каждой клетки хромосомы происходит по формуле (5):

$$\begin{cases} \eta_g \times PVB_i + (1-\eta_g) \times PVC_i < \eta_{DR} & EPV_i = \eta_{DR} \\ \eta_g \times PVB_i + (1-\eta_g) \times PVC_i > 1-\eta_{DR} & EPV_i = 1-\eta_{DR} \\ otherwise & EPV_i = \eta_g \times PVB_i + (1-\eta_g) \times PVC_i \end{cases} \quad (5)$$

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).

На втором этапе, на основе подсчитанной эффективной вероятности EPV_i расчитывается особь по следующему правилу: с вероятностью EPV_i значение в i–ой клетке особи будет равно 1, с вероятностью $(1-EPV_i)$ будет равно 0.

Например, первая клетка хромосомы на рисунке 4 имеет значение 0.41, следовательно, вероятность того, что значение на первой позиции

полученной особи будет равно единице, составляет 41%, а вероятность того, что оно будет равно нулю соответственно 59%.

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 (6): 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.

$$PVC_i = \begin{cases} \max(PVC_i + \eta_{ind}, 1 - \eta_{DR}), & Ind_i = 1 \\ \min(PVC_i - \eta_{ind}, \eta_{DR}), & Ind_i = 0 \end{cases} \quad (6)$$

, 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.

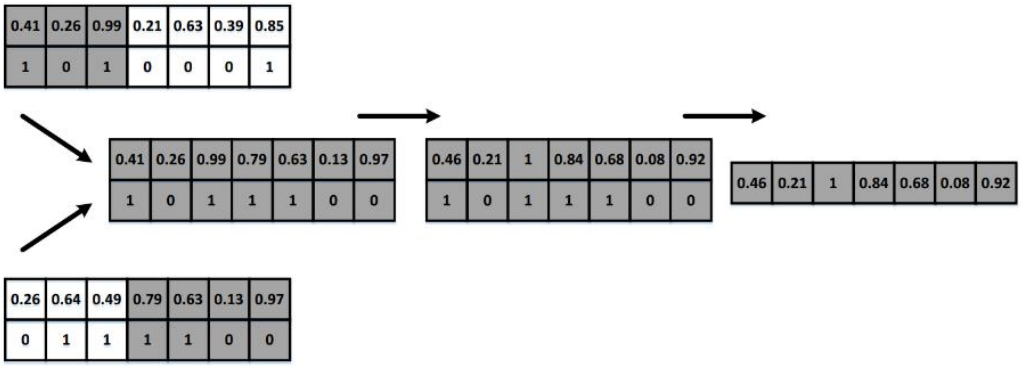

Рисунок 5 – оператор кроссинговер

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. Кроме 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].

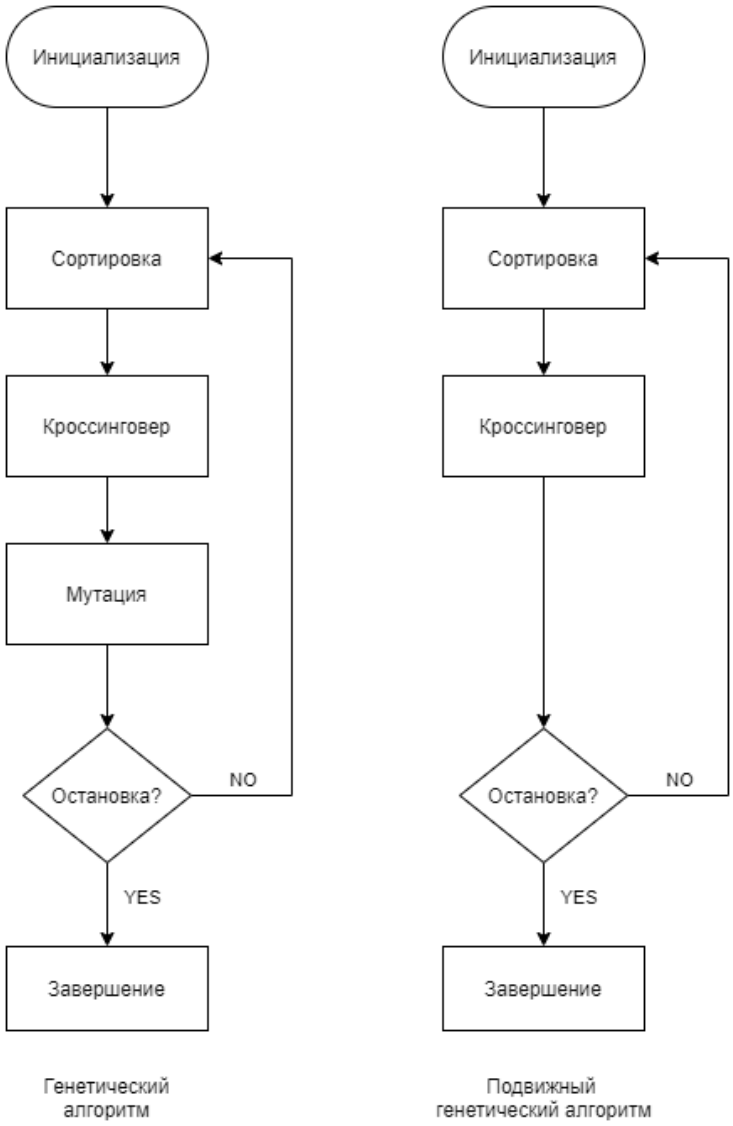

Рисунок 6 – блок-схемы ГА и ПГА

2.3 Алгоритмы для решения задачи маршрутизации транспорта

Помимо двух генетических алгоритмов, описанных выше, существует множество других подходов для решения поставленной задачи. Далее рассмотрим наиболее известные из этих подходов.

Метод ветвей и границ. Алгоритм является точным, то есть результат алгоритма совпадает с точным решением. В основе метода ветвей и границ лежит идея последовательного разбиения множества допустимых решений на подмножества. Впервые в применении к данной задаче данный подход был предложен Фишером в 1994 году [11]. Главным недостатком алгоритма является сложность в определении целевой функции (критериев отсечения) так, чтобы алгоритм работал эффективно по сравнению с другими методами.

Алгоритм сбережений Кларка-Райта. Алгоритм является приближенным. Данный алгоритм считается достаточно простым и эффективным и с точки зрения времени выполнения алгоритма, и с точки зрения результата работы. На первом этапе алгоритма строятся пути от депо до каждой из вершин и обратно. Затем строится матрица сбережений, отображающая насколько уменьшится суммарная длина путей при объединении двух вершин в один путь. Сбережения для двух вершин считаются по формуле (7):

$$s_{ij} = c_{i0} + c_{0j} - c_{ij} \quad (7)$$

, где s_{ij} – элемент матрицы сбережений, c_{ij} – расстояние между вершинами i и j. Далее крайние вершины на путях соединяются, объединяя пути. Порядок этого соединения задаётся жадным алгоритмом: соединяются пары вершин с максимальным значением из матрицы сбережений. Алгоритм предложен Кларком и Райтом в 1964 году [12].

