

**Санкт–Петербургский государственный университет**

***ВАНЬКОВИЧ Сергей Витальевич***

**Выпускная квалификационная работа**

***Разработка инструментов анализа потоков на  
транспортной сети***

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5005.2020 «Прикладная  
математика, фундаментальная информатика и программирование»

Научный руководитель:

Кандидат физико-математических  
наук, доцент кафедры  
математической теории  
экономических решений  
Раевская А. П.

Рецензент:

кандидат физико-математических  
наук, старший преподаватель  
кафедры теории управления  
Зараник Ульяна Петровна

Санкт-Петербург

2024 г.

# Содержание

<b>Введение</b> . . . . .	4
<b>Глава 1. Постановка задачи.</b> . . . . .	5
1.1. Сбор картографических данных . . . . .	6
1.2. Анализ картографических данных . . . . .	7
1.3. Математическая модель . . . . .	7
<b>Глава 2. Сбор данных с сервиса OpenStreetMap</b> . . . . .	8
2.1. OpenStreetMap API . . . . .	8
2.2. OSM формат данных . . . . .	9
2.3. Используемые значения тегов OSM . . . . .	10
2.4. Сбор данных в формате OSM с сервиса bbbike . . . . .	15
<b>Глава 3. Анализ полученных данных</b> . . . . .	17
3.1. Вычисление перекрёстков . . . . .	17
3.2. Кластеризация перекрёстков . . . . .	20
3.3. Вычисление количества людей, выходящих из здания или входящих в него . . . . .	23
3.4. Привязка числа людей от помещения к перекрёстку . . . . .	26
3.5. Нахождение наикратчайшего маршрута и его длины между каждой парой перекрёстков . . . . .	28
<b>Глава 4. Математическая модель</b> . . . . .	33
4.1. Матрицы корреспонденции . . . . .	33
4.2. Гравитационная модель . . . . .	33
4.3. Энтропийная модель . . . . .	34
4.4. Итерационный алгоритм построения матрицы корреспонденции гравитационным методом . . . . .	36
<b>Глава 5. Результаты</b> . . . . .	38
5.1. Статистические показатели матрицы корреспонденции . . . . .	38
5.2. Визуализация матрицы корреспонденции . . . . .	38
<b>Выводы</b> . . . . .	40
<b>Заключение</b> . . . . .	41

<b>Список литературы</b>	<b>43</b>
--------------------------	-----------

## Введение

С развитием транспорта, инфраструктуры городов и автомобилизацией населения в крупных и средних городах возникают проблемы с пропускной способностью транспортной сети. С каждым днем проблема пробок на дорогах становится все более заметной и требует срочного внимания. Увеличение числа авто и интенсивности дорожного движения вызывает переполнение транспортных артерий, что приводит к износу дорожного покрытия и удлинению времени на дороге. Чтобы решить этот вопрос, необходимо применение многогранного подхода и объединение усилий правительства, населения и водителей. Важно внедрять современные технологии и улучшать инфраструктуру, налаживать расписание движения транспорта и больше использовать возможности общественного транспорта, чтобы снизить давление на дорожную сеть и обеспечить удобство путешествий для всех участников дорожного движения. Поэтому возникает потребность в анализе и оптимизации сети путей сообщения.

Анализ транспортной сети города играет ключевую роль в планировании и развитии урбанистической инфраструктуры. Понимание транспортных потоков помогает снизить пробки, уменьшить время в пути и повысить общую эффективность транспортной системы. Анализ дорожной сети может выявить опасные участки дорог, где происходит много аварий, что позволит принять меры по улучшению безопасности. Также, изучение коммуникационной системы города может помочь снизить выбросы вредных веществ от транспорта, определяя наиболее эффективные пути и способствуя разработке маршрутов для общественного транспорта, который экологичнее личных автомобилей. Хорошо спланированная транспортная сеть обеспечивает доступность всех районов города и помогает интегрировать социально и экономически маргинализированные группы населения.

В зависимости от размера и сложности города, анализ транспортной сети может быть простым или включать сложные моделирование и симуляцию. Успешный анализ и его последующее применение способствует созданию устойчивых и функциональных городских сред, обеспечивающих высокое качество жизни и стимулирующих экономическое развитие.

## Глава 1. Постановка задачи.

Одним из основных аспектов анализа системы транспортных путей является моделирование отношений между узлами сети. Главными параметрами таких моделей являются выбор узлов и выбор метода построения отношений между узлами.

Для выбора узлов транспортной сети можно воспользоваться административным делением города. При этом для более крупных административных единиц полученные модели будут иметь меньший размер, чем для более мелких единиц. К примеру, в городе Петрозаводске насчитывается 29 районов и 527 улиц. Таким образом, модель, узлами которой выбраны районы, будет являться обобщенной моделью к той, у которой узлами выбраны улицы.

С другой стороны, узлами коммуникационной системы могут быть перекрёстки дорог. Они имеют более естественную интерпретацию в контексте данной задачи, потому что по своей сути являются вершинами графа транспортной системы. Модель, основанная на перекрёстках, гораздо крупнее аналогичной модели, основанной на административных единицах. Развивая предыдущий пример, в Петрозаводске насчитывается примерно 983 перекрёстков.

Для моделирования отношений между узлами сети используются так называемые матрицы корреспонденций. Матрица корреспонденций в транспортном планировании — это квадратная матрица, которая отображает количество поездок между всеми парами зон в изучаемом регионе за определенный период времени. Каждая строка и столбец матрицы соответствуют определенной зоне или местоположению в зоне транспортного анализа.

Структура матрицы такова:

- Колонки представляют точки отправления.
- Строки представляют точки назначения.
- Каждая ячейка (пересечение строки и столбца) отображает количество поездок от конкретной начальной точки к конкретной конечной точке.

Эта матрица является ключевым компонентом в моделях транспортного спроса и используется для анализа паттернов перемещения людей или грузов, а также для прогнозирования изменений в транспортной системе при введении новых дорог, транспортных режимов или изменениях в зонировании.

## **1.1 Сбор картографических данных**

Устройство дорожной сети города – это комплексная инфраструктура, которая обеспечивает движение транспортных средств и пешеходов, соединение различных частей города, а также доступ к разнообразным объектам и услугам. Основные элементы дорожной сети включают в себя дороги и перекрёстки.

Каждый из этих элементов выполняет свою специфическую функцию и важен для обеспечения безопасности и комфорта пользователей. Дороги могут быть различных категорий – от магистралей до улиц местного значения, при этом они должны соответствовать определённым стандартам ширины проезжей части, покрытия и освещения.

Перекрёстки – это ключевые точки в дорожной сети, где пересекаются или сходятся дорожные потоки, и их эффективность напрямую влияет на пропускную способность всей дорожной системы. Они могут быть регулируемыми с помощью светофоров или нерегулируемыми, оборудованными знаками или дорожной разметкой.

Информацию о количестве людей в какой-нибудь ограниченной области города несут в себе здания, расположенные в этой области. Эти данные могут быть получены через наблюдения за плотностью и функциональным назначением зданий, что позволяет оценить потенциальную загруженность подразделений городской инфраструктуры и интенсивность человеческих потоков. Например, большое количество офисных зданий в определенной части города указывает на высокую концентрацию работников в рабочие часы, в то время как наличие жилых комплексов предполагает большую активность в утренние и вечерние часы.

Таким образом, этап сбора данных включает в себя получение данных

о дорогах, перекрёстках и зданиях города. Обработка и анализ всех этих данных необходимы для разработки эффективных решений по улучшению дорожной сети и повышению уровня комфорта и безопасности жителей города. Кроме того, обновленные данные помогают муниципалитетам реагировать на меняющиеся потребности граждан и динамично развивающуюся городскую инфраструктуру.

