Санкт-Петербургский государственный университет

***ТАРАСЕНКО Любовь Владимировна***

**Выпускная квалификационная работа**

***Оценка и распределение бонусных баллов при составлении трассы в спортивном ориентировании***

Уровень образования: бакалавриат

Направление 02.03.02

«Фундаментальная информатика и информационные технологии»

ООП «Программирование и информационные технологии»

Научный руководитель:  
доцент кафедры компьютерного моделирования и многопроцессорных систем, кандидат технических наук  
Гришкин Валерий Михайлович

Рецензент:

старший преподаватель кафедры технологии программирования,

Стученков Александр Борисович

Санкт-Петербург

2019

**Содержание**

[Введение 4](#_Toc9997030)

[Постановка задачи 8](#_Toc9997031)

[Обзор литературы 10](#_Toc9997032)

[Глава 1. Основные сведения о проблеме ориентирования 11](#_Toc9997033)

[1.1 Понятие проблемы ориентирования 11](#_Toc9997034)

[1.2 Математическая модель проблемы ориентирования 12](#_Toc9997035)

[1.3 Вариации проблемы ориентирования 13](#_Toc9997036)

[1.4 Применение задачи на практике 14](#_Toc9997037)

[Глава 2. Обзор существующих алгоритмов 15](#_Toc9997038)

[2.1 Генетический алгоритм 16](#_Toc9997039)

[2.2 Эвристический алгоритм 18](#_Toc9997040)

[2.3 Сравнение генетического и эвристического алгоритмов 19](#_Toc9997041)

[Глава 3. Математическое обоснование задачи и подготовка к реализации 20](#_Toc9997042)

[3.1 Математическое обоснование задачи и составление модели для контрольного тестового примера 20](#_Toc9997043)

[3.2 Составление алгоритма на естественном языке 21](#_Toc9997044)

[3.3 Усовершенствование алгоритма в сторону балансировки трассы 25](#_Toc9997045)

[Глава 4. Практическая реализация 27](#_Toc9997046)

[4.1 Выбор средств для реализации 27](#_Toc9997047)

[4.2 Описание работы программы 27](#_Toc9997048)

[4.3 Составление тестовых графов 27](#_Toc9997049)

[4.4 Оценка работы алгоритма 29](#_Toc9997050)

[Выводы 30](#_Toc9997051)

[Заключение 31](#_Toc9997052)

[Список литературы 33](#_Toc9997053)

[Приложения 35](#_Toc9997054)

[Приложение 1. Карта-трасса для соревнований с контрольными пунктами 35](#_Toc9997055)

[Приложение 2. Тестовые графы 36](#_Toc9997056)

[Приложение 3. Результаты работы алгоритма на тестовых графах 42](#_Toc9997057)

# Введение

В современном мире компьютерные технологии применяются практически в каждой отрасли деятельности человека. Программы помогают человеку проводить трудоемкие вычисления в любой сфере, начиная с науки и заканчивая увлечениями. Одним из видов деятельности, где компьютерные технологии используются еще не в полную силу, является спорт и туризм. Спортивное ориентирование – это такой вид спорта, в котором участники с помощью карты и компаса должны пройти определенную дистанцию по пересеченной местности и при этом посетить некоторое количество контрольных пунктов. Победителем считается тот участник, который прибыл на финиш раньше других. Существует еще и другой вид спорта – рогейн, который можно считать разновидностью спортивного ориентирования. В рогейне за каждый посещенный контрольный пункт команде начисляется определенное количество бонусных баллов или очков. Чтобы победить, нужно набрать как можно большее количество очков за ограниченное время. В отличие от спортивного ориентирования, в рогейне команде необязательно посещать все контрольные пункты, она вольна сама строить свой маршрут. Результат участников соревнований зависит не только от их физической формы и умения ориентироваться по картам и компасу, но и от верно выбранной стратегии прохождения трассы. Для организаторов подобных соревнований важно правильно составить трассу: помимо всего прочего, нужно справедливо рассчитать, какое количество очков начислять команде за каждый взятый контрольный пункт. Данный процесс называется балансировкой трассы. Существует еще один смежный вид спорта – спортивный автостоп. В спортивном автостопе командам также нужно посетить определенное количество контрольных пунктов, получая за каждый из них баллы, и прибыть на финиш, уложившись в контрольное время. Разница лишь в том, что трасса гораздо длиннее, чем в спортивном ориентировании, а передвигаться участники могут лишь автостопом или пешком.

Итак, допустим, что у нас есть какая-то территория, какой-либо административный округ Российской Федерации. На этой территории мы расставляем контрольные пункты – места, в которых должны побывать участники соревнований. Мы также устанавливаем максимальное время прохождения дистанции. Допустим, в соревнованиях участвует определенное количество команд, мы заранее знаем, сколько их. Нам нужно рассчитать количество очков, которое мы будем присуждать каждой команде за каждый посещенный контрольный пункт. Важно учитывать тот факт, что первая команда, посетившая контрольный пункт получает наибольшее возможное количество очков за него, вторая команда получает меньше очков, чем первая, и так далее.

На данный момент, подобные трассы для видов спорта с ориентированием чаще всего, к сожалению, составляются вручную. Данный процесс можно было бы ускорить и избежать большого количества ошибок, если его автоматизировать. В научных кругах разновидности и вариации этой задачи относятся к проблеме ориентирования, известной, в основном, из англоязычных источников. Итак, проблема ориентирования – задача маршрутизации на графах. Дан граф – ориентированный или неориентированный; каждой его вершине присвоено определенное количество бонусных очков; ребра графа также имеют определенный вес. Задача заключается в том, чтобы найти маршрут, максимизирующий количество бонусных очков и имеющий вес не более заданного. Проблема ориентирования имеет множество вариаций, таких как проблема командного ориентирования или ориентирование с временными окнами. Все подобные задачи можно считать также частными случаями задачи коммивояжера. Тем не менее, все вышеперечисленные проблемы нацелены лишь на то, чтобы составить оптимальный маршрут. Задачи балансировки очков за контрольные пункты, что является подзадачей составления трассы, в литературе не встречается.

Несмотря на то, что в этой работе мы будем рассматривать применение данных проблем в сферах спорта и туризма, можно найти еще несколько сфер, где решение данной проблемы было бы полезным. Так, построенные модели могут быть полезными при составлении расписания для курьеров или дежурных, которым нужно обойти определенное количество клиентов или объектов за фиксированное время. В случае с курьерами бонусными очками считается прибыль от клиента, в случае с дежурными, приняв вместо очков на контрольном пункте время возможной задержки дежурного, мы рассчитаем оптимальное расписание. Другими словами, проблема ориентирования актуальна во многих отраслях деятельности современного человека.

Цель данной работы заключается в создании либо выборе такого алгоритма, который позволял бы рассчитывать количество очков для контрольных пунктов в трассах спортивного ориентирования. Для данной работы достаточно сформулировать основные идеи алгоритма на естественном языке. Среди задач данной работы можно выделить сбор и изучение уже существующей информации по теме, обзор и сравнение алгоритмов по смежным темам, выбор оптимальных технических средств для реализации алгоритма, разработка примера для тестирования полученного алгоритма, создание либо выбор и реализация самого алгоритма, решающего подзадачу основной проблемы, анализ работы созданной программы, а также ее результатов.

На данный момент, к сожалению, в русскоязычных источниках почти нет информации, связанной с проблемой ориентирования. В англоязычных источниках огромная часть литературы посвящена выбору маршрута в данной задаче. В этой работе главной целью ставится предложение идеи для создания алгоритма, нацеленного на составление сбалансированной трассы для спортивного ориентирования или рогейна. Научная новизна состоит в самой постановке задачи. Первая часть данной работы посвящена уже известным моделям по данной проблеме, во второй части мы рассмотрим алгоритмы поиска наиболее оптимального маршрута и сравним их. Далее, рассмотрим подготовительный этап для создания нашей программы, коснемся самой реализации алгоритма, и в финальной части проанализируем результаты работы алгоритма.