Муравьиные алгоритмы относятся к эвристическим алгоритмам. Идея муравьиного алгоритма – моделирование поведения муравьёв, связанного с их способностью быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям, находя новый

кратчайший путь. На первом этапе муравьи отправляются случайными маршрутами, на последующих – пропорционально содержанию феромонов. Поскольку на более коротких путях остаётся больше феромонов, в итоге алгоритм сходится к достаточно оптимальному решению. [13]. При этом для алгоритма гарантируется сходимость к оптимальному решению, однако скорость сходимости может быть достаточно медленной [14].

Метод имитации отжига. Еще один эвристический алгоритм. Основывается на имитации физического процесса, который происходит при кристаллизации вещества, в том числе при отжиге металлов. Алгоритм итеративно строит решения. На каждом шаге происходит случайное изменение текущего решения. Новое решение принимается и заменяет старое решение с определенной вероятностью, которая зависит от превосходства одного решения над другим и от некоторой убывающей в течение работы алгоритмы функции. Таким образом, алгоритм сходится на некотором итоговом решении. [15]. Применение данного алгоритма было описано американским ученым Османом в 1993 году [16].

Двухфазные алгоритмы. Алгоритмы, в которых сначала выполняется кластеризация. На первом этапе вершины разбиваются на компактные кластеры. На втором этапе для каждого кластера решается задача коммивояжёра. Пример такого подхода описан в работе Фишера и Джайкумара [17] .

В качестве метода решения задачи маршрутизации транспорта с временными окнами существует метод ближайшего соседа. Основным достоинством данного метода является его простота, однако к недостаткам можно отнести то, что он может выдать не оптимальное решение. Тем не менее, исследования показывают его эффективность [18]. Работу метода можно описать так: на каждой итерации ищем ближайшего клиента, в чьё временное окно транспортное средство может попасть, при отсутствии

такого клиента ищется тот, опоздание к которому или же время ожидания будет минимально.

3 Разработка алгоритма

В общем и целом, работу алгоритмов, разработанных в рамках данной работы, можно разбить на решение двух подзадач.

Первая подзадача заключается в поиске оптимальной перестановки. Эта задача не имеет точного полиномиального решения и требует применения эвристических алгоритмов, как например ПГА. Здесь мы должны уметь генерировать корректные перестановки вершин, стремиться к тому, чтобы перестановки с лучшей метрикой имели большую вероятность появления, а с низкой метрикой – меньшую вероятность.

Вторая подзадача заключается в подсчете метрики по перестановке. Данная задача может быть решена за полиномиальное время. Эти задачи имеют слабую зависимость.

Далее будут рассмотрены три алгоритма «ПГА с явным кодированием», «Двухфазный ПГА» и «ПГА с гибкой структурой». Они отличаются подходами к решению первой подзадачи и совершенно не зависят от того, как решается вторая. Это позволяет при изменении в постановки задачи маршрутизации транспорта, пересмотреть только лишь постановку и решение второй подзадачи.

В рамках данных трех алгоритмов рассмотрим решение первой постановки задачи, описанной в главе 1. Для решения второй постановки задачи достаточно будет лишь описать решение второй подзадачи, поскольку первая подзадача останется неизменной.

3.1 ПГА с явным кодированием

Алгоритм построен на том, что особь будет явно представлять собой перестановку вершин, ген хромосомы в таком случае будет набором из N вероятностей появления числа на соответствующей гену позиции в особи, где N – количество целей. Пример хромосомы приведен на рисунке 7.

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 7 – пример хромосомы для ПГА с явным кодированием

При таком представлении возникает необходимость следить за корректностью перестановки, это будет учтено в функции соответствия. Алгоритм функции соответствия устроен следующим образом:

1. Подсчет эффективной вероятности по формуле (5);
2. Далее последовательно для каждого гена:
    2.1 Обнулить вероятности для вершин из списка использованных вершин;
    2.2 Нормализовать вероятности для гена так, чтобы сумма стала равна единице;
    2.3 Сгенерировать по полученным на шаге 2.2 вероятностям случайное значение для текущей позиции особи (совпадает с позицией гена);
    2.4 Добавить сгенерированное значение в список использованных вершин.

В результате работы алгоритма выше будет получена корректная перестановка или, другими словами, посчитана корректная особь.

Далее все особи отсортируются в порядке увеличения приспособленности. Для лучших особей будут пересчитаны их хромосомы по формуле (6). 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.

Алгоритм, описанный выше, будем называть ПГА с явным кодированием.

На рисунке 8 представлена иллюстрация изменения хромосомы, приведённой на рисунке 7, для ПГА с явным кодированием при условии, что по ней была посчитана особь {3, 1, 4, 2, 5} с низкой относительно других особей приспособленностью, индивидуальная скорость обучения для данного примера 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 8 – изменение вероятностей для особи с низкой приспособленностью

3.2 Двухфазный ПГА

Алгоритм с явным кодированием можно ускорить, разделив его на два этапа. На первом этапе выделить K $(K<N)$ переходов с наибольшей вероятностью, далее такие переходы будем называть доминантными. А на втором этапе рассматривать только доминантные переходы. Таким образом, уменьшиться размер хромосомы, который является определяющими в асимптотике времени работы алгоритма. Кроме того, это задает направление в поиске ответа для алгоритма, и он быстрее сходится к оптимуму, однако

увеличивается риск попасть в локальный оптимум. Данную версию алгоритма будем называть двухфазным ПГА.

На рисунке 9 приведена хромосома до выделения переходов. Красным выделены доминантные переходы.

	Гены хромосомы						
Значение гена		X1	X2	X3	X4	X5	...	X10
	1	0.1	0.25	0.01	0.19	0.01	...	...
	2	0.01	0.01	0.02	0.11	0.30	...	...
	3	0.01	0.01	0.4	0.09	0.01	...	...
	4	0.01	0.01	0.02	0.07	0.23	...	...
	5	0.01	0.4	0.1	0.04	0.01	...	...
	6	0.01	0.01	0.4	0.01	0.01	...	...
	7	0.15	0.01	0.01	0.22	0.01	...	...
	8	0.3	0.01	0.02	0.18	0.4	...	...
	9	0.3	0.25	0.01	0.01	0.01	...	...
	10	0.1	0.04	0.01	0.08	0.01	...	...

Рисунок 9 – хромосома двухфазного ПГА до сокращения

На рисунке 10 приведена хромосома после уменьшения размерности матрицы. Ячейка матрицы теперь описывается парой чисел: первое – вероятность появления значения, второе – значение.