## 1.2 Анализ картографических данных

Для успешного построения модели необходимо произвести анализ и обработку данных, полученных на предыдущем этапе. Анализ включает в себя агрегацию данных, расчёт показателей выезжающих из каждого перекрёстка и въезжающих в него человек и представление полученных результатов в удобном для построения модели виде. Процесс обработки данных и их анализа является важным для создания точных и реалистичных моделей, которые могут быть использованы для оптимизации городской дорожной сети и для планирования развития городской инфраструктуры.

## 1.3 Математическая модель

Проблема определена следующим образом: Для заданного набора узлов  $V$ , характеризующихся  $V_i^{out}$  - число людей, выезжающих из узла  $V_i$  и  $V_i^{in}$  - число людей въезжающих в узел  $V_i$  построить матрицу корреспонденций  $T$ . При этом для расчёта  $T$  используются:

- $V^{out}, V^{in}$  - вектора, описывающие выезд и въезд для заданного набора узлов,

- $c_{ij}$  - расстояние от узла  $V_i$  до узла  $V_j$  по рёбрам транспортной сети.

В общем виде элементы матрицы  $T$  рассчитываются следующим образом:

$$T_{ij} = F(V_i^{out}, V_j^{in}, c_{ij}).$$

## Глава 2. Сбор данных с сервиса OpenStreetMap

### 2.1 OpenStreetMap API

OpenStreetMap (OSM) — это проект, созданный с целью создания свободной и открытой картографической базы данных мира [2]. Главной идеей OSM является предоставление пользователям доступа к географическим данным, которые могут быть использованы в различных приложениях, начиная от карт и навигационных систем, и заканчивая геоаналитическими исследованиями. Основные характеристики OpenStreetMap:

- Свободная и открытая база данных: Данные OpenStreetMap доступны под лицензией ODbL (Open Database License), что позволяет пользователям свободно использовать, изменять и распространять картографические данные.
- Коллективное создание данных: Карты OSM создаются сообществом участников, которые вносят свой вклад, добавляя новые объекты, исправляя ошибки и обновляя существующую информацию. Любой желающий может присоединиться к этому сообществу и стать участником проекта.
- Разнообразие данных: OSM содержит разнообразные географические данные, такие как дороги, здания, реки, железные дороги, границы административных единиц, места отдыха и многое другое. Эта многообразная информация делает OSM полезным инструментом для различных приложений.
- Открытые API: OpenStreetMap предоставляет открытые программные интерфейсы (API), что позволяет разработчикам интегрировать карты OSM в свои приложения и сервисы.
- Использование в различных областях: Карты OpenStreetMap активно используются в различных областях, таких как навигация, туризм, геоинформационные системы, гуманитарная помощь в чрезвычайных ситуациях и многие другие.



Участие в проекте OpenStreetMap доступно каждому, и каждый может внести свой вклад в улучшение карты. Официальный сайт проекта (<https://www.openstreetmap.org>) предоставляет инструменты для просмотра карт, редактирования данных и получения информации о том, как присоединиться к сообществу OSM.

## 2.2 OSM формат данных

Формат данных OpenStreetMap (OSM) представляет собой структурированный XML-формат, предназначенный для хранения географической информации. Он используется для описания географических объектов, таких как дороги, здания, реки и другие элементы, на карте мира. Каждый объект в OSM представлен в виде элемента XML. Основные компоненты формата данных OSM:

**osm** элемент. Это корневой элемент, содержащий все другие элементы. Содержит атрибуты, такие как `version` (версия данных), `generator` (инструмент, использованный для создания данных) и другие.

### Листинг 1: "osm" элемент

```
1 <osm version="0.6" generator="OpenStreetMap server">
2   <? other elements ?>
3 </osm>
```

**node** элемент. Представляет точку на карте с уникальным идентификатором, широтой, долготой и другими атрибутами. Пример:

### Листинг 2: "node" элемент

```
1 <node id="123456789" lat="12.3456" lon="78.9012"
2 version="1" changeset="123456">
3   <? other elements ?>
4 </node>
```

**way** элемент. Представляет линию, состоящую из узлов (точек). Описывает, например, дороги, реки или границы. Включает в себя список узлов и другие атрибуты.

### Листинг 3: "way" элемент

```

1 <way id="987654321" version="2" changeset="987654">
2   <nd ref="123456789"/>
3   <nd ref="123456787"/>
4   <? other elements ?>
5 </way>

```

**relation** элемент. Представляет отношение между объектами, такое как группа узлов или путь, образующий, например, границу административной единицы. Включает в себя элементы `member`, представляющие связанные объекты, и другие атрибуты.

Листинг 4: "relation" элемент

```

1 <relation id="135792468" version="3" changeset="135792">
2   <member type="way" ref="987654321" role="outer"/>
3   <member type="node" ref="345678901" role="admin_centre"/>
4   <? other elements ?>
5 </relation>

```

**tag** элемент. Представляет собой пару "ключ-значение" и используется для атрибутов объектов (`node`, `way`, или `relation`).

Листинг 5: "tag" элемент

```

1 <tag k="highway" v="residential"/>

```

Это лишь общий обзор основных элементов формата данных OSM. Данные могут содержать другие типы элементов, а атрибуты и значения ключей/значений могут различаться в зависимости от типа объекта.

## 2.3 Используемые значения тегов OSM

В данном исследовании для поиска узлов графа транспортной сети интерес представляют автомобильные дороги и перекрёстки этих дорог. За обозначение дорог в формате OSM отвечает тег с ключом **"highway"**. В качестве автомобильных дорог выступают:

- **motorway** — Автомагистрали (обозначенные знаком "Автомагистраль") — автомобильные дороги, имеющие многополосную проезжую часть с

центральной разделительной полосой, и не имеющие пересечений в одном уровне с другими автомобильными и железными дорогами, трамвайными путями, велосипедными и пешеходными дорожками.

- **trunk** — Важные дороги, не являющиеся автомагистралями.
- **primary** — Автомобильные дороги регионального значения, соединяющие города и/или областные центры, дороги межрегионального значения, не являющиеся trunk. В населённых пунктах — центральные магистрали городов (в небольших населённых пунктах могут отсутствовать).
- **secondary** — Автомобильные дороги областного значения, соединяющие областные центры с крупными населёнными пунктами (районными центрами), а также крупные населённые пункты между собой. В населённых пунктах — основные магистрали районов города (в небольших населённых пунктах могут отсутствовать).
- **tertiary** — Более важные автомобильные дороги среди прочих автомобильных дорог местного значения, например соединяющие районные центры с сёлами, а также несколько сёл между собой. В населённых пунктах — основные микрорайонные или межмикрорайонные транзитные улицы.
- **residential** — Дороги, которые проходят внутри жилых зон, а также используются для подъезда к ним. В основном используется теми людьми, которые живут на этой улице или на примыкающей. Обычно имеют название.

Для анализа количества людей выезжающих с перекрёстка или въезжающих в него были использованы теги с ключами "**building**", "**amenity**", "**shop**". Ключ **building** используется для обозначения здания. В OpenStreetMap здание — это искусственная конструкция с крышей, более или менее постоянно стоящая на одном месте. **amenity** — тег верхнего уровня, описывающий полезные и важные объекты для посетителей и местных жителей (такие как

туалеты, телефоны, банки, аптеки и школы). Тегом **shop** обозначается место розничной торговли или оказания услуг.

В качестве используемых значений **building** для жилых помещений были выбраны:

- **house** — Индивидуальный жилой дом для проживания одной, реже нескольких семей.
- **apartments** — Многоквартирный жилой дом.
- **detached** — Частный отдельный жилой дом, обычно на одну семью.
- **dormitory** — Общежитие. Здание, используемое студентами колледжа или университета (но не общая комната для нескольких человек, как подразумевается словом "dormitory" в британском английском).
- **hotel** — Здание гостиницы.
- **residential** — Общий тег для жилых зданий.
- **semidetached\_house** — Жилой дом, который разделён на 2 половины общей стеной (рассчитанный на 2 семьи, проживающих с разных сторон). Обычно на американском английском языке называется «дуплекс».