# Постановка задачи

Целью данной работы является создание либо выбор такого алгоритма, который позволял бы рассчитывать наилучший маршрут для участников соревнований, а также предложение идей по усовершенствованию данного алгоритма для решения проблемы балансировки трассы в спортивном ориентировании. Иными словами, нужно предложить идеи по автоматизации процесса расчета и начисления очков командам, участвующим в соревнованиях по спортивному ориентированию. Данная задача делится на две логических части: взгляд на проблему со стороны участников и взгляд на проблему со стороны организаторов соревнований. Участникам нужно набрать наибольшее количество очков и приехать на финиш до истечения фиксированного срока времени. При этом все контрольные пункты одной команде посетить невозможно. За взятие одного и того же контрольного пункта разные команды получают разное количество баллов: в зависимости от порядка прибытия на контрольный пункт. Отсюда вытекает взгляд на проблему со стороны организаторов: сколько очков присуждать командам за каждый контрольный пункт? На сколько очков понижать стоимость контрольного пункта с прибытием на него новой команды? Мы не можем присвоить контрольным пунктам баллы случайно: тогда не выполнится условие того, что команда, двигающаяся по изначально выгодной траектории, обязательно победит. Также, нам важно разгрузить самые простые дороги и легкодоступные контрольные пункты и сделать так, чтобы у команд была мотивация достигнуть отдаленных контрольных пунктов.

Данная задача тесно связана с теорией графов. Поэтому, если представить карту соревнований в виде графа, для расчета нужного нам маршрута достаточно лишь составить или выбрать определенный алгоритм обхода графа.

В связи с описанной выше формулировкой проблемы в рамках данной научной работы были поставлены следующие прикладные задачи:

1. Реализовать на любом языке программирования алгоритм максимального обхода графа за фиксированное время (это задача со стороны участников соревнований);
2. Предложить дальнейшие действия для усовершенствования алгоритма в сторону балансировки трассы для соревнований по спортивному ориентированию (это взгляд со стороны организаторов соревнований);
3. Разработать контрольный тестовый пример трассы для соревнований на основе реально существующей карты и протестировать на нем алгоритм максимального обхода графа за фиксированное время.

# Обзор литературы

В статьях [1] и [2] рассматривается проблема ориентирования в целом, а также представляются ссылки на алгоритмы и другие статьи, связанные с темой. Две данных работы – это обзоры на новые методы решения проблемы ориентирования.

В источниках [3] и [4] рассмотрены эвристический метод и генетический алгоритм для решения общей проблемы ориентирования. Именно на этих алгоритмах мы сосредоточимся в нашей дальнейшей работе. Подробнее данные алгоритмы рассмотрены в главе 2. Еще один генетический алгоритм рассмотрен в статье [5], но в нем, помимо прочего, добавляется адаптационная функция штрафов.

[6],[7],[8],[9],[10],[11] статьи нацелены на проблемы командного ориентирования, в том числе командного ориентирования с временными окнами. Вкратце данные задачи будут рассмотрены в главе 1 этой научной работы.

Смешанное командное ориентирование с временными окнами рассмотрено в статье [12]. Статья [13] рассматривает кластерную проблему ориентирования на примере ставшей недавно популярной игры – Pokemon GO. В источнике [14] говорится о проблеме ориентирования, в которой очки ставятся в соответствие не вершинам, а ребрам графа; а в источнике [15] - о стохастической проблеме ориентирования. [16] и [17] статьи рассказывают о новых приближенные алгоритмах для решения проблемы ориентирования, а также ее вариаций. В статье [18] рассматривается евклидов вариант проблемы ориентирования.

Наконец, статья [19] рассматривает ту вариацию проблемы ориентирования, в которой очки зависят друг от друга. Данная проблема достаточно близка к теме данной научной работы.

# Глава 1. Основные сведения о проблеме ориентирования

Словосочетание «проблема ориентирования» тесно связано с теорией графов, если быть точнее, это раздел теории графов. Он содержит в себе большое количество нерешенных проблем и недоказанных гипотез. Название этого раздела происходит от популярного вида спорта – спортивного ориентирования, в котором спортсменам нужно искать маршрут на пересеченной местности с помощью карты и компаса.

## Понятие проблемы ориентирования

Проблема ориентирования – это задача маршрутизации, ее цель – выбрать множество вершин (построить путь), которые нужно посетить так, чтобы максимизировать сумму очков за все посещённые вершины и не превысить контрольное время. Существует несколько вариаций данной задачи: проблема командного ориентирования, проблема командного ориентирования с временными окнами, а также, проблема ориентирования, зависящая от времени. Данные темы популярны в таких отраслях жизни человека как логистика или туризм. Сама проблема ориентирования тесно связана с задачей коммивояжера и с задачей о рюкзаке.

Задача коммивояжера – задача комбинаторной оптимизации, в которой цель – найти самый выгодный (кратчайший, дешевый и тому подобное) маршрут, обязательно проходящий через каждую вершину. Стартовая и конечная вершины в данной задаче совпадают. Задача является NP-полной, поэтому эффективные методы решения – эвристические. Популярным в решении данной задачи является метод ветвей и границ, муравьиные алгоритмы, генетические алгоритмы и различные алгоритмы динамического программирования. Одним из возможных способов решения данной задачи является эвристика Лина-Кернигана. Это алгоритм локального поиска, основная идея которого состоит в том, чтобы построить маршрут (зачастую, используя другие алгоритмы поиска кратчайшего маршрута в графе), пересекающий сам себя, а потом изменить порядок вершин в нем так, чтобы избавиться от пересечений. Алгоритм решает на каждом этапе, сколько маршрутов и между какими именно вершинами он должен изменить, чтобы сделать общий путь короче.

Задача о рюкзаке – задача комбинаторной оптимизации, в которой цель – найти самый выгодный набор предметов, помещающихся в рюкзак при ограниченном объеме рюкзака. Также как и задача коммивояжера, задача о рюкзаке – NP-полная. Для ее решения также подходят метод ветвей и границ, жадные алгоритмы, методы динамического программирования.

В литературе проблема ориентирования может встречаться под разными названиями: задача коммивояжера с наградами или частный случай ограниченной в ресурсах задачи коммивояжера. Многие вариации проблемы ориентирования являются трансвычислительными, это значит, что уже при относительно небольшом числе вершин задачи не могут быть решены полным перебором за приемлемое время. В этом случае используется эвристический подход: на выходе выдается лишь приблизительное решение.

## Математическая модель проблемы ориентирования

Дан граф *G*=(*V*,*E*). Известно множество вершин, их *N* штук. У каждой вершины есть своё количество баллов – *si*. Стартовая и конечная точка – вершины 1 и *N* - зафиксированы. Известно время передвижения для всех существующих ребер между каждой вершиной *i* и вершиной *j* – *tij*. Не все вершины могут быть посещены за данное ограниченное время – *Tmax*. Найти в задаче нужно путь, ограниченный *Tmax*, который включает в себя некоторые вершины таким образом, чтобы максимизировать суммарный счет, полученный на всех посещенных вершинах. В различных вариациях задачи встречаются также условия о том, что каждая вершина может быть посещена лишь единожды или стартовая и конечная вершины совпадают и так далее.

## Вариации проблемы ориентирования

Проблема командного ориентирования – это задача, в которой нужно построить *P* путей, каждый из которых ограничен *Tmax*, так, чтобы максимизировать общий полученный счет. Данная проблема рассчитана на несколько команд или человек, каждая или каждый из которых пытается набрать как можно большее количество очков за один и тот же временной промежуток.

Проблема ориентирования с временными окнами: в этой вариации задачи у каждой вершины помимо баллов есть еще и временной промежуток [*Oi*, *Ri*]. Посетить вершину можно только в этот временной промежуток. Есть еще и проблема командного ориентирования с временными окнами: для решения подобных задач пробовали использовать различные эволюционные алгоритмы (пчелиный алгоритм), различные быстрые итеративные поиски, и так далее.

Проблема ориентирования, зависящая от времени: здесь, время передвижения между двумя вершинами зависит от времени прибытия на первую из них. Эта задача может встречаться на практике в сферах общественного транспорта – сколько минут нужно будет ждать маршрутку, если приехать строго в назначенное время или, возможно, ее вообще не придется ждать?

Существуют и другие вариации проблемы ориентирования: иногда вместо финальной суммы используется определенная нелинейная функция для подсчета итоговой награды, или нужно не максимизировать сумму очков, а минимизировать, или вариация командного ориентирования, в которой у каждого участника количество набранных очков не должно превышать заранее фиксированного. В некоторых задачах бонусные баллы распределяются не по вершинам, а по ребрам графа. В других задачах, называемых стохастической проблемой ориентирования, предсказать время передвижения между двумя вершинами крайне сложно либо невозможно. Кластерная проблема ориентирования заключается в том, что вершины графа делятся на группы, а баллы за достижение вершин можно получить, лишь посетив все вершины в кластере. Мультиагентная проблема ориентирования является разновидностью командного ориентирования, но вместо централизованного планирования всех маршрутов, здесь каждый агент может взаимодействовать с другими в целях личной выгоды.