	Гены хромосомы						
Значение гена		X1	X2	X3	X4	X5	...	X10
	1	(0.1 ; 1)	(0.25, 1)	(0.4, 3)	(0.19, 1)	(0.3, 2)	...	...
	2	(0.3, 8)	(0.4, 5)	(0.1, 5)	(0.22, 7)	(0.23, 4)	...	...
	3	(0.3, 9)	(0.25, 9)	(0.4, 6)	(0.18, 8)	(0.4, 8)	...	...

Рисунок 10 – хромосома двухфазного ПГА после сокращения

3.3 ПГА с гибкой структурой

Проблема двух предыдущих алгоритмов заключается в том, что при генерации текущей вершины маршрута, алгоритм почти никак не учитывал какие вершины были получены до этого. Удобно рассмотреть данную проблему на примере. Допустим у нас есть 5 целей, расположение целей

приведено на рисунке 11, депо имеет номер 0. На рисунке 11 также изображен оптимальный путь, порядок движения отмечен на ребрах.

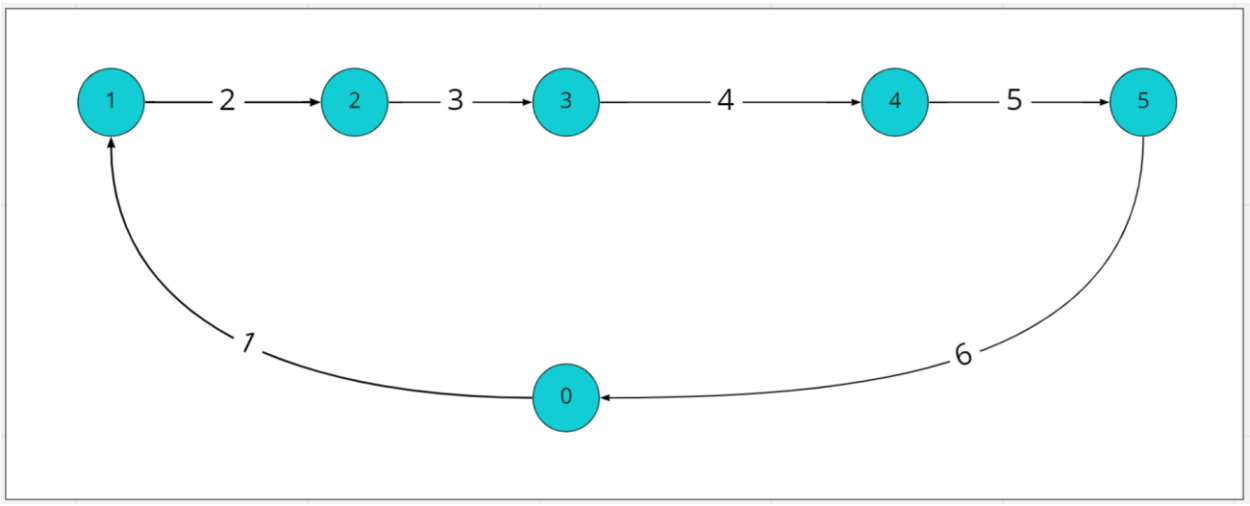

Рисунок 11 – Оптимальный маршрут

Предположим, что появилась особь, соответствующая маршруту на рисунке 11: [1, 2, 3, 4, 5]. Таким образом, будет увеличена вероятность появления на первой позиции вершины с номером один, на второй позиции – вершины с номером два и т.д. Теперь предположим, что на первой позиции появилась вершина с номером 5. На второй позиции все также большую вероятность имеет вершина с номером два, на третьей вершина с номером три и т.д. В итоге, с большой вероятностью мы получим маршрут [5, 2, 3, 4, 1]. Данный маршрут изображен на рисунке 12. Полученный путь будет одним из худших, и мы уменьшим вероятности появления вершин с номерами 2 – 4 на своих позициях, хотя они расположены оптимально.

В текущем алгоритме предлагается учитывать последовательность предыдущих вершин. Для этого пересмотрим определение вероятностей для хромосомы.

Теперь в столбце для гена с номером i на позиции j предлагается хранить вероятность P_{ij} – которая соответствует вероятности появления вершины с номером j в перестановке после вершины i. Таким образом, неважно, на какой позиции появилась вершина с номером i, следующая вершина будет получена, опираясь на вероятности в гене с номером i.

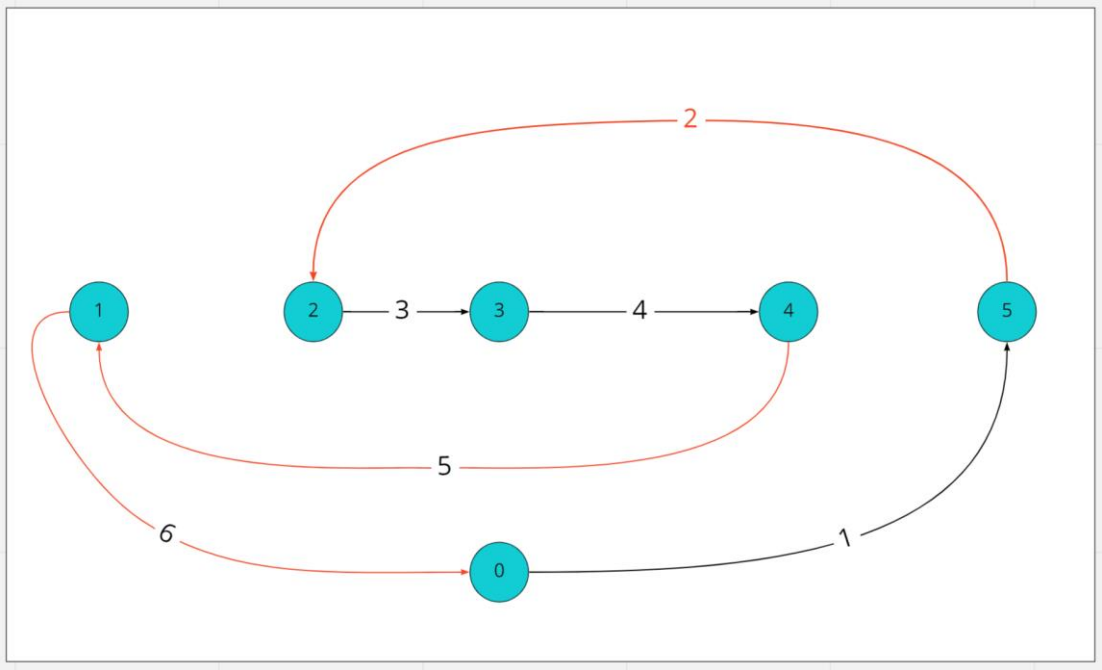

Рисунок 12 – Неоптимальный маршрут

Данный алгоритм требует незначительных изменений в пересчете вероятностей и получении новой особи. Назовем алгоритм, описанный в текущем подразделе как «ПГА с гибкой структурой».