В качестве используемых значений **building** для нежилых помещений были выбраны:

- **kindergarten** — Любой тип детских садов.
- **school** — Любой тип школьных зданий.
- **college** — Колледж.
- **university** — ВУЗ, университет.
- **office** — Офисное здание.

- **government** — Правительственные зданий в целом, включая муниципальные, провинциальные и административные управления, правительственные учреждения и департаменты, ратуши, (региональные) парламенты и суды.
- **public** — Общественное здание.
- **commercial** — Места деловой активности.
- **warehouse** — Складские здания.
- **industrial** — Промышленное здание.
- **retail** — Здание, используемое в основном для розничной торговли.
- **supermarket** — Супермаркет (крупный магазин самообслуживания).
- **stadium** — Стадион.
- **hospital** — Здание, корпус больницы.
- **museum** — Здание, которое было спроектировано как музей.

В качестве используемых значений **amenity** для нежилых помещений были выбраны:

- **kindergarten** — То же, что и для **building**.
- **school** — То же, что и для **building**.
- **college** — То же, что и для **building**.
- **university** — То же, что и для **building**.
- **research\_institute** — Научно-исследовательский институт является учреждением, предназначенным для проведения исследований.
- **post\_office** — Место, откуда можно получить или отправить письма и посылки.

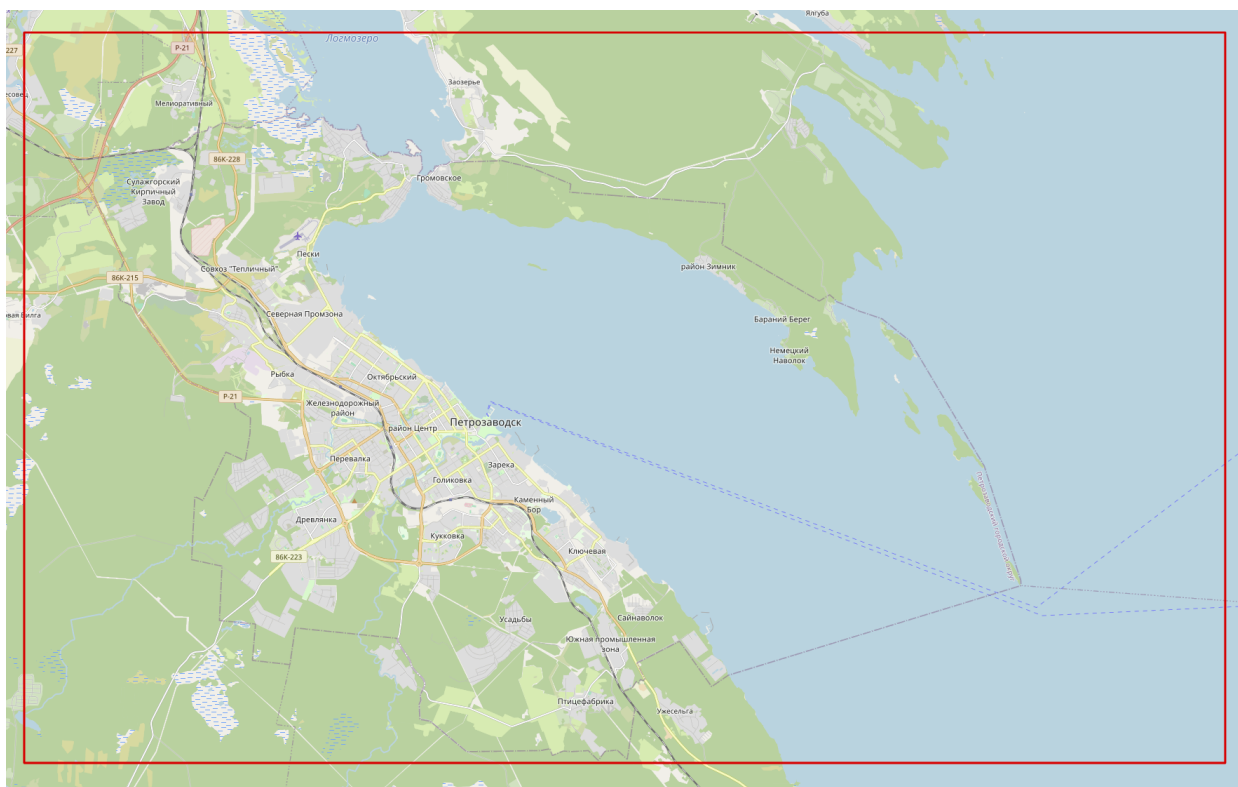
- **police** — Полицейский участок (место, которое служит основной контактной точкой для общественности, то есть полицейские участки, ориентированные на общественность).
- **courthouse** — Здание, в котором находится суд, который отправляет правосудие в соответствии с законами.
- **townhall** — Администрация населённого пункта.
- **fire\_station** — Пожарная часть (станция, с которой выезжает пожарная команда).
- **bank** — Банк.
- **cinema** — То же, что и для **building**.
- **stadium** — Стадион.
- **restaurant** — Ресторан.
- **hospital** — То же, что и для **building**.
- **clinic** — Поликлиника, клиника: медицинский центр с количеством сотрудников.
- **arts\_centre** — Центр искусств (место, где проводится широкий спектр художественных мероприятий).
- **community\_centre** — Место для проведения общественных праздников и торжеств, а также клубы по интересам.

В качестве используемых значений **shop** для нежилых помещений были выбраны:

- **department\_store** — Универсам, универмаг, торговый дом. Магазин, имеющий несколько отделов различной направленности.
- **mall** — Торговый центр, торговый комплекс, молл.
- **supermarket** — То же, что и для **building**.

## 2.4 Сбор данных в формате OSM с сервиса bbbike

Чтобы собрать данные с сервиса OpenStreetMap (OSM), было использовано API сервиса bbbike.org [3]. Среди прочих открытых API bbbike предоставляет наиболее полную информацию об объектах инфраструктуры города. С помощью интерактивного редактора был выделен и выгружен в OSM формат прямоугольный участок карты Петрозаводска с координатами левого нижнего угла (61.711, 34.164) и правого верхнего угла (61.88, 34.752). Выгруженный участок представлен на рисунке 1 красным цветом.



**Рис. 1:** Участок карты Петрозаводска, выгруженный с помощью сервиса bbbike.

В результате использования сервиса bbbike был получен OSM файл, представляющий информацию о выбранном участке карты. Первые несколько строк полученного файла представлены в листинге 6.

**Листинг 6:** Первые 13 строк OSM файла полученного с помощью сервиса bbbike

```
1 <osm version="0.6" generator="osmconvert 0.8.11" timestamp="
  2024-01-25T00:00:00Z">
2   <bounds minlat="61.711" minlon="34.164" maxlat="61.88"
     maxlon="34.752"/>
3   <node id="253541" lat="61.8002089" lon="34.314076" version="
     1">
4     <tag k="button_operated" v="no"/>
5     <tag k="crossing" v="traffic_signals"/>
6     <tag k="highway" v="traffic_signals"/>
7     <tag k="traffic_signals:direction" v="both"/>
8   </node>
9   <node id="253545" lat="61.8033524" lon="34.2998495" version="
     "1"/>
10  <node id="253546" lat="61.8038567" lon="34.2987219" version="
     "1">
11    <tag k="crossing" v="uncontrolled"/>
12    <tag k="highway" v="crossing"/>
13  </node>
14  ...
```



## Глава 3. Анализ полученных данных

### 3.1 Вычисление перекрёстков

В формате OSM не присутствует тега или атрибута, обозначающего перекрёсток. Для этого был разработан соответствующий алгоритм. В файле OSM дорога представлена тегом **way**, в котором содержатся ссылки на теги **node**, отвечающие за координаты точек, образующих ломаную линию дороги. Для примера рассмотрим автомобильную дорогу, проходящую по улице Огородной города Петрозаводска. OSM обозначение этой дороги представлено на листинге 7, а соответствующее изображение дороги улицы на рисунке 2.