## Применение задачи на практике

Наиболее распространенная потребность в решении данной задачи возникает у различных служб, которым нужно обслужить несколько клиентов, например, курьерская служба. Допустим, есть множество клиентов, которых можно посетить сегодня, но ясно, что всех посетить не удастся из-за ограничения рабочего времени, поэтому нужно выбрать клиентов так, чтобы собрать максимальный доход за день. Помимо очевидных примеров полезности данной задачи на практике, встречаются и другие: в 2003 году была описана подобная задача, связанная с построением телекоммуникационных сетей. В 2008 году было найдено еще одно применение задачи: мобильный туристический гид. Это приложение в мобильном устройстве туриста, которое позволяет ему из всего множества интересных для посещения мест выбрать те, что кажутся туристу наиболее привлекательными и построить план (порядок) их посещения.

# Глава 2. Обзор существующих алгоритмов

Итак, мы разобрались в постановке задачи ориентирования. Теперь рассмотрим уже имеющиеся алгоритмы решения задачи. В этой главе мы не будем перечислять все существующие алгоритмы, так как их достаточно много, мы рассмотрим подробно только два из них – те, с которыми планируем работать в дальнейшем. Данные алгоритмы были представлены в научных статьях [3],[4] 1996 и 2018 годов и нацелены лишь на проблему ориентирования с наградами.

Вообще, по различным вариантам проблемы ориентирования было создано довольно много различных алгоритмов. В первую очередь, их можно поделить на точные и неточные. Если граф включает в себя больше десяти вершин, то точные алгоритмы работают слишком долго, поэтому гораздо более применимы на практике неточные методы и алгоритмы. Самый простой пример точного алгоритма: полный перебор. Если говорить, например, о задаче коммивояжера, то при количестве вершин более 70, она не может быть решена компьютерами полным перебором за время, меньшее нескольких миллиардов лет.

Существуют роевые алгоритмы, генетические алгоритмы, жадные алгоритмы, алгоритмы метода ветвей и границ (являются наиболее популярными). Существуют и различные гибридные алгоритмы, в которых используются сочетания вышеназванных методов. Также, в научных работах встречаются эвристические алгоритмы, которые нельзя отнести ни к одной из вышеуказанных групп. В данной работе подробно рассматриваются два алгоритма – генетический и эвристический на основе двухточечного обмена. Данные алгоритмы были выбраны для подробного изучения в связи с тем, что описание тех задач, которые они решают, наиболее близко к описанию нашей задачи.

Для полного понимания описанных ниже алгоритмов, необходимо вспомнить еще несколько понятий. Жадный алгоритм – алгоритм, принимающий в локальных ситуациях оптимальные решения, и, таким образом, получающий (надеющийся получить) глобальное оптимальное решение. В задачах на графах данный алгоритм редко дает оптимальные решения. Тем не менее, в локальных подзадачах, а также для поиска приближенных решений, его можно использовать.

Генетический алгоритм – алгоритм, созданный по принципу естественного отбора в природе. Всегда имеет такие этапы как размножение и наследование, мутация, селекция (создание нового поколения). Плюсом данного алгоритма является скрещивание (кроссовер) – ребенок может унаследовать положительные (выгодные) черты от родителей и не унаследовать отрицательные. Данный вид алгоритмов распространен в задачах, связанных с графами.

## 2.1 Генетический алгоритм

Особенность алгоритма, представленного в статье [4], заключается в том, что на протяжении всей своей работы алгоритм поддерживает те решения, которые кажутся невозможными. Поэтому, данный алгоритм не теряет хорошие варианты решений, которые изначально могут показаться проигрышными.

Алгоритм имеет схему стандартного генетического алгоритма с некоторыми улучшениями: оптимизация полученного маршрута путем добавления и удаления новых вершин. Алгоритм был протестирован на 344 экземплярах и показал себя хорошо на графах среднего размера (до 400 вершин) и превосходно на графах большого размера (около 7000 вершин). Алгоритм делится на несколько частей. Рассмотрим подробнее каждую из них.

Инициализация популяции: случайно выбирается некоторое множество вершин. Далее случайно создается путь на этих вершинах.

Выбор родителей: два случайных индивида сравниваются между собой с помощью целевой функции или ожидаемого количества очков. При достаточной разнице, маршруты считаются пригодными для родительства. Обычно в генетических алгоритмах для выбора из всего возможного множества членов популяции пары родителей используется так называемая функция пригодности. В данном алгоритме функция пригодности – это пропорциональный выбор. Вероятность выбора родителей при пропорциональном выборе считается по формуле:

.

Кроссовер: создание нового дочернего маршрута. Сохраняются все вершины, являющиеся общими для обоих родителей; вершины, которые есть только в одном родительском маршруте, добавляются в дочерний в случайном порядке. Алгоритм строит «карту ребер» - совмещая два родительских маршрута, получает новый граф. На этом графе строится новый маршрут. Промежуточный путь между двумя вершинами также выбирается случайно.

Мутация: выполняется после каждого пересечения. Если вершина входит в уже построенный дочерний маршрут, то она выбрасывается из него, а на ее место вставляются две соседних вершины. Если вершина не входит в дочерний маршрут, алгоритм пытается добавить ее в наилучшем месте.

Улучшение маршрута: решения, которые получились больше ограничения маршрута, алгоритм пытается сократить. Также, там, где это возможно, алгоритм уменьшает длины полученных маршрутов, чтобы вставить новые вершины.

В алгоритме также используются операторы удаления и добавления вершин. Для того чтобы оценить, какую вершину лучше всего добавить в маршрут, и где именно, мы сначала ищем три ближайших посещенных вершины. Мы вводим специальную функцию и, в зависимости от смежности данных четырех вершин, присваиваем ей определенное значение (например, стоимость вставки вершины-кандидата между двумя смежными вершинами). Из всех возможных вариантов мы выбираем тот, в котором наша функция имеет наименьшее значение.

Критерий останова: алгоритм останавливается либо когда истекает заранее определенное время выполнения программы, либо когда все новые маршруты перестают отличаться от лучшего построенного маршрута больше чем на заранее определенную разницу.

Для непосредственного решения подзадачи коммивояжера, данный алгоритм использует эвристику Лина-Кернигана, описанную в первой главе данной научной работы.

## 2.2 Эвристический алгоритм

Этот алгоритм состоит из двух этапов: инициализация и улучшение.

Сначала строится эллипс по всему набору точек, в качестве фокусов эллипса используются стартовая и финишная точки. Ограничение по времени используется в качестве длины большой оси. В процессе создания маршрута используются только точки, лежащие внутри эллипса, так как при использовании всех остальных нарушается ограничение по времени.

На этапе инициализации используется жадный метод, для вставки в маршрут точки, путь до которой наименьший (здесь игнорируется количество очков, получаемое за посещение этой точки). Строится *L* решений для каждой задачи, где *L* – минимум из [*10*, *N*], а *N* – количество точек в эллипсе. Для получения *l*-ого решения находится точка с *l*-м наибольшим расстоянием от стартовой и финишной точек, через эти три точки формируется путь. Затем жадным образом вставляются оставшиеся точки (относительно расстояния, не стоимости). Когда вставка новых точек начинает нарушать ограничение по времени, мы строим путь через оставшиеся точки жадным методом. Строим пути, пока все точки внутри эллипса не окажутся внутри какого-нибудь из путей. Теперь мы среди всех полученных путей выбираем тот, у которого наибольший счет. Мы получили исходное решение.

Следующий этап - «двухточечный обмен». Мы пытаемся обменять две точки – одну из исходного пути, а вторую из любого другого пути – между собой, и, таким образом, найти более выгодное решение. Новая точка вставляется в исходный путь так, чтобы увеличение расстояния было минимальным.

Следующий этап: добавление точки в маршрут. Мы пытаемся вставить точку, пока не состоящую в пути, между двумя вершинами, где суммарное расстояние получится минимальным. Итак, мы получили первый вариант решения задачи.

После первой итерации (прохождения всех вышеуказанных этапов) алгоритм начинает все сначала, пытаясь убрать из первого исходного пути несколько точек, заменив их другими. Так продолжается до тех пор, пока не будет найден более выгодный маршрут, либо, пока не будет превышено время работы программы. Данный алгоритм был протестирован на 107 различных тестовых экземплярах.

## 2.3 Сравнение генетического и эвристического алгоритмов

Генетический алгоритм сложнее в реализации, и лучше работает на графах с большим количеством вершин. Он также быстрее приходит к хорошему решению. Тем не менее, эвристический алгоритм хорош в своей простоте реализации и относительно легок для понимания.