3.4 Решение задачи с ограничениями

Как уже говорилось ранее в начале данного раздела для второй постановки задачи никак не будет меняться первая подзадача. Все также в рамках алгоритма будет рассматриваться поиск оптимальной перестановки, а для ее поиска будет использоваться «ПГА с гибкой структурой».

В рамках второй постановки задачи скажем, что перестановка вершин, в виде которой мы представляем особь, теперь задает нам ориентированность для всех ребер в исходном графе по следующему правилу: если вершина *i* стоит в перестановке раньше, чем вершина *j*, то тогда можно перейти по ребру (i, j) только в направлении из вершины *i* в вершину *j*. Нетрудно доказать, что в таком графе не будет циклов, и, в конечном итоге, исходный граф сведется к ациклическому ориентированному графу.

Напомним, что критерием оптимальности является максимизация количества посещенных целей, а при равенстве посещенных целей

дополнительным условием является минимизация пройденного расстояния. При условии, что задача сформулирована для ациклического ориентированного графа, она может быть решена с помощью метода динамического программирования за полиномиальное время.

Если бы не ограничение на грузоподъемность транспортного средства, то это была бы задача одномерного динамического программирования:

- в значении $dp[v]$ хранится пара из двух значений: максимального количества посещенных целей и минимального времени необходимого для того, чтобы посетить текущее максимальное количество целей;
- релаксация значений $dp[v]$ производится из всех вершин u, смежных с v или, другими словами, из вершин, которые стоят в перестановке ранее, чем вершина v. Релаксация возможна при условии попадания во временное окно.

Но ограничение на грузоподъемность делает данную структуру некорректной в силу того, что иногда мы можем пренебречь оптимальным ответом, чтобы получить выигрыш в расстоянии, которое не будем преодолевать от последней цели в маршруте до депо и от депо до первой цели в следующем маршруте, что в итоге даст возможность попасть во временные окна больших целей.

Задача все еще может быть решена методом динамического программирования, но теперь требует двух измерений: номер текущей вершины в перестановке vertex и количество целей $cntVisitedVertex$, которое посещено. В значении $dp[vertex][cntVisitedVertex]$ будем хранить минимальное время необходимое для того, чтобы посетить текущее количество целей $cntVisitedVertex$ при этом последней посещенной вершиной является вершина с номером $vertex$. Релаксация для текущей вершины v будет производиться из всех состояний вершин u, которые

смежны с ней (стоят раньше в перестановке), по формуле (8) при условии попадания во временное окно вершины v:

$$\begin{cases} tl_v \le dp[u][i] + dist(u,v) \le tr_v \\ dp[v][i+1] = \min\big(dp[v][i+1], dp[u][i] + dist(u,v)\big), i = 1..N \end{cases} \quad (8)$$

В случае, если состояние недостижимо будем хранить бесконечность (значение, которое больше максимального значения расстояния). Тогда потенциальными ответами будут достижимые состояния с максимальным вторым измерением, а среди всех таких окончательным ответом будет состояние с минимальным значением dp.

3.5 Пересчет вероятностей

Стоит также обратить особое внимание на пересчет вероятностей хромосом в рамках разработанных алгоритмов. Далее рассмотрим пример пересчета вероятностей для ПГА с гибкой структурой, который несильно отличается от пересчета в других алгоритмах: во всех трех алгоритмах мы пересчитываем гены, которые представляют из себя столбец с вероятностями.

Наивный пересчет вероятностей у хромосомы, соответствующей одной из наилучших особей равной перестановке A, в рамках текущей структуры может быть выполнен по формулам (9) (10):

$$\text{decreaseValue} = \frac{\text{iLR}}{\text{N}} \quad (9)$$

$$\begin{cases} P_{A_i j} = \max\big(1 - DR, P_{A_i j} + iLR\big), j = A_{i+1}, i = 1..N \\ P_{A_i j} = \min\big(DR, P_{A_i j} - \text{decreaseValue}\big), \text{j} \neq A_{i+1}, i = 1..N \end{cases} \quad (10)$$

, где N – количество целей, iLR – индивидуальная скорость обучения, DR – коэффициент разнообразия, P_{ij} – вероятность в i-ом гене хромосомы на позиции j.

При таком пересчете уже через несколько итераций главное свойство вероятностей одного гена, сумма вероятностей равна единице, перестанет выполняться, и числа, записанные в гене, станут приоритетами, на основе которых будут рассчитаны вероятности. Но гораздо большая проблема, что

эти приоритеты будут пересчитываться непропорционально, и при переводе приоритетов в вероятности некоторые значения могут стать меньше нижней границы или выше верхней границы, что неправильно. Для избежания данных проблем было предложено новое правило пересчета вероятностей. Рассмотрим пересчет одного гена, все остальные гены будут пересчитаны аналогично. Пусть в i–ой позиции перестановки было рассчитано A_i, а в следующей позиции A_{i+1}. Тогда правило пересчета A_i–ого гена выглядит следующим образом:

1) Вычисляем величину decreaseValue, которая будет вычитаться из вероятностей вершин, отличных от следующей вершины с номером A_{i+1}, используя величину canIncrease, на которую мы можем увеличить вероятность для вершины A_{i+1} в текущем гене (11) (12):

$$\text{canIncrease} = \min\left((1 - DR) - P_{A_iA_{i+1}}, \text{iLR}\right) \quad (11)$$

$$\text{decreaseValue} = \frac{\text{canIncrease}}{\text{vertexCnt}-1} \quad (12)$$

2) Из вероятностей появления вершин отличных от A_{i+1} вычитаем значение decreaseValue, так чтобы вероятность была не меньше нижней границы (DR) (13) (14). Вычитаемые величины суммируем в totalDecreaseValue (15):

$$\text{canDecrease} = \min\left(P_{A_ij} - DR,\ \text{decreaseValue}\right), j \neq A_{i+1} \quad (13)$$

$$P_{A_ij} = P_{A_ij} - \text{canDecrease},\ j \neq A_{i+1} \quad (14)$$

$$\text{totalDecreaseValue} \mathrel{+}= \text{canDecrease} \quad (15)$$

3) К вероятности появления вершины A_{i+1} для текущего гена прибавляем totalDecreaseValue (16):

$$P_{A_iA_{i+1}} = P_{A_iA_{i+1}} + \text{totalDecreaseValue} \quad (16)$$

Процесс изменения вероятности у хромосом, соответствующих особям с наименьшей приспособленностью, выглядит аналогично.