**Листинг 7:** OSM обозначение дороги на улице Огородной г. Петрозаводска

```
1 <way id="34264852" version="1">
2   <nd ref="392977655"/>
3   <nd ref="392977657"/>
4   <nd ref="392977659"/>
5   <nd ref="392977661"/>
6   <nd ref="392977648"/>
7   <nd ref="392977651"/>
8   <nd ref="392977652"/>
9   <nd ref="392977654"/>
10  <tag k="highway" v="residential"/>
11  <tag k="maxspeed" v="RU:urban"/>
12  <tag k="name" v="Ogorodnaya street"/>
13  <tag k="postal_code" v="185013"/>
14 </way>
```



Рис. 2: Улица Огородная

Таким образом, суть алгоритма такова: если на точку ссылается более одной дороги, значит эта точка — перекрёсток. При таком подходе на выходе получаются все перекрёстки, но также точки соединения тегов **way** в составных дорогах. Для того чтобы отбросить лишние точки, пришлось добавить в алгоритм проверку на то, ссылаются ли теги **way** на точки, относящиеся к разным улицам. Алгоритм на языке программирования Go представлен в листинге 8.

#### Листинг 8: Алгоритм поиска перекрёстков на языке Go

```
1 // List of accepted highways
2 var AcceptedHighWays = map[string]bool{
3     "motorway": true,
4     "trunk": true,
5     "primary": true,
6     "secondary": true,
7     "tertiary": true,
8     "residential": true,
9 }
10
11 type Crossroad struct {
12     Refs int
13     Street string
14 }
15
```

```

16 func LoadCrossRoads(mapData *osm.OSM) {
17     var crossroads = map[osm.NodeID]*Crossroad{}
18
19     // Iterate on Way tags
20     for _, way := range mapData.Ways {
21         highway := way.Tags.Find("highway")
22         if AcceptedHighWays[highway] {
23
24             // Iterate on Nodes for each Way
25             for _, node := range way.Nodes {
26                 crossroad, contains := crossroads[node.ID]
27                 street := way.Tags.Find("name")
28
29                 // Check if this node was reffered from another way
30                 if contains {
31                     // Check if street name way is different from another way
32                     // reffered on this node
33                     if crossroad.Street != street {
34                         crossroads[node.ID].Refs += 1
35                     }
36                 } else {
37                     crossroads[node.ID] = &Crossroad{Refs: 1, Street: street}
38                 }
39             }
40         }
41     }
42     var nodes = osm.Nodes{}
43     for k, v := range crossroads {
44         if v.Refs > 1 {
45             node, err := FindNode(k, mapData.Nodes)
46             utils.ProcessError(err)
47             nodes = append(nodes, node)
48         }
49     }
50     p, err := json.Marshal(nodes)
51     utils.ProcessError(err)
52
53     f, err := os.Create("crossroads.json")
54     utils.ProcessError(err)

```

```

54     defer f.Close()
55     f.Write(p)
56 }

```

Результат работы алгоритма представлен на рисунке 3. Перекрёстки отмечены синими маркерами.

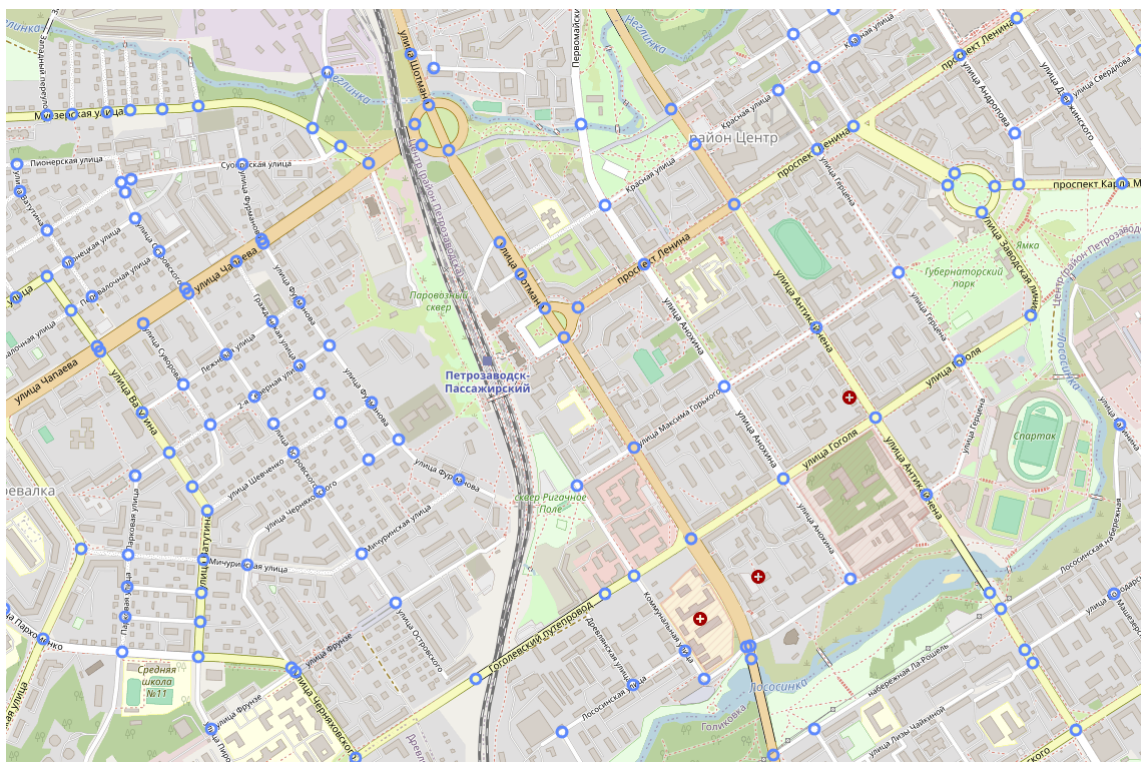


Рис. 3: Результат работы алгоритма поиска перекрёстков

## 3.2 Кластеризация перекрёстков

В процессе анализа размеченных перекрёстков выяснилось, что некоторые перекрёстки отмечены несколько раз. К примеру, на кольце отмечено количество перекрёстков, равное количеству дорог, примыкающих к кольцу. А для многополосных дорог каждое пересечение с другой дорогой на одном перекрёстке отмечается как ещё один перекрёсток. Пример с кольцевыми перекрёстками отображен на рисунке 4.

В результате было решено объединить каждый перекрёсток с ближайшим к нему перекрёстком в радиусе 55 метров.

Ошибочные метки приводили к излишнему усложнению модели и могли влиять на точность дальнейших расчетов. К тому же, соединение нескольких точек в одну улучшило общую читаемость карты и позволило более эффективно использовать данные для построения модели.

Код алгоритма представлен на листинге 9. В результате получилось сократить число перекрёстков с 1190 до 983 и избавиться от проблемы близко стоящих и накладывающих бесполезную нагрузку на модель перекрёстков.