# Глава 3. Математическое обоснование задачи и подготовка к реализации

## 3.1 Математическое обоснование задачи и составление модели для контрольного тестового примера

Одной из прикладных задач данной научной работы является составление контрольного тестового примера трассы на основе реально существующей карты. В этом параграфе мы зададим исходные данные для тестового примера и построим общую математическую модель задачи.

Дано:

граф *G* = (*V*,*E*), где *V* = [*v1*,…,*vn*] – вершины графа, из них: *v1* = *start* – вершина, из который мы обязаны начать наш маршрут, а *vn* = *finish* – вершина, в которой мы обязаны закончить наш маршрут.

*E* = [*e1*,…,*em*] – ребра графа.

*n* = 50 – количество вершин графа.

*t* = 24 часа = 1440 минут – ограничение по времени.

*N* = 9 – количество команд, участвующих в соревнованиях.

*weight1*,…,*weightm* – веса ребер в графе.

Дополнительное условие «а»:

*score1*,…,*scoren* – цена каждой вершины (задаются случайным образом)

Дополнительное условие «б»:

*v11*,…,*v1n*; *v21*,…,*v2n*;…;*v91*,…,*v9n* – 9 маршрутов на заданном графе, имеющих максимальные количества очков и удовлетворяющих временному ограничению.

Найти:

а) такой маршрут в графе, который имел бы наибольшее количество очков и не превышал бы ограничение по времени.

б) *score1*,…,*scoren*, так, чтобы 9 заданных маршрутов имели как можно меньшее различие в цене.

## 3.2 Составление алгоритма на естественном языке

Чтобы реализовать какой-либо алгоритм с помощью технических средств, в первую очередь нужно хорошо представлять себе сам алгоритм, хотя бы на естественном языке. Для решения задачи «а» был создан генетический алгоритм, составленный на основе первого алгоритма, рассмотренного в главе 2. Ключевые различия между алгоритмами – алгоритм, представленный в статье, работает с циклами, то есть начальная и конечная точка полученного на выходе маршрута совпадают. Также, в новом алгоритме используется другой критерий останова. Ознакомиться с алгоритмом можно в конце этого параграфа (Рис.1).

Разберемся подробнее в каждом этапе данного алгоритма.

1. Инициализация

Заранее выбираем, сколько мы хотим членов популяции. Член популяции – это маршрут. По количеству членов популяции выполняем: включаем в новое подмножество вершин старт и финиш. В случайном порядке добавляем в это подмножество случайные вершины (не все). Создаем маршруты.

1. Улучшение маршрутов

Здесь мы пытаемся сделать имеющиеся маршруты более оптимальными – оставляем те же вершины, что принадлежат конкретному маршруту, и с помощью описанного в предыдущей главе алгоритма Лина-Кернигана пытаемся найти такой порядок вершин, с которым маршрут становится короче.

1. Оператор удаления

Данный этап в алгоритме нужен для того, чтобы если конкретный маршрут не соответствует временному ограничению, мы могли бы подкорректировать его – оператор служит для удаления вершины из маршрута. Чтобы определить, какую именно вершину лучше всего удалить с наименьшими потерями для итоговой суммы баллов, используется индекс удаления. Он рассчитывается как разница в количестве очков, приходящаяся на единицу времени пути:

,

где *si* – вес текущей вершины, *dpi* – вес ребра, соединяющего предыдущую и текущую вершины, *din* – вес ребра, соединяющего текущую и следующую вершины, *dpn* – вес ребра, соединяющего предыдущую и следующую вершины. Для удаления из маршрута выбирается та вершина, индекс удаления которой минимален.

1. Оператор добавления

Идея, по которой работает данный оператор, полностью взята из статьи [4]. Из всех точек, не входящих в маршрут, выбирается одна, с наибольший индексом ценности. Индекс ценности рассчитывается по формуле:

,

где *si* – вес текущей вершины, а

.

Здесь, предыдущей и последующей вершинами являются те, между которыми было решено вставить точку в маршрут. Для того чтобы определить эти вершины изначально выбираются три ближайших вершины к рассматриваемой. Если среди них есть хотя бы одна пара смежных, то из всех пар имеющихся смежных вершин выбирается та, *addcost* которой минимален. Конечно же, новая вершина добавляется в маршрут только при условии выполнения общего временного ограничения.

1. Критерий останова

Алгоритм останавливается либо когда истекает заранее определенное время выполнения программы, либо когда все новые маршруты перестают отличаться от лучшего построенного маршрута больше чем на заранее определенную разницу.

1. Специальный критерий

В качестве специального критерия задается какое-либо число. Каждый раз, когда номер очередной итерации основного алгоритма кратен специальному критерию, алгоритм выбирает в условии «если» второй вариант. Критерий нужен, чтобы переформировывать имеющиеся популяции улучшением входящих в них маршрутов. Выбираемое число зависит от количества вершин исходного графа, а также от количества членов популяции.

1. Выбор родителей

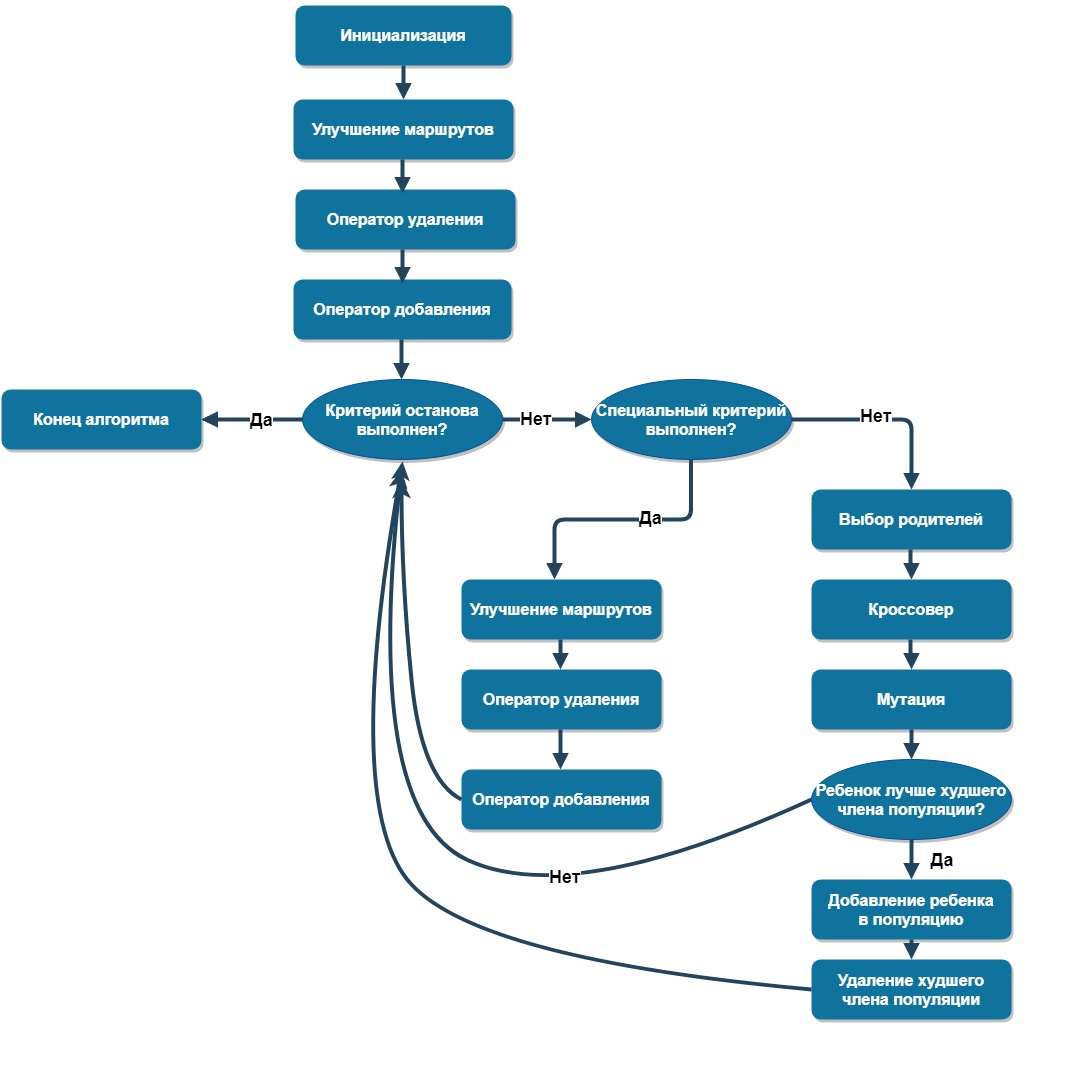
Из полученных на этапе инициализации маршрутов выбираются два. Здесь используется так называемая функция пригодности, рассмотренная в предыдущей главе:

,

где *m* = *min*(*fi*).