4 Тестирование и анализ результатов работы алгоритмов

4.1 Генерация входных данных

Для генерации входных данных задачи был реализован модуль, генерирующий тестовые файлы с заданным количеством целей и транспортных средств. Координаты целей генерируются случайным образом в рамках квадрата заданного размера. Цели имеют целочисленные координаты. Полученный набор входных данных записывается в указанный файл. Инструкцию для запуска данной утилиты можно найти в файле TestGenerator/Readme.md на ресурсе с приложением [20].

Всего было выделено 6 групп вершин со значениями: 10 вершин, 30 вершин, 45 вершин, 60 вершин, 80 вершин, 100 вершин.

Для отладки и тестирования было сгенерировано по 6 различных тестов для каждого набора выбранных вершин при помощи описанной утилиты. Далее утилита была модифицирована для создания еще 6 специальных тестов для каждого набора. Специальные тесты генерировались по следующему принципу: выделялись относительно небольшие области на карте, в границах которых генерировались вершины; таким образом, в оптимальном ответе эти вершины должны располагаться также близко. Помимо того, что специальные тесты помогают удостовериться в правильности работы алгоритма, они еще и соответствуют реальным значениям, возникающим при доставке товаров в рамках нескольких городов.

Генератор тестов также опционально позволяет генерировать временные окна, временные окна генерируются случайным образом. Исходный код модуля можно найти в репозитории [20].

4.2 Модуль для настройки гиперпараметров

Ранее мы ввели и использовали по ходу алгоритма такие гиперпараметры, как:

- индивидуальная скорость обучения (Individual learning rate);
- глобальная скорость обучения (Global learning rate);
- коэффициент разнообразия (Diversity rate);

, а также:

- размер популяции;
- количество итераций алгоритма.

Значения данных параметров очень сильно влияют на эффективность работы алгоритма, поэтому подбор оптимальных значений для них является достаточно важной задачей. В рамках данной ВКР был разработан инструмент для настройки этих параметров, и с его помощью были найдены значения для индивидуальной скорости обучения и коэффициента разнообразия, которые будут использованы на этапе тестирования.

Модуль настройки гиперпараметров на входе принимает:

- количество целей;
- путь к папке с тестовыми данными, соответствующими количеству целей;
- диапазон для перебора гиперпараметров и шаг перебора;
- значения тех гиперпараметров, которые не требуют перебора;
- приоритет перебора гиперпараметров.

Модуль фиксирует все параметры кроме одного и запускает перебор с заданным шагом для незафиксированного параметра. Вложено перебирается каждый из параметров в рамках его диапазона. Таким образом, просматриваются все возможные наборы.

В результате работы алгоритма мы имеем промежуточные результаты тестирования для каждого набора параметров, график оптимальных значений метрики для наиболее приоритетного параметра.

В качестве метрики было выбрано значение приспособленности лучшей особи среди всех операций.

Промежуточные результаты тестирования записываются в текстовый файл, а график оптимальных значений метрики для наиболее приоритетного параметра сохраняется в формате PNG. Пример графика приведен на рисунке 13.

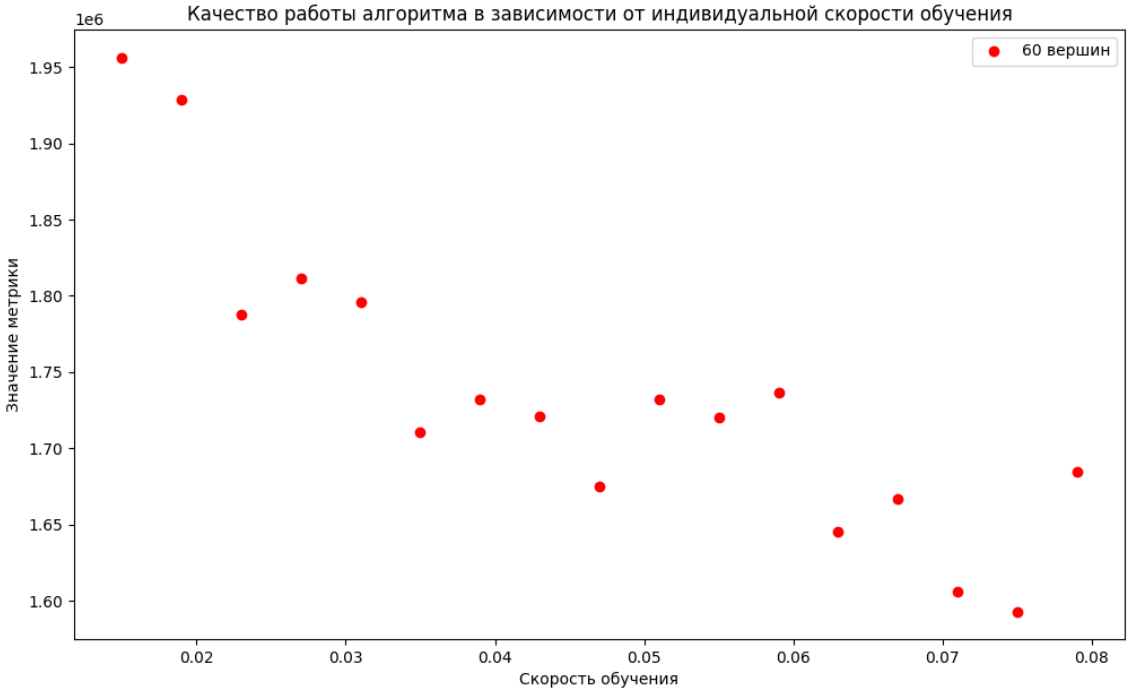

Рисунок 13 – График результатов перебора гиперпараметра

4.3 Поиск оптимальных значений гиперпараметров

Из опыта прошлых научных работ [19] можно сделать вывод, что количество итераций нужно максимизировать так, чтобы при этом время работы алгоритмы оставалось приемлемым (меньше одной минуты). Поэтому было принято решение, соблюдая баланс между эффективностью и временем работы алгоритма, взять количество итераций равное 2000. Размер популяции был взят за 30 особей.

Значение глобальной скорости обучения было установлено в 0.1. Было принято решение зафиксировать этот параметр, а для двух остальных осуществить перебор.

На первом этапе настройки параметров ПГА исследования проводились следующим образом:

- индивидуальная скорость обучения перебиралась с шагом 0.01 в диапазоне [0.001; 0.2];
- для каждого зафиксированного значения скорости обучения перебирался коэффициент разнообразия с шагом 0.001 в диапазоне [0.001 до 0.10].