**Листинг 9:** Алгоритм кластеризации перекрёстков на языке Go

```
1  func MergeCrossroads() {
2  var crossroads []*CrossRoad
3  LoadJson("weightedCrossroads.json", &crossroads)
4
5  var removedCrossroads = make(map[int64]struct{})
6  var leftCrossroads []*CrossRoad
7  // Iterate through all crossroads
8  for i, crossroad := range crossroads {
9      // Check if crossroad is already marked as removed
10     if _, removed := removedCrossroads[int64(crossroad.Node.ID)]; !
        removed {
11         // Iterate for all crossroads from i+1 to end to not check
            already checked crossroads
12         for _, mCrossroad := range crossroads[i+1:] {
13             if _, mRemoved := removedCrossroads[int64(mCrossroad.Node.ID)
                ]; !mRemoved {
14                 distance := DistanceTrue(
15                     [2]float64{crossroad.Node.Lat, crossroad.Node.Lon},
16                     [2]float64{mCrossroad.Node.Lat, mCrossroad.Node.Lon},
17                 )
18                 if distance <= 55 {
19                     inOutSum1 := crossroad.InWeight + crossroad.OutWeight
20                     inOutSum2 := mCrossroad.InWeight + mCrossroad.
                        OutWeight
21                     // Decide in what crossroad merge another one and mark
                        as removed
22                     if inOutSum1 < inOutSum2 {
23                         mCrossroad.InWeight += crossroad.InWeight
24                         mCrossroad.OutWeight += crossroad.OutWeight
```

```

25             removedCrossroads[int64(crossroad.Node.ID)] =
26                 struct{}{}
27         } else {
28             crossroad.InWeight += mCrossroad.InWeight
29             crossroad.OutWeight += mCrossroad.OutWeight
30             removedCrossroads[int64(mCrossroad.Node.ID)] =
31                 struct{}{}
32         }
33     }
34 }
35 }
36 }
37 }
38 // Iterate through crossroads again and left only not removed
39 for _, crossroad := range crossroads {
40     if _, removed := removedCrossroads[int64(crossroad.Node.ID)]; !
41         removed {
42         leftCrossroads = append(leftCrossroads, crossroad)
43     }
44 }
45 println(len(removedCrossroads))
46 println(len(leftCrossroads))
47 MarshallToJsonAndSave(leftCrossroads, "weightedCrossroads.json")
48 }

```



**Рис. 4:** Результат работы алгоритма кластеризации перекрёстков (слева до кластеризации, справа после)

### 3.3 Вычисление количества людей, выходящих из здания или входящих в него

Следующим этапом анализа OSM данных является подсчёт количества людей в зданиях. Количество выходящих или входящих людей в здание зависит от рассматриваемого промежутка времени. Но нас интересует в данной задаче 2 основных промежутка:

- Утро, когда люди из жилых помещений перемещаются в рабочие офисы, учебные учреждения, коммерческие помещения и т. д.
- Вечер, когда люди возвращаются домой в жилые помещения.

Таким образом, для упрощения модели было решено использовать в качестве «входящих» и «исходящих» людей – количество людей, проживающих в данном жилом помещении или, если помещение нежилое, количество людей, которые теоретически могут находиться в данном здании на протяжении дня. Тогда утром поток граждан будет направлен из жилых помещений во все остальные, а вечером в обратную сторону.

Для начала определим число людей, проживающих в жилом помещении. Законодательством РФ предусмотрены 2 вида нормы площади жилого

помещения [4]:

- Норма предоставления — это минимальный размер площади на 1 человека, который используется для определения общего метража жилья, предоставляемого по договору социального найма;
- Учетная норма — это минимально допустимый размер площади жилого помещения, на основании которого оценивается уровень обеспеченности граждан жильем. Данный норматив используется уполномоченными органами при принятии решения о постановке граждан на учет в качестве нуждающихся в улучшении жилищных условий. Размер учетной нормы не может превышать размер нормы предоставления.

Норма предоставления в Республике Карелия установлена в размере 18 м<sup>2</sup> [5]. В данном исследовании использовано именно это значение. Таким образом среднее значение людей, проживающих в жилом помещении, вычисляется

$$p = \frac{al\alpha}{b}$$

Где  $a$  — площадь помещения,  $l$  — количество этажей,  $\alpha$  — коэффициент полезной площади здания ( $\alpha < 1$  и для каждого города подбирается так, чтобы сумма всех жителей совпадала с населением города с определенной погрешностью),  $b$  — норма жилого помещения на 1 человека.

Используя теги **building** и **amenity** были отобраны жилые дома. Далее была получена площадь каждого здания и количество этажей. Подбирая коэффициент полезной площади  $\alpha$  было решено остановиться на значении 0.5. В результате суммарное число жителей оказалось равным 287 736 при реальном населении города по разным источникам около 280 000 человек.

Для нежилых помещений использовались другие формулы. В общем виде формула проста:

$$p = \frac{al}{b_{coeff}}$$

Где  $a$  — площадь помещения,  $l$  — количество этажей,  $b_{coeff}$  — норма помещения соответствующего типа для 1 человека. Для начала разберем количество людей, работающих в офисном здании. Согласно п. 20.1 МР 2.2.0244-21,



утвержденных Главным государственным санитарным врачом РФ 17.05.2021 [6], площадь на одно рабочее место пользователя ПЭВМ (независимо от количества используемых устройств отображения информации (жидкокристаллические, плазменные, LED, OLED и другие мониторы)) должна составлять не менее 4,5 м<sup>2</sup>. Это значение и используется в качестве  $b_{coeff}$  для зданий офисного типа.

Для учебных заведений (школ, колледжей и вузов) существуют нормы площади помещений на 1 человека. Нормативы площадей основных помещений образовательных организаций представлены в виде таблицы в СанПиН 1.2.3685-21 [7]. К примеру, площадь учебных помещений при организации групповых форм работы и индивидуальных занятий берется из расчета 3,5 м<sup>2</sup> на человека, помещений, оборудованных индивидуальными рабочими местами с персональным компьютером – 4,5 м<sup>2</sup> на рабочее место, мастерских трудового обучения, кабинета кулинарии и домоводства – 6 м<sup>2</sup> на рабочее место, туалетов – 0,1 м<sup>2</sup> на человека, комнат гигиены девочек – 3 м<sup>2</sup>. Актальный зал должен быть не менее 0,65 м<sup>2</sup> на посадочное место, а спортивный зал – 10 м<sup>2</sup> на человека с раздевалкой минимальной площадью 14 м<sup>2</sup>. Просуммировав все значения кроме туалетов, комнат гигиены, актового зала и раздевалки, получилось значение 6 м<sup>2</sup> на ученика. Таким образом, для учебных заведений  $b_{coeff} = 6$

Нормы на одного ребенка для детских садов составляют не менее 7,2 м<sup>2</sup> – на одного ребенка возраста до 3 лет; не менее 9 м<sup>2</sup> – на одного ребенка дошкольного возраста [8]. Для  $b_{coeff}$  было выбрано значение 8.

Так как в Петрозаводске довольно большое количество производственных зданий, для них было решено взять  $b_{coeff}$  равным 10, чтобы немного сократить их влияние на результаты модели.

В медицинских учреждениях согласно СанПиН 2.1.3.2630-10 [9] перечислены все значения для разных коек. Так как их очень много, а по данным OSM идентифицировать тип медицинского учреждения сложно, а иногда и вообще невозможно, то в качестве  $b_{coeff}$  для таких учреждений было выбрано значение 10 выражающее эвристически полученное значение и являющееся допущением в модели.

Для торговых центров и музеев не получилось найти нормативных актов

о предоставляемой площади для одного человека. К тому же посещаемость таких мест очень сильно зависит от сезона, времени суток, праздников и выходных дней. Поэтому для этих мест  $b_{coeff}$  также равен 10 и является допущением.

Также сильное влияние на модель оказывают складские помещения, у которых большая площадь, но малое количество работников. Для таких помещений  $b_{coeff}$  увеличен до 100, чтобы сократить это влияние и приблизить результаты модели к реальным.