1. Кроссовер

Выделяются вершины, принадлежащие обоим родителям. Далее начинает строиться новый маршрут, считается, что вершины, принадлежащие только одному из родителей – особые части ребра. На каждой итерации случайным образом выбирается следующая основная вершина и путь (ребро), по которому будет строиться маршрут. Так повторяется, пока не будет включена конечная вершина. Таким образом, на выходе получаем новый маршрут, содержащий все общие вершины родителей и случайное подмножество вершин, принадлежащих только одному из родителей.



*Рис.1* Новый генетический алгоритм на естественном языке

1. Мутация

Проверяем, принадлежит ли любая точка исходного графа (не считаем здесь начальную и конечную вершины) текущему маршруту. Если принадлежит – удаляем ее из маршрута, если нет – добавляем, используя описанные выше операторы.

На выходе данного алгоритма получим искомый маршрут.

## 3.3 Усовершенствование алгоритма в сторону балансировки трассы

Итак, у нас есть алгоритм на естественном языке для решения подзадачи «а». В этом параграфе предложим несколько идей по усовершенствованию алгоритма для решения задачи «б».

Представленный выше алгоритм легко изменяется для новой задачи: достаточно подавать на выход не один лучший маршрут, а девять. Из этого следует и начальное условие: членов популяции всегда должно быть не менее девяти.

Представим, что по нашему графу двигается 9 команд. Все они начали из стартовой точки, но каждая из команд двигается по одной из девяти наилучших траекторий. Когда одна из команд достигает первой вершины, мы должны пересчитать все траектории заново (мы помним, что вес вершины уменьшается с каждой новой посетившей ее командой). Итак, теперь мы должны для каждой команды найти новую наиболее выгодную траекторию. При этом началом движения каждой команды теперь будет считаться та вершина, к которой они держали путь изначально.

Мы пересчитываем траектории для каждой команды каждый раз, когда какая-либо из них достигает новой вершины. За начало траекторий всегда берем те вершины, к которым команды на данный момент движутся. Для пересчета траекторий можно использовать описанный во втором параграфе данной главы алгоритм.

Таким образом, на выходе нового алгоритма мы получим 9 наилучших траекторий (маршрутов) в условиях меняющегося исходного графа. Все они укладываются в ограничение по времени, но также все имеют разные суммарные количества очков. Напомним, что нашей задачей является найти такой набор очков за вершины, при котором количества очков, набранные каждым маршрутом, будут приблизительно равны.

Высчитываем разницу между маршрутом с наибольшим количеством очков и остальными:

.

Считаем количество тех вершин в текущем маршруте, которые не входят в другие маршруты – *gi*. Добавляем каждой такой вершине по *h* баллов:

.

Теперь мы получили одинаковое количество баллов за каждый маршрут. Но при этом оптимальные траектории могли поменяться. Поэтому мы запускаем наш алгоритм с самого начала, но с новыми весами вершин. Повторяем всю эту процедуру установленное количество раз. Если после *i*-того повторения мы все еще не пришли к сбалансированной трассе (трасса считается сбалансированной в том случае, когда количества очков, полученные за каждый маршрут, приблизительно равны), меняем тактику: вместо добавления разницы к маршрутам с меньшим суммарным количеством очков, отнимаем очки у маршрутов с большим баллом относительно наихудшего из девяти полученных маршрутов. Если даже в этом случае трасса не становится сбалансированной, возвращаемся к самому началу: раскидываем случайным образом очки на вершинах и пересчитываем трассу.

# Глава 4. Практическая реализация

## 4.1 Выбор средств для реализации

Для реализации алгоритма, подробно описанного в предыдущей главе, была выбрана среда Visual Studio, так как она удобна в работе и давно известна автору. Язык программирования, на котором разрабатывался алгоритм - С++. Именно этот язык был выбран из-за наличия библиотеки The Boost Graph Library. Используя данную библиотеку, можно выбрать один из способов представления графа: матрицу смежности, матрицу инцидентности или список ребер. Также, в данной библиотеке реализован довольно большой перечень различных алгоритмов на графах, таких как алгоритм поиска в глубину или алгоритм поиска компонент связности. Помимо этого, в реализованной в данной работе программе используется стандартная библиотека шаблонов STL – Standard Template Library.

## 4.2 Описание работы программы

Программа делится на две части. В первой части задаются и визуализируются тестовые графы. Вторая часть программы – это непосредственно сам алгоритм. Графы в программе задаются с помощью матрицы смежности, где элементами матрицы являются веса ребер. На вход алгоритму подается тестовый граф: матрица смежности и отдельный массив с весами вершин, где вершина с индексом 1 обязательно должна быть стартовой, а вершина с индексом *n* – конечной. Веса вершин можно определить случайным образом либо задать вручную в первой части программы. На вход алгоритму также подается временное ограничение. На выходе программа выдает искомый маршрут в виде последовательности вершин. Отдельными функциями реализованы все этапы алгоритма, описанного в предыдущей главе на естественном языке.

## 4.3 Составление тестовых графов

Для того чтобы протестировать работу составленного алгоритма, были составлены десять различных графов, из них граф с наибольшим количеством вершин – это контрольный тестовый пример. В контрольном графе 50 вершин, в остальных – от 7 до 30 вершин. Все графы – связные и неориентированные. Для каждого графа были установлены веса ребер (время), в случайном порядке – очки за посещение каждой вершины, а также ограничение по времени. Ограничение по времени для каждого графа определялось свое – методом подбора. Главный тестовый образец составлялся на основе реальной карты севера Ленинградской области. Были выбраны контрольные пункты – вершины графа, а также старт и финиш. Контрольные пункты были отмечены в сервисе «Яндекс. Карты». С изображением карты севера Ленинградской области с отмеченными на ней контрольными пунктами можно ознакомиться в Приложении 1 (Рис.2). Далее, был составлен граф – ребрами на реальной карте являются дороги между контрольными пунктами, а вес ребер – это среднее время движения по данным дорогам автостопом. Важно отметить то, что в данной постановке задачи не соблюдается «неравенство треугольника» - длина любой стороны треугольника всегда меньше суммы двух других его сторон. Здесь же вовсе необязательно, что если дорога А короче дороги Б, то время прохождения дороги А меньше времени прохождения дороги Б: это зачастую зависит от класса дороги, от количества машин на ней и так далее. Для данного графа, также как и для остальных, каждой вершине в случайном порядке присвоили вес – бонусное количество очков за посещенный контрольный пункт. Со всеми графами можно ознакомиться в Приложении 2 (Рис. 3 – Рис. 12).

Программа для визуализации тестовых графов создавалась на языке С++ в среде Visual Studio. Было решено разрабатывать алгоритм для визуализации самостоятельно и не пользоваться готовыми для дальнейшего удобства работы с программой: все алгоритмы в одном пакете.

## 4.4 Оценка работы алгоритма

С результатами работы алгоритма на различных тестовых графах можно ознакомиться в Приложении 3 (Табл.2). Очевидно то, что чем меньше количество вершин графа, тем меньше время работы программы.

В задачах, так или иначе связанных с обходом графа, особенно в случае большого количества вершин, довольно сложно, а порой и невозможно, просчитать правильный ответ вручную. В связи с этим, алгоритмы, как правило, сравниваются между собой для оценки их пригодности или эффективности.

В качестве критериев оценивания работы вышеописанного алгоритма были выбраны такие параметры как скорость нахождения удовлетворяющего всем условиям пути, количество итераций, которые проходит алгоритм на протяжении работы с одним графом, а также суммарное количество очков, полученное на найденном маршруте (критерий корректной работы алгоритма). По сравнению с алгоритмами, описанными в статьях, реализованный в данной работе алгоритм имеет меньшее время поиска удовлетворяющего маршрута и, в среднем, такое же количество итераций, как и другие алгоритмы. Оценка работы алгоритма по суммарному количеству очков проводилась на основе критерия доверия: если алгоритм правильно работает на простых тестах, то, с большой вероятностью, он также правильно будет работать на более сложных тестах. В связи с этим, результаты работы алгоритма на графах с небольшим количеством вершин проверялись вручную. Алгоритм тестировался на графах с большим количеством вершин уже после проверки его работы на простых экземплярах. Тестирование на более сложных графах проводилось также и для оценки времени работы алгоритма, где, как уже было указано выше, алгоритм проявил себя лучше алгоритмов, представленных в вышеперечисленных научных работах.