В данном случае был неправильно выбран диапазон для перебора. Обратимся к инициализации вероятностей гена. В начале алгоритма, все вероятности гена P_{ij} совпадают и обратно пропорциональны количеству целей vertexCnt (16):

$$P_{ij} = \frac{1}{vertexCnt} \tag{16}$$

Таким образом, если коэффициент разнообразия установить не меньше, чем в формуле выше, то учитывая процесс пересчета вероятностей, получится, что вероятности совсем не будут меняться. В итоге мы получаем верхнюю границу для коэффициента разнообразия. Нужно понимать, что близкие к этой верхней границе значения будут сильно тормозить алгоритм и оптимальные решения не будут иметь высокую вероятность появления по

сравнению с неоптимальными, поэтому разумно взять верхнюю границу диапазона перебора для коэффициента разнообразия на порядок меньше значения в формуле (16). За нижнюю границу было принято взять значение еще на порядок меньше, а шаг установить так, чтобы получить перебор около 20 значений. Из предположения, что индивидуальная скорость обучения должна быть в разы больше коэффициента разнообразия, диапазон перебора индивидуальной скорости обучения был взят на один порядок больше, чем диапазон коэффициента разнообразия, с шагом таким, чтобы получить перебор около 20 значений.

Для каждой из выделенных на этапе тестирования групп было проведено исследование, выделены наиболее перспективные значения параметров, а далее эти значения были интерполированы линейной функцией для индивидуальной скорости обучения и экспоненциальной – для коэффициента разнообразия. Графики, по которым принималось решение для индивидуальной скорости обучения приведены на рисунке 14. Для определения значения коэффициента разнообразия анализировались результаты, полученные при оптимальных значениях индивидуальной скорости обучения.

По итогам исследования, оптимальная скорость обучения определяется формулами (17) (18):

$$iLR(cntVertex) = F_1(cntVertex) \quad (17)$$

$$F_1(x) = \frac{0.025*x+3.45}{70} \quad (18)$$

А оптимальное значение коэффициента разнообразия можно вычислить по формулам (19) (20):

$$DR(cntVertex) = F_2(cntVertex) \quad (19)$$

$$F_2(x) = 0.003903 * exp^{(-0.036642* x)} \quad (20)$$

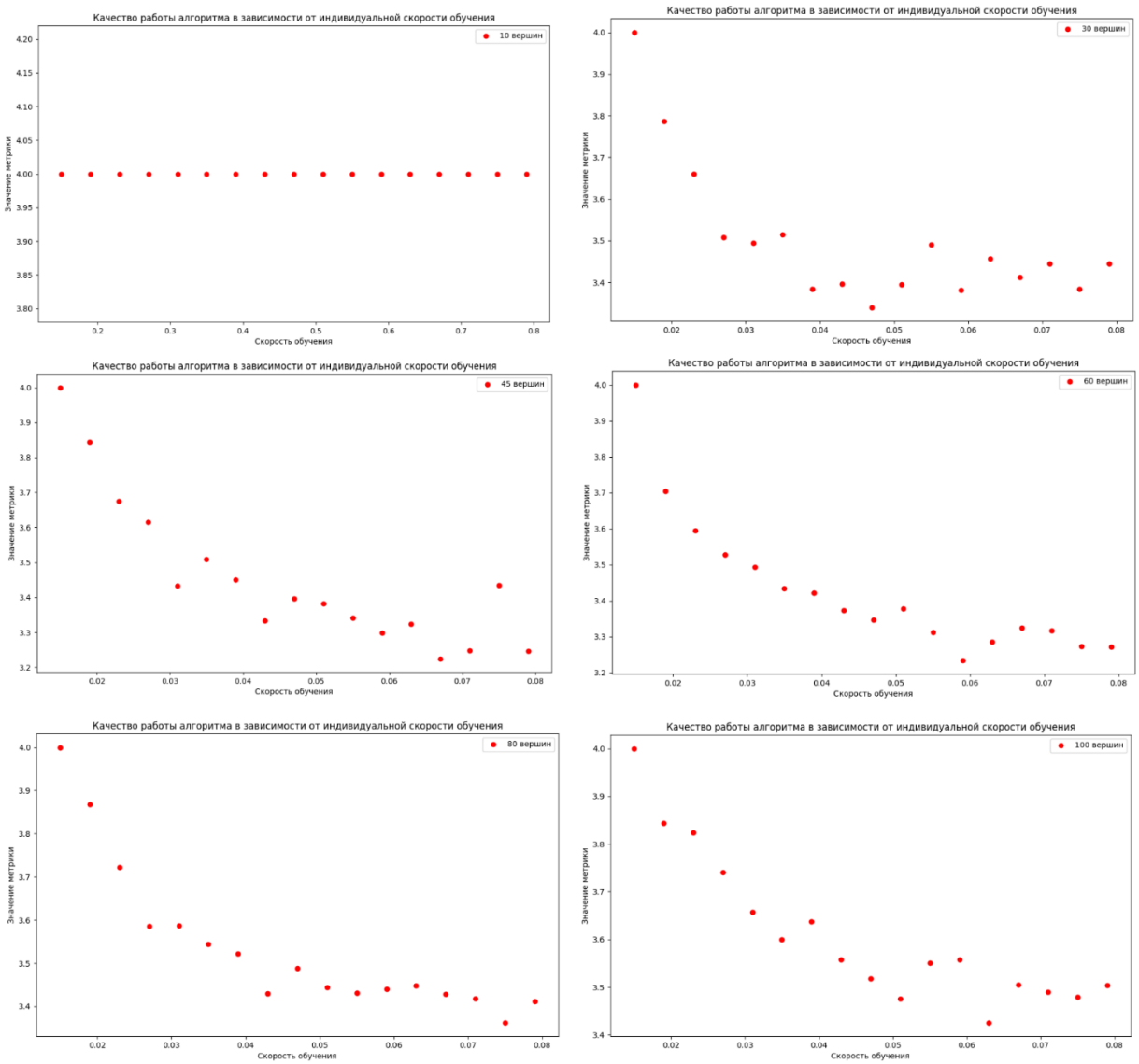

Рисунок 14 – результаты настройки для индивидуальной скорости обучения

4.4 Анализ результатов работы алгоритмов

В качестве эталонного алгоритма для тестирования был взят генетический алгоритм из ВКР Александра Цаплина [21]. На первом этапе сравнивались ПГА с явным кодированием, ПГА с гибкой структурой и эталонное решение. Тестирование проводилось для 6 групп вершин, выбранных на этапе генерации входных данных, со значениями гиперпараметров:

– индивидуальная скорость обучения равен: $1/(\text{Число вершин})$;

– глобальная скорость обучения равен: 0.1;

– коэффициент разнообразия равен: $1/(\text{Число вершин})^2$;

– размер популяции равен: 50;

– время работы ПГА было ограничено 2000 итераций.

Результаты тестирования приведены в таблице 1. Для каждого алгоритма приведена суммарная длина лучшего маршрута и количество итераций алгоритма, которое было выполнено.