### 3.4 Привязка числа людей от помещения к перекрёстку

На данном этапе для каждого здания определяется ближайший к зданию перекрёсток, и если здание жилое, то к  $V_{out}$  перекрёстка прибавляется число жителей данного здания. Если здание нежилое, то к  $V_{in}$  прибавляется число людей, рассчитанных для этого здания. Значения  $V_{in}$  и  $V_{out}$  могут меняться местами, в соответствии с половиной дня, для которой строится модель. В данном примере приведены значения для модели, построенной для первой половины дня, так как утром люди из жилых помещений перемещаются в нежилые (на работу, учебу, в поликлинику или торговый центр). Алгоритм привязки значений числа людей от зданий к перекрёсткам представлен в листинге 10.

**Листинг 10:** Алгоритм привязки числа людей от зданий к ближайшим перекрёсткам на языке Go

```
1 // Init crossroad weights V_in, V_out with buildings' population
2 func FindNearestCrossroads() {
3     var buildings []*Building
4     var crossroads []*osm.Node
5     var weightedCrossRoads = map[int64]*CrossRoad{}
6
7     LoadJson("buildings.json", &buildings)
8     LoadJson("crossroads.json", &crossroads)
9
10    for _, crossroad := range crossroads {
11        weightedCrossRoads[int64(crossroad.ID)] = &CrossRoad{[]*Building{},
            crossroad, 0, 0}
```

```

12     }
13
14     // Find nearest crossroad for each building
15     for _, building := range buildings {
16         minCrossRoad := crossroads[0]
17         minDistance := Distance([2]float64{building.Center.Lon, building.
            Center.Lat}, minCrossRoad.Point())
18         for _, crossroad := range crossroads {
19             distance := Distance([2]float64{building.Center.Lon, building.
                Center.Lat}, crossroad.Point())
20             if distance < minDistance {
21                 minDistance = distance
22                 minCrossRoad = crossroad
23             }
24         }
25         crossroad, ok := weightedCrossRoads[int64(minCrossRoad.ID)]
26         if ok {
27             // Append building to buildings' array of nearest crossroad
28             crossroad.Buildings = append(crossroad.Buildings, building)
29         }
30     }
31
32     var crossroadsToSave []*CrossRoad
33
34     // Calculate V_in, V_out for each crossroad
35     for _, value := range weightedCrossRoads {
36         value.InWeight = CalculateNonLivingPopulation(value.Buildings)
37         value.OutWeight = CalculateLivingPopulation(value.Buildings)
38         crossroadsToSave = append(crossroadsToSave, value)
39     }
40
41     // Save crossroads
42     MarshallToJsonAndSave(crossroadsToSave, "weightedCrossroads.json")
43 }

```

### 3.5 Нахождение наикратчайшего маршрута и его длины между каждой парой перекрёстков

При построении модели матрицы корреспонденций необходимо знать наименьшее расстояние между каждой парой узлов. А для визуализации модели нужно также получить наикратчайший маршрут. Таким образом возникает задача построения 2 матриц:

- Матрицы  $M$  ( $N \times N$ ), в которой на пересечении  $i$ -той строки и  $j$ -того столбца находится элемент, описывающий линию маршрута между перекрёстком  $i$  и  $j$ ;
- Матрицы  $D$  ( $N \times N$ ), в которой на пересечении  $i$ -той строки и  $j$ -того столбца находится длина маршрута, соответствующая линии, стоящей на той же позиции в матрице  $M$ .

Маршрут в матрице  $M$  задается с помощью строки в формате `geometry polyline`. `Geometry polyline` (геометрическая полилиния) — это способ кодирования набора координат в одну сжатую строку [10]. Этот метод широко используется в веб-картографии и GIS (геоинформационных системах) для удобной передачи информации о маршрутах и формах. `Google Maps Directions API` [11] является одним из примеров сервисов, который использует этот способ для описания маршрутов. Полилинии закодированы следующим образом:

- Координаты (широта и долгота) преобразуются в значения с плавающей точкой.
- Эти значения умножаются на определенный коэффициент (например,  $10^5$  в случае Google), округляются и преобразуются в целые числа.
- Целые числа затем преобразовываются с использованием переменной длины кодирования, которое называется кодированием переменной длины.
- Каждое значение представлено в виде последовательности символов ASCII, что делает строку хорошо сжимаемой и удобной для передачи в URL.

Для декодирования полилинии необходимо выполнить обратные шаги — раскодировать каждое значение из ASCII, привести его к исходному масштабу (на этом этапе мы получаем дельты координат) и просуммировать их с начальной точкой, чтобы получить реальные географические координаты каждого изгиба маршрута.

Этот метод особенно полезен для сжатия данных о маршрутах, так как он позволяет уменьшить размер передаваемых через интернет данных, что является важным для улучшения производительности веб-приложений. Кроме того, он предохраняет систему от избыточности данных, что особенно важно при работе с крупномасштабными картографическими сервисами.

Для построения матриц  $M$  и  $D$  был использован маршрутизатор с открытым исходным кодом OSRM. OSRM (Open Source Routing Machine) – это бесплатное программное обеспечение с открытым исходным кодом, предназначенное для вычисления маршрутов на основе данных OpenStreetMap [12]. Основная задача OSRM заключается в предоставлении возможностей поиска кратчайших путей между двумя или несколькими точками на карте, что делает его ценным инструментом для работы с геоданными, навигации и планирования маршрутов. OSRM оптимизирован для быстрых вычислений, позволяя обрабатывать запросы на поиск маршрута в режиме реального времени даже на больших дорожных сетях, что делает его подходящим для использования в веб-сервисах и мобильных приложениях. Он может обрабатывать различные типы маршрутов, включая:

- Автомобильные маршруты;
- Велосипедные маршруты;
- Пешеходные маршруты.

OSRM состоит из нескольких компонентов, включая:

- Предварительная обработка данных. При первоначальной обработке данных OpenStreetMap, OSRM строит граф дорог и применяет алгоритм вычисления кратчайших путей, часто используя алгоритмы, такие как Contraction Hierarchies, для ускорения поиска маршрутов.

- HTTP сервер. После предварительной обработки данных, OSRM может запускаться как сервер, который обрабатывает входящие HTTP запросы для построения маршрутов.
- Библиотеки. OSRM предоставляет библиотеки, которые можно интегрировать в другие программы для выполнения поиска маршрутов непосредственно из кода.
- Инструменты разработчика. OSRM включает в себя различные инструменты для разработки и тестирования маршрутов, утилиты для обработки и преобразования геоданных.

Используя OSRM, можно значительно упростить задачи связанные с логистикой, поскольку он способен не только находить маршруты, но и оптимизировать их, например, для расчета наиболее эффективного пути по нескольким точкам (Traveling Salesman Problem - TSP), что полезно для распределения доставок или выездов на вызовы. Открытый исходный код позволяет разработчикам адаптировать и улучшать OSRM в соответствии с их уникальными требованиями, а также вносить дополнения и расширения в сообщество, что способствует развитию и улучшению проекта.