# Выводы

В данной дипломной работе были поставлены и решены следующие задачи:

1. Сбор и изучение уже существующей информации по теме проблемы ориентирования, а также изучение уже имеющихся алгоритмов;
2. Разработка и реализация алгоритма максимального обхода графа за фиксированное время;
3. Предложение по усовершенствованию созданного алгоритма с целью расчета баллов за контрольные пункты в трассах спортивного ориентирования либо автостопа или рогейна на естественном языке;
4. Составление тестовых примеров и тестирование реализованного алгоритма с их помощью. Контрольный тестовый пример составлен на основе реальной карты. Дополнительно была разработана программа для визуализации тестовых графов;
5. Анализ работы реализованного алгоритма на основе результатов тестирования и сравнения его работы с другими существующими алгоритмами

Реализованный в данной работе алгоритм справляется со своей основной целью – поиском в графе маршрута с максимальным количеством баллов, соответствующего временным ограничениям. Представленный алгоритм прост в реализации и легок для понимания, а также способен конкурировать с различными алгоритмами из научной литературы.

# Заключение

В настоящее время проблема ориентирования является достаточно актуальной. Различные навигационные средства, мобильные и интернет приложения зачастую вынуждены решать задачу коммивояжера или ее различные модификации. Задачи на графах часто встречаются не только в сферах спорта и туризма, с которыми тесно связана данная научная работа, но и в различных проблемах организации и менеджмента. Автоматизированное решение различных вариаций проблемы ориентирования могло бы помочь составителям расписаний различных обходов (например, дежурные на складе) или службам доставки.

Важным замечанием является то, что в области задач, связанных с ориентированием, до сих пор не существует многих решений. Так, задача коммивояжера при большом количестве городов может быть решена за приемлемое время только эвристическими методами – методами, дающими лишь приближенное решение. Многие задачи решаемы лишь в конкретной постановке: под известные не меняющиеся ограничения можно создать алгоритм решения. Проблема ориентирования также относится к таким задачам и в настоящее время для ее решения используются преимущественно эвристические методы.

В данной дипломной работе проблема ориентирования была рассмотрена с разных сторон. Из литературных источников был выбран и доработан алгоритм максимального обхода графа за фиксированное время. Поставлена и выполнена конкретная задача с разработкой итогового тестового графа для проверки работы алгоритма на примере реально существующей карты севера Ленинградской области. Проведен анализ реализованного алгоритма и его результатов. Все цели и задачи, поставленные в начале данной работы, были достигнуты и выполнены.

В качестве перспектив развития и дальнейшей работы можно выделить такие задачи как:

1. Реализация второй части поставленной задачи – алгоритма, который рассчитывал бы баллы для контрольных пунктов в трассах спортивного ориентирования;
2. Реализация приложения, решающего задачу максимального обхода графа за ограниченное время, с целью использования в профессиональной среде;
3. Создание новых тестовых примеров и анализ работы приложения с их помощью. Доработка выявленных в процессе слабых мест приложения;
4. Относительно уже реализованного алгоритма среди перспектив развития можно выделить создание новых операторов для расширения круга задач, решаемых алгоритмом.

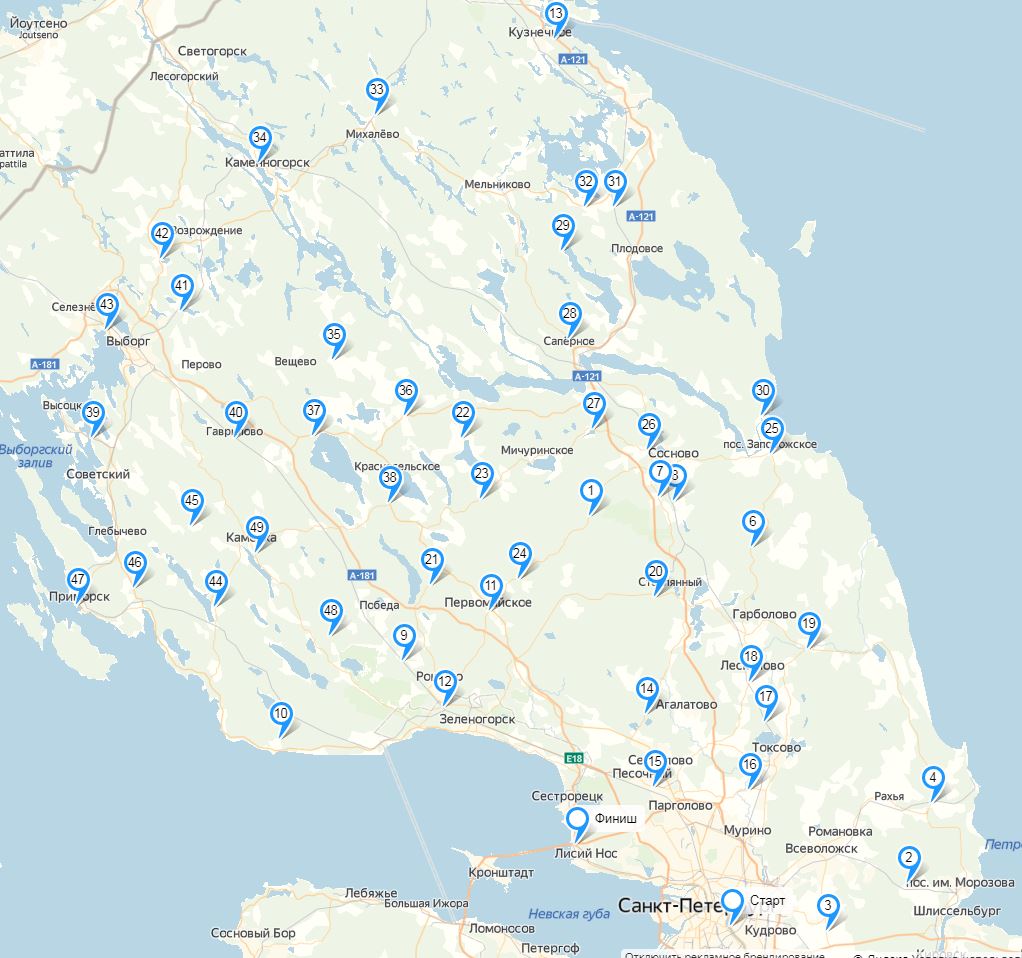
К сожалению, в настоящее время большая часть научной литературы по проблеме ориентирования написана на английском языке. Поэтому, помимо всего прочего, в дальнейшие перспективы работы можно также внести популяризацию проблемы в русскоязычных источниках и перевод иностранных статей на русский язык.