Таблица 1 – Сравнение ПГА с гибкой структурой с другими алгоритмами

Количество целей	ГА		ПГА с гибкой структурой		ПГА с явным кодированием	
	Сумм. длина	Количество итераций	Сумм. длина	Количество итераций	Сумм. длина	Количество итераций
10	345884.35	2961909	346285.75	2000	348362.975	2000
30	490515.76	2710759	1033697.96	2000	1080216.15	2000
45	559805.71	2123089	1376465.25	2000	1416498.46	2000
60	600534.97	1761745	2203576.34	2000	2292222.85	2000
80	713635.42	1327395	2805996.97	2000	2880439.33	2000
100	809134.72	992384	3266081.97	2000	3973603.30000000	2000

На рисунке 15 приведены графики индивидуальной приспособленности, что соответствует обратной величине к суммарной длине маршрута, лучшей особи итерации для алгоритмов на базе ПГА. На рисунке 16 приведено сравнение общей приспособленности популяции,

среднего значения индивидуальной приспособленности по всем особям, для алгоритмов на базе ПГА.

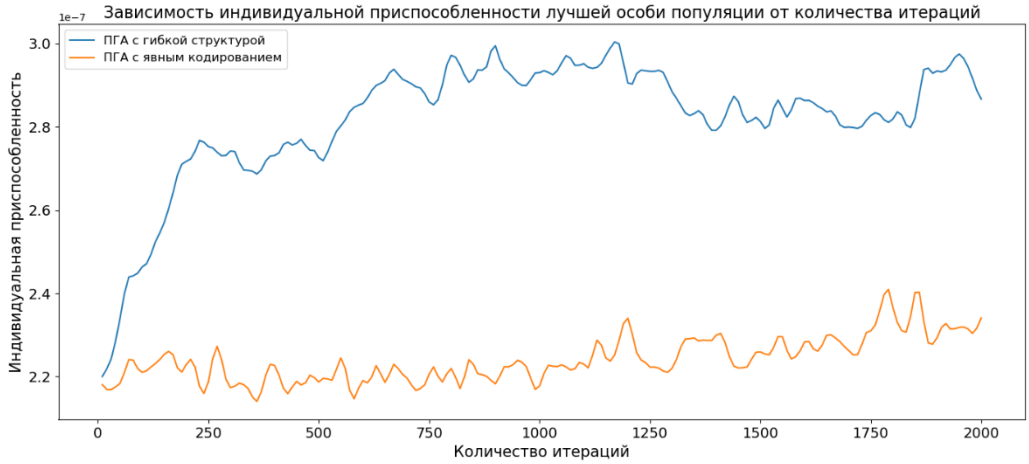

Рисунок 15 – графики индивидуальной приспособленности лучшей особи ПГА для первого этапа тестирования

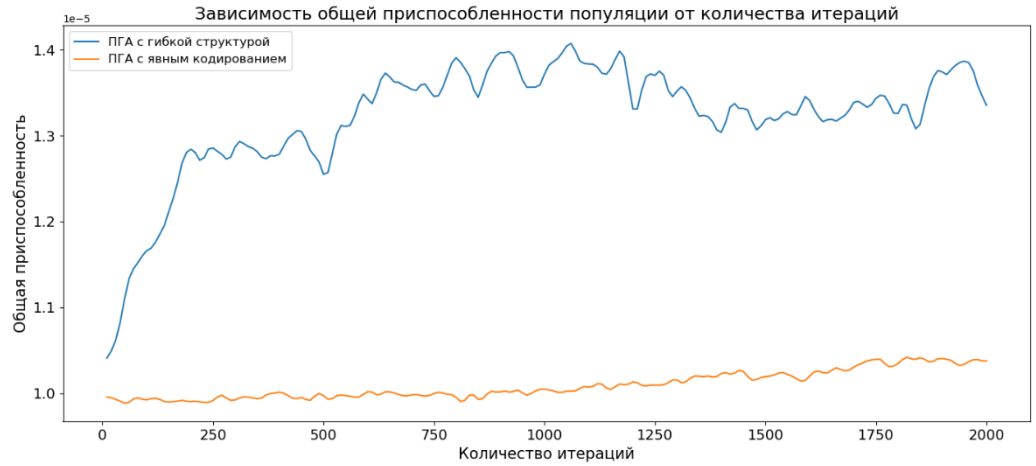

Рисунок 16 – графики общей приспособленности для первого этапа тестирования

По результатам первого этапа тестирования можно сказать, что ПГА с гибкой структурой работает чуть лучше, чем ПГА с явным кодированием, но уступает эталонному решению примерно в четыре раза на больших входных данных (больше 60 целей).

Далее была проведена настройка гиперпараметров. По результатам настройки были выбраны: скорость обучения в соответствии с формулой (17); коэффициент разнообразия в соответствии с формулой (19); глобальная скорость обучения взята со значением 0.1.

После настройки для второго этапа тестирования к алгоритмам с первого этапа были добавлены ПГА с гибкой структурой и ПГА с явным кодированием со значениями гиперпараметров, найденными в ходе настройки гиперпараметров. Результаты данного этапа приведены на рисунках 17 и 18.