Таким образом, для заполнения матриц  $M$  и  $D$  решено было развернуть 10 Docker [13] контейнеров с запущенными внутри сервисами OSRM и с помощью 10 потоков параллельно запрашивать из сервисов информацию о маршруте и дальности для каждой пары точек. Обращения к API OSRM средствами многопоточности позволяют существенно сократить время, необходимое для получения данных по всем парам точек, особенно когда работа ведется с большим объемом запросов. Распределение запросов между несколькими контейнерами Docker также увеличивает пропускную способность и устойчивость системы к отказам, обеспечивая более высокую доступность и надежность сервиса. Алгоритм для заполнения обеих матриц  $M$  и  $D$  представлен в листинге 11. Подматрица матрицы  $D$  размера  $(5 \times 5)$  представлена в выражении 1

**Листинг 11:** Алгоритм заполнения матриц  $M$  и  $D$  на языке Go

```
1 func BuildDistancesAndLinesMatrix() {
```

```

2      var crossroads []*CrossRoad
3      LoadJson("weightedCrossroads.json", &crossroads)
4
5      wg := sync.WaitGroup{}
6      n := len(crossroads)
7      nContainers := 10
8
9      // Initialize Distances matrix with zeros
10     var distances = make([][]float32, n)
11     for i := range distances {
12         distances[i] = make([]float32, n)
13     }
14     // Initialize Lines matrix with empty strings
15     var lines = make([][]string, n)
16     for i := range lines {
17         lines[i] = make([]string, n)
18     }
19
20     step := n / 10
21     start := 0
22     stop := step
23     for i := range nContainers {
24         wg.Add(1)
25         // Start thread on batch of rows
26         go func(start, stop, i int) {
27             defer wg.Done()
28             FillSubMatrix(start, stop, i, crossroads, distances, lines)
29         }(start, stop, i)
30         start = stop
31         stop += step
32     }
33     wg.Wait()
34     for _, distance := range distances {
35         fmt.Println(distance[:10])
36     }
37
38     // Save distances and lines into txt file
39     SaveDistances(distances)
40     SaveLines(lines)

```

$$\begin{pmatrix} 0 & 4611.1 & 7991 & 304.5 & 11846.1 \\ 4611.1 & 0 & 12080.1 & 4432 & 16227.9 \\ 7465 & 11566.1 & 0 & 7769.5 & 15149.9 \\ 304.5 & 4432 & 8092.2 & 0 & 12150 \\ 12682.8 & 16831.4 & 15310.3 & 12987.3 & 0 \end{pmatrix} \quad (1)$$



## Глава 4. Математическая модель

### 4.1 Матрицы корреспонденции

Матрицы корреспонденции в транспортных сетях (или матрицы перемещений) — это математическая модель для анализа и планирования транспортных систем, которая используется для описания потоков перемещений между различными зонами на транспортной карте. Эта модель позволяет понять, как и сколько людей или грузов перемещается между разными точками в определённый период времени. В данной работе используется матрица корреспонденции в качестве математической модели для анализа перемещений людей в транспортной сети.

### 4.2 Гравитационная модель

Гравитационная модель в контексте создания матриц корреспонденций является одним из ключевых инструментов прогнозирования транспортного спроса. Эта модель основана на аналогии с законом всемирного тяготения Ньютона и используется для оценки потоков перемещений между разными зонами или регионами. Основная идея гравитационной модели заключается в том, что величина потока перемещений между двумя точками прямо пропорциональна массе (или активности) этих точек и обратно пропорциональна расстоянию между ними [14]. Формула 2 является обобщенной формулой матрицы корреспонденции гравитационной модели [15].

$$p_{ij}^* = k \frac{s_i d_j}{f(c_{ij})}, \quad (2)$$

где:

- $p_{ij}^*$  — потенциальные корреспонденции между районами  $i$  и  $j$ ;
- $k$  — калибровочный коэффициент;
- $s_i$  — общий объем пользователей, выезжающих из источника  $i$ ;
- $d_j$  — общий объем пользователей, въезжающих в сток  $j$ ;

- $f(c_{ij})$  — функция тяготения, зависящая от удельных расходов  $c_{ij}$  на передвижение из источника  $i$  в сток  $j$ .

Для корректности гравитационной модели должны быть соблюдены условия баланса общего прибытия и отправления. Они представлены равенствами 3.

$$\begin{aligned}\sum_{j=1}^n p_{ij} &= s_i, \\ \sum_{i=1}^m p_{ij} &= d_j,\end{aligned}\tag{3}$$

$$p_{ij} \geq 0$$

С учетом этих условий выражение 2 может быть переписано в виде 4:

$$p_{ij}^* = \alpha_i \beta_j s_i d_j f(c_{ij}),\tag{4}$$

где  $\alpha_i, \beta_j$  - калибровочные коэффициенты, которые вычисляются из системы уравнений:

$$\begin{aligned}\alpha_i &= [\sum_j \beta_j d_j f(c_{ij})]^{-1} \\ \beta_j &= [\sum_i \alpha_i s_i f(c_{ij})]^{-1}.\end{aligned}\tag{5}$$

Система 5 является совместной при выполнении равенства исходящих и входящих потоков:

$$\sum_i s_i = \sum_j d_j$$

### 4.3 Энтропийная модель

Энтропийная модель матрицы корреспонденции в транспортном планировании представляет собой статистический подход, целью которого является распределение транспортного потока в сети таким образом, чтобы максимизировать энтропию системы, что в контексте транспортных потоков означает стремление к максимально возможному равномерному распределению потоков с учётом некоторых ограничений. Этот подход основан на

принципе, согласно которому в отсутствие дополнительной информации оптимальным предположением о распределении является самое неопределённое или наиболее равномерное распределение.

Энтропийная модель исходит из предположения, что при отсутствии информации о поведении конкретных водителей или пассажиров, все возможные способы реализации наблюдаемых агрегированных потоков между зонами равновероятны. Эта модель стремится найти наиболее вероятное распределение потоков перемещений, которое согласуется с известными агрегированными данными (например, общим количеством поездок, происходящих из каждой зоны и в каждую зону) и минимизирует дополнительную информацию, необходимую для описания распределения.

Максимизация взвешенной энтропии позволяет достигать не только равновесного состояния, но и наиболее точно описывать реальную ситуацию, учитывая индивидуальные предпочтения пользователей. Индивидуальные предпочтения моделируются с помощью функции распределения вероятностей  $a_{ij}$ , которая отражает вероятность того, что пользователь выберет перемещение из зоны  $i$  в зону  $j$ . Ограничения для этой модели совпадают с ограничениями, используемыми в гравитационной модели. Следовательно, энтропийная модель формулируется следующими уравнениями 6, 7, 8 [16]:

$$\max_{p_{ij}} \sum_{i=1}^n \sum_{j=1}^m p_{ij} \ln \frac{\alpha_{ij}}{p_{ij}}, \quad (6)$$

$$\sum_{j=1}^m p_{ij} = s_i, \quad (7)$$

$$\sum_{i=1}^n p_{ij} = d_j, \quad (8)$$

$$p_{ij} \geq 0, \forall i \in 1 \dots n, \forall j \in 1 \dots m.$$

Функция распределения вероятности  $\alpha_{ij}$  имеет вид:

$$\alpha_{ij} = e^{-\gamma c_{ij}}, \forall i \in 1 \dots n, \forall j \in 1 \dots m,$$

где  $\gamma$  — параметр расселения;  $c_{ij}$  — средние затраты на передвижение.

При разработке энтропийной модели используется априорная информация о предпочтениях пользователей к определенным корреспонденциям. Матрица априорных предпочтений  $\alpha_{ij}$  не только служит для распределения транспортных потоков, но и отображает распределение рабочих мест (в случае анализа трудовых перемещений) и жителей между районами. Это распределение учитывает предпочтения жителей относительно мест жительства и использования транспортной системы. К сожалению, в открытых источниках информацию о предпочтениях перемещения людей не получить, поэтому энтропийная модель не была реализована в данном исследовании.

#### 4.4 Итерационный алгоритм построения матрицы корреспонденции гравитационным методом

На предыдущих этапах работы были собраны и проанализированы данные о перекрёстках и населении зданий, а также посчитаны расстояния между перекрёстками. С помощью этой информации можно построить гравитационную модель матрицы корреспонденций. Рассмотрим итерационный алгоритм расчёта элементов матрицы [16].

- Определение начальных данных.

Общее количество отправок из перекрёстков  $Q_i$  — количество поездов, исходящих из каждого перекрёстка  $i$ .

Общее количество прибытий в перекрёсток  $D_j$  — количество поездов, прибывающих в каждый перекрёсток  $j$ .

$c_{ij}$  — расстояние между перекрёстками  $i$  и  $j$ .