# Список литературы

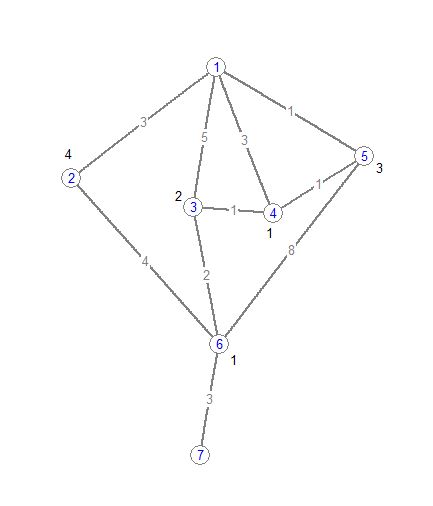
1. Gunawan A., Lau H.C., Vansteenwegen P. Orienteering problem: a survey of recent variants, solution approaches and applications // European journal of operational research, 2016. P. 315 – 332
2. Vansteenwegen P., Souffriau W., Oudheusden D.V. The orienteering problem: a survey // European journal of operational research, 2011. P. 1 – 10
3. Chao I-M., Golden B.L., Wasil E.A. A fast and effective heuristic for the orienteering problem // European journal of operational research, 1996. P. 475 – 489
4. Kobeaga G., Merino M., Lozano J.A. An efficient evolutionary algorithm for the orienteering problem // Computers and Operations research, 2018. P. 42 – 59
5. Tasgetiren M.F. A genetic algorithm with an adaptive penalty function for the orienteering problem // Journal of economic and social research, 2002. P. 1 – 26
6. Ke L., Zhai L., Li J., Chan F.T.S. Pareto mimic algorithm: an approach to the team orienteering problem // Omega, 2016. P. 155 – 166
7. Montemanni R., Gambardella L.M. An ant colony system for team orienteering problems with time windows // Foundations of Computing and Decision Sciences, 2009. P. 287 – 306
8. Bianchessi N., Mansini R., Speranza M.G. A branch-and-cut algorithm for the Team Orienteering Problem // International Transactions in Operational Research, 2018. P. 627 – 635
9. Boussier S., Feillet D., Gendreau M. An exact algorithm for team orienteering problems // 4OR quarterly journal of the Belgian, French and Italian Operations Research Societies, 2007. P. 211 – 230
10. Keshtkaran M., Ziarati K., Bettinelli A., Vigo D. Enhanced exact solution methods for the team orienteering problem // International journal of production research, 2016. P. 591 – 601
11. Gunawan A., Lau H.C., Vansteenwegen P., Lu K. Well-tuned algorithms for the team orienteering problem with time windows // Journal of the operational research society, 2017. P. 861 – 876
12. Gavalas D., Konstantopoulos C., Mastakas K., Pantziou G., Vathis N. Efficient metaheuristics for the mixed team orienteering problem with time windows // International Conference on Applied Algorithms, 2014. P. 152 – 163
13. Alvarez-Miranda E., Luipersbeck M., Sinnl M. Gotta (efficiently) catch them all: Pokemon GO meets orienteering problems // European journal of operational research, 2018. P. 779 – 794
14. Gavalas D., Konstantopoulos C., Mastakas K., Pantziou G., Vathis N. Approximation algorithms for the arc orienteering problem // Information Processing Letters, 2015. P. 313 – 315
15. Ilhan T., Iravani S.M.R., Daskin M.S. The orienteering problem with stochastic profits // Industrial engineering and management sciences, 2008. P. 406 – 421
16. Chekuri C., Korula N., Pal M. Improved algorithms for orienteering and related problems // ACM transactions on algorithms, 2008. P. 661 – 670
17. Blum A., Chawla S., Karger D.R., Lane T., Meyerson A., Minkoff M. Approximation algorithms for orienteering and discounted-reward TSP // 44th annual IEEE symposium on foundations of computer science, 2003.
18. Penicka R., Faigl J., Vana P., Saska M. Dubins Orienteering Problem // Robotics and automation letters, 2017. P. 1210 – 1217
19. Pietz J., Royset J.O. Generalized orienteering problem with resource dependent rewards // Naval research logistics, 2013. P. 294 – 312

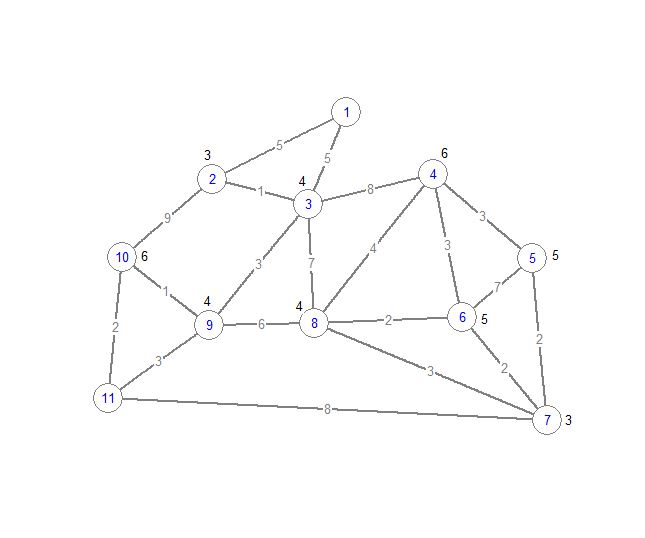
# Приложения

## Приложение 1. Карта-трасса для соревнований с контрольными пунктами

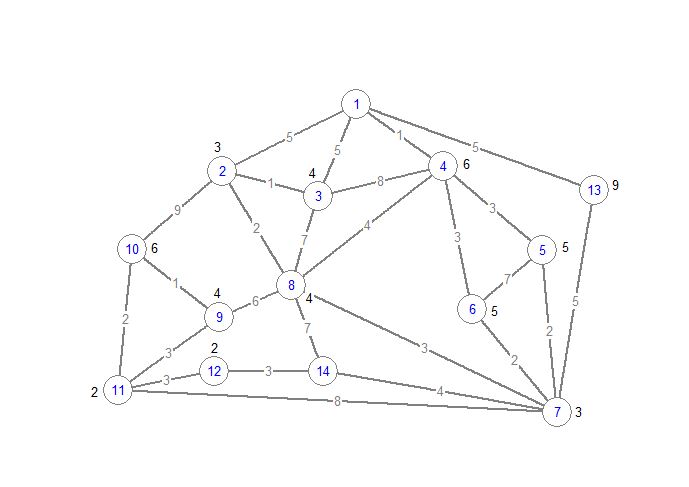
*Рис.2* Карта севера Ленинградской области с отмеченными контрольными пунктами

## Снимок.JPG3.JPGПриложение 2. Тестовые графы

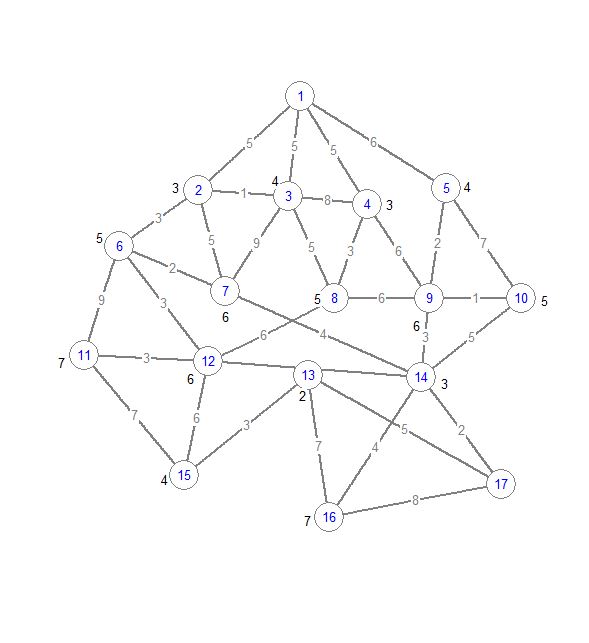
*Рис. 3* Тестовый граф № 1 *Рис. 4* Тестовый граф № 2