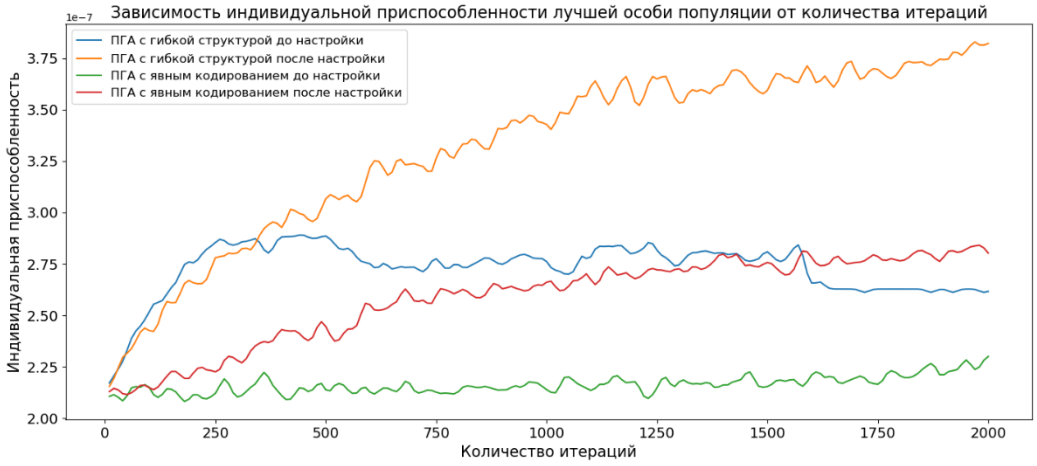

Рисунок 17 – графики общей приспособленности для второго этапа тестирования

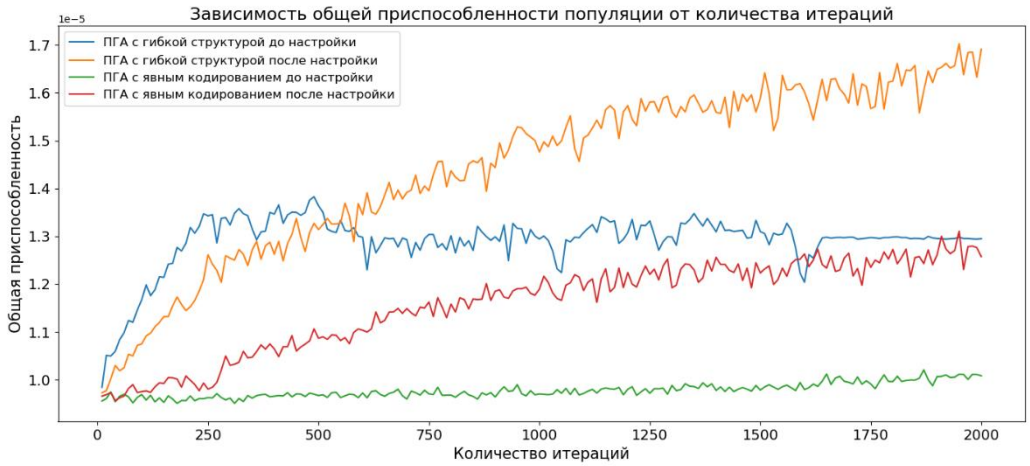

Рисунок 18 – графики индивидуальной приспособленности лучшей особи ПГА для второго этапа тестирования

Результаты второго этапа тестирования оказались ожидаемыми в силу рассуждений, изложенных в подразделе 3.3. Настроенный ПГА с гибкой структурой показал результаты лучше относительно других алгоритмов на всех группах входных данных. Можно сделать вывод, что для ПГА настройка

параметров является крайне важным аспектом, сильно влияющим на эффективность работы алгоритма. ПГА с гибкой структурой после настройки гиперпараметров был признан самым эффективным в рамках данной работы и будет использован далее в финальном тестировании вместе с эталонным решением.

На финальном этапе тестирование проводилось для шести групп вершин, выбранных на этапе генерации входных данных. Количество итераций для ПГА было ограничено в 2000 итераций, при таком ограничении алгоритмы работают примерно одинаковое количество времени, время работы ГА было ограничено тремя секундами. Результаты данного этапа тестирования приведены в таблице 2.

Таблица 2 – Результаты финального тестирования

Количество целей	ГА		ПГА с гибкой структурой	
	Сумм. длина	Количество итераций	Сумм. длина	Количество итераций
10	208514.03	3339480	208514.03	2000
30	439058.44	2857261	578660.80	2000
45	520942.24	2445380	968628.77	2000
60	637340.95	1925459	1266166.77	2000
80	717204.73	1547024	2030567.85	2000
100	782287.09	1026626	2344149.40	2000

По результатам тестирования можно сделать вывод, что ПГА уступает эталонному решению в эффективности в 2–3 раза. Это можно объяснить высокой эффективностью эталонного решения и наличием локальных оптимизаций в эталонном решении, которые сильно влияют на эффективность результатов генетического алгоритма.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы было решено 8 задач:

1) осуществлен анализ научной литературы, связанной с генетическими алгоритмами и подвижными генетическими алгоритмами. Выделены основные структурные элементы ПГА, а также основные гиперпараметры;

2) осуществлен анализ научной литературы, связанной с вариациями и возможными ограничениями для задачи маршрутизации транспорта. Сформулированы две постановки задачи как без ограничений, так и с ограничениями;

3) осуществлен анализ научной литературы, связанной с существующими подходами для решения задачи маршрутизации транспорта. За эталонное решения для сравнения качества разработанных алгоритмов взят генетический алгоритм;

4) предложены три алгоритма на базе подвижного генетического алгоритма для решения задачи маршрутизации транспорта;

5) реализовано два алгоритма на базе подвижного генетического алгоритма для решения задачи маршрутизации транспорта на языке C++;

6) разработан инструмент для поиска оптимального набора гиперпараметров алгоритма для произвольного количества вершин на языке Python;

7) выделены пять гиперпараметров ПГА. Для оптимальных значений двух ключевых из них подобраны линейная и экспоненциальная функции, зависящие от набора входных данных;

8) проведено тестирование алгоритмов и выполнено сравнение эффективности работы разработанных подходов. ПГА с гибкой структурой выделен как лучший алгоритм и сравнен с эталонным решением.

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

Перспективы дальнейших исследований и разработок заключаются в разработке графического приложения для возможности взаимодействия с алгоритмом пользователей, не имеющих опыта в программировании, а также в дополнительном изучении гиперпараметров алгоритма.

Апробация работы была проведена на 7–й научно–практической конференции «Математическое и программное обеспечение информационных и интеллектуальных систем», Пермь, ПГНИУ, 28–29 апреля 2022 г.

Исходный код алгоритмов и модулей, разработанных в рамках данной работы, находится в открытом доступе. Исходный код можно скачать из репозитория ресурса Github [20].

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. *Fisher M. L.* Optimal Solution of Vehicle Routing Problems Using Minimum K-trees // Operations Research 42, 1994. P. 626-642.

12. *Clarke G., Wright J. W.* Scheduling of Vehicles from a Central Depot to a Number of Delivery Points // Operations Research 12, 1964. P. 568-581.

13. *Bullnheimer B., Hartl R. F., Strauss C.* Applying the Ant System to the Vehicle Routing Problem // 2nd International Conference on Metaheuristics, Sophia-Antipolis, France, 1997.

14. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях, 2003, №4, с.70-75.]

15. van Laarhoven P.J.M., Aarts E.H.L. (1987) Simulated annealing. In: Simulated Annealing: Theory and Applications. Mathematics and Its Applications, vol 37. Springer, Dordrecht.

16. Osman, I.H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann Oper Res 41, 421–451 (1993).

17. *Fisher M. L., Jaikumar R.* A Generalized Assignment Heuristic for Vehicle Routing // Networks, 11, 1981. P. 109-124.

18. Рассадникова Е. Ю. Методы и алгоритмы определения рациональных маршрутов для задачи маршрутизации транспортных средств с учетом временных окон и других условий / George Kovács, Nafissa Yusupova, Olga Smetanina, Rassadnikova E. Yu. Methods and Algorithms of Rational Routes Determination for Vehicle Routing Problem with Time Windows and Other // Pollack Periodica. – 2018. – V. 13. – №. 1. – pp. 65-76.

19. Сидоренко Д. О., Городилов А. Ю. О решении задачи маршрутизации транспорта с помощью подвижного генетического алгоритма. Вестник ПГУ. Математика. Механика. Информатика. 2021. № 4(55).

20. Using FGA to solve Vehicle Routing Problem [Электронный ресурс] [Режим доступа: https://github.com/DimaSidorenko/VRP.FGA_2022].

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