**

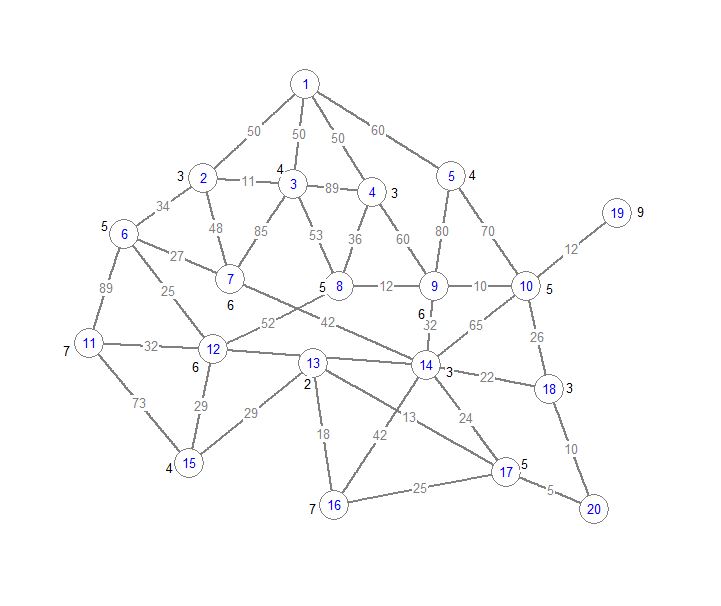
*Рис. 5* Тестовый граф № 3

**

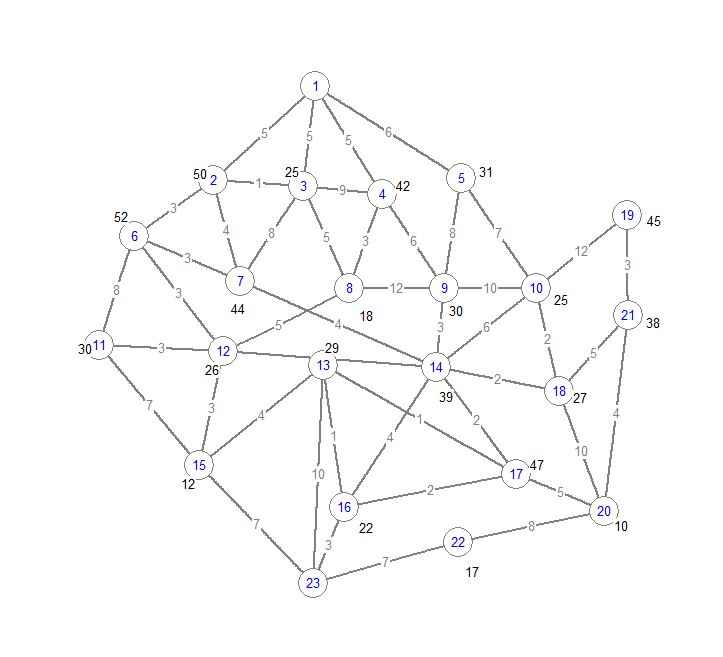
*Рис. 6* Тестовый граф № 4

**

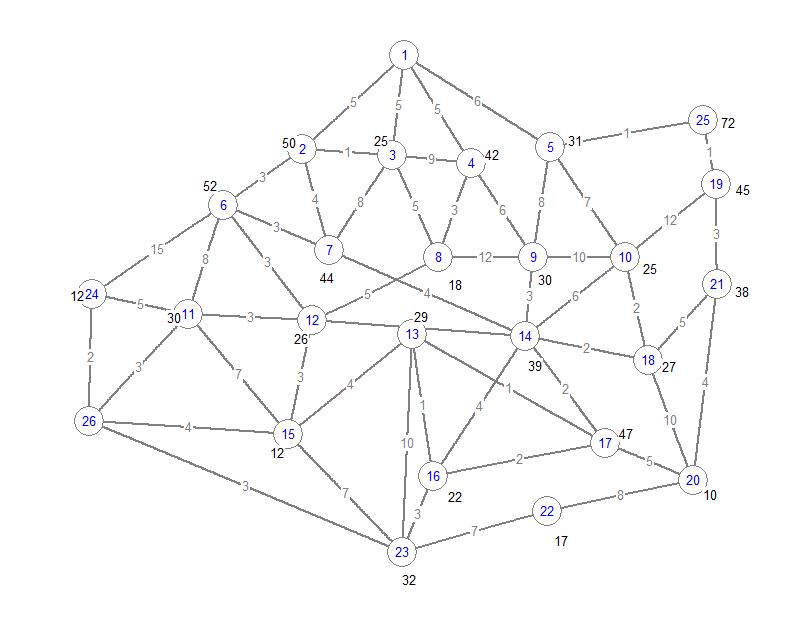
*Рис. 7* Тестовый граф № 5

**

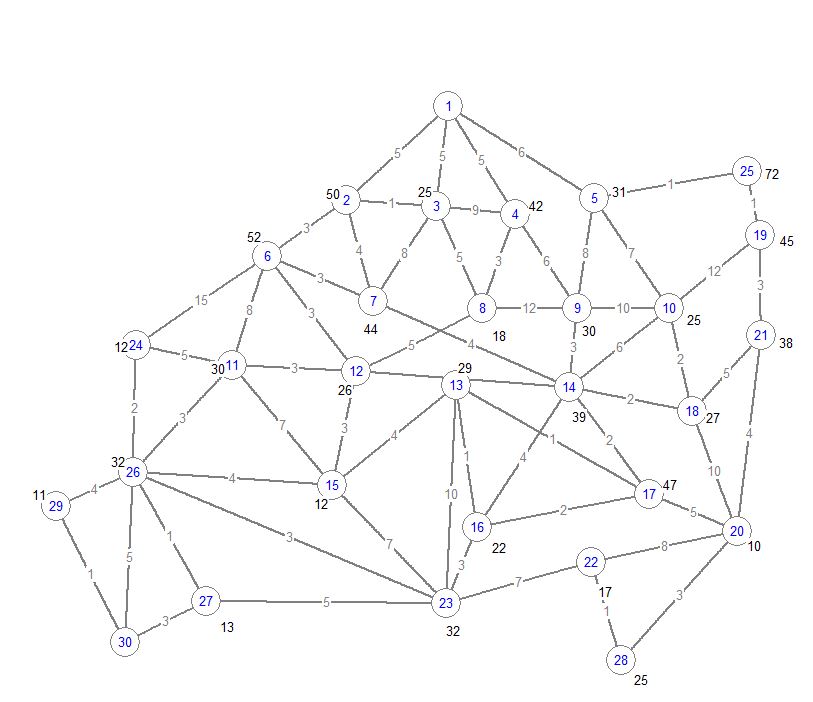
*Рис. 8* Тестовый граф № 6

**

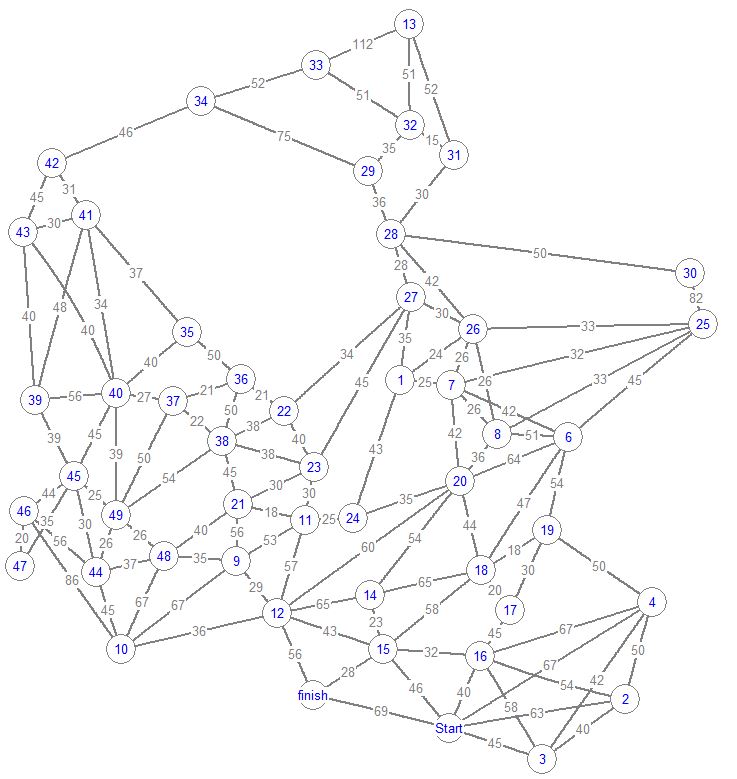
*Рис. 9* Тестовый граф № 7

**

*Рис. 10* Тестовый граф № 8

**

*Рис. 11* Тестовый граф № 9

*Рис.12* Контрольный тестовый экземпляр на основе карты из Приложения 1, тестовый граф № 10

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № вершины | 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| Количество очков | 12 | 8 | 6 | 6 | 11 | 5 | 6 | 8 | 8 | 4 | 6 | 9 | 7 | 6 | 5 | 6 |

*Табл.1* Количество очков на вершинах контрольного тестового графа

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № вершины | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
| Количество очков | 5 | 8 | 6 | 7 | 7 | 4 | 6 | 8 | 6 | 2 | 6 | 8 | 11 | 7 | 4 | 6 |

*Табл. 1, продолжение* Количество очков на вершинах контрольного тестового графа

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № вершины | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 |
| Количество очков | 9 | 9 | 4 | 5 | 6 | 7 | 5 | 6 | 8 | 7 | 5 | 7 | 5 | 6 | 5 | 4 |

*Табл. 1, продолжение* Количество очков на вершинах контрольного тестового графа

## Приложение 3. Результаты работы алгоритма на тестовых графах

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № тестового графа | Количество вершин в графе | Время работы программы (сек.) | Сумма очков итогового маршрута | Длина итогового маршрута | Временное ограничение |
| 1 | 7 | ~1 | 7 | 8 | 9 |
| 2 | 8 | ~1 | 9 | 15 | 15 |
| 3 | 11 | ~1 | 17 | 12 | 21 |
| 4 | 14 | ~2 | 14 | 10 | 10 |
| 5 | 17 | ~3 | 18 | 19 | 20 |
| 6 | 20 | ~3 | 26 | 160 | 180 |
| 7 | 23 | ~3 | 279 | 23 | 25 |
| 8 | 26 | ~5 | 183 | 18 | 20 |
| 9 | 30 | ~8 | 272 | 26 | 27 |
| 10 | 50 | ~14 | 113 | 1339 | 1440 |

*Табл. 2* Результаты работы алгоритма