- Инициализация начальных значений: Назначение начальных значений матрице корреспонденции  $T_{ij}^{(0)}$ .
- Определение функции притяжения (гравитационная модель). Используется базовая формула гравитационной модели 9:

$$T_{ij} = A_i B_j Q_i D_j f(c_{ij}) \quad (9)$$

Среди которых  $f(c_{ij})$  — функция сопротивления. В нашем случае используется экспоненциальная функция 10:

$$f(d) = e^{-\beta d} \quad (10)$$

Параметр  $\beta$  был выбран равным 0.065.

- Вычисление промежуточных коэффициентов  $A_i$  для каждой зоны  $i$  (11):

$$A_i^{(k)} = [\sum_j B_j^{(k-1)} D_j f(c_{ij})]^{-1} \quad (11)$$

- Обновление коэффициентов  $B_j$  для каждой зоны  $(j)$  (12):

$$B_j^{(k)} = [\sum_i A_i^{(k)} O_i f(c_{ij})]^{-1} \quad (12)$$

- Обновление значений матрицы корреспонденции  $T_{ij}$  (13):

$$T_{ij}^{(k)} = A_i^{(k)} B_j^{(k)} O_i D_j f(c_{ij}) \quad (13)$$

- Условие завершения. Для завершения итерационного алгоритма используются критерии сходимости. Сравниваются относительные изменения в матрице поездок  $T_{ij}$  между итерациями 14:

$$\left| \frac{T_{ij}^{(k)} - T_{ij}^{(k-1)}}{T_{ij}^{(k-1)}} \right| < \epsilon \quad (14)$$

Используя приведенную выше итерационную схему и собранные на предыдущих этапах данные, была построена матрица корреспонденции для города Петрозаводска.

## Глава 5. Результаты

В результате построения гравитационной модели с использованием итерационного алгоритма получилась матрица корреспонденции размером  $983 \times 983$  элементов. Далее проводится анализ и визуализация результатов.

### 5.1 Статистические показатели матрицы корреспонденции



Ниже представлены статистические показатели матрицы корреспонденции:

- Наименьшее ненулевое значение —  $5 \times 10^{-324}$ ;
- Наибольшее значение — 2559;
- Среднее арифметическое значений — 0.9755210141270656;
- Медиана ненулевых значений —  $8.575466425406286 \times 10^{-138}$ ;
- 95-персентиль ненулевых значений —  $6.293518548573857 \times 10^{-26}$ ;
- 99.9-персентиль ненулевых значений — 312.8838855850142;
- Сумма всех значений — 197662.01892445429

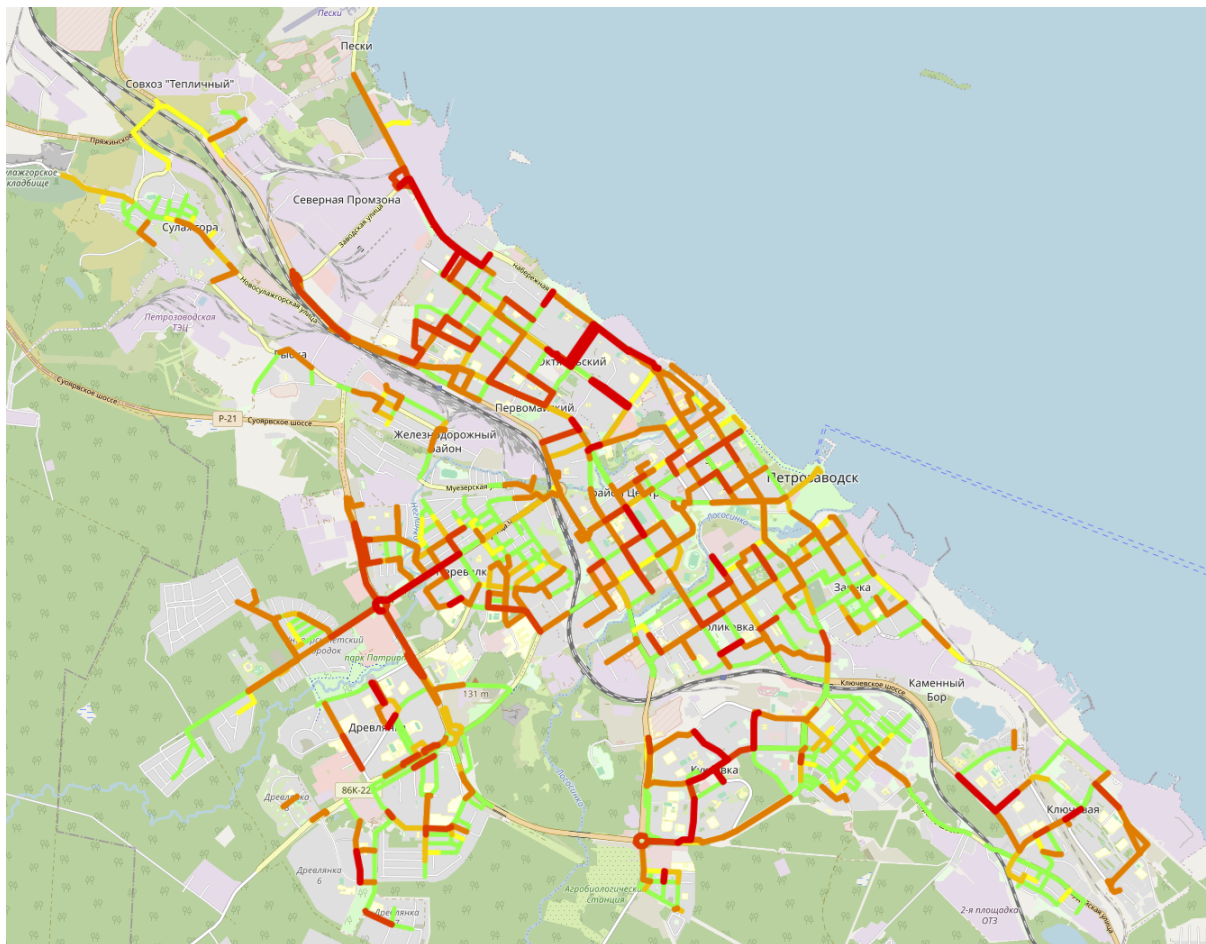
### 5.2 Визуализация матрицы корреспонденции

Визуализация матрицы корреспонденции была построена по следующему принципу: значения разбиты на интервалы, и каждому интервалу присвоен свой цвет, которым покрашена линия, соединяющая перекрёстки. Рассмотрим интервалы и цвета.

- $(0.000001; 20]$  —  (светло-зеленый)
- $(20; 50]$  —  (светло-желтый)
- $(50; 100]$  —  (светло-оранжевый)
- $(100; 500]$  —  (оранжевый)

- $(500; 1000]$  —  (красный)
- $(1000; \infty)$  —  (тёмно-красный)

Предложенная раскраска представлена на рисунке 5



**Рис. 5:** Раскраска перекрёстков с использованием значений матрицы корреспонденции

Таким образом модель даёт представление о загруженности города Петрозаводска. Изменяя параметры перекрёстков, можно смотреть за динамикой изменения загруженности транспортной системы.

## Выводы

Полученная в ходе работы модель отражает показатель загруженности транспортной сети города. Точечное изменение количества жителей в домах определенной области может более точно отразить показатели загруженности и проанализировать транспортную систему выбранной области. Важно учитывать, что такие изменения могут оказывать значительное влияние на поток транспорта, что, в свою очередь, влияет на эффективность и удобство перемещения по городу.

Также можно добавить новые дороги на карту, и пересчет модели даст новый результат в соответствии с измененными показателями. Таким образом, можно оценить, насколько новые дороги улучшат или ухудшат текущую транспортную ситуацию, а также определить, какие дополнительные меры могут потребоваться для оптимизации движения. Включение новых дорог помогает моделировать сценарии изменения сети, анализировать возможные заторы и планировать меры по их предотвращению.

Таким образом, предложенную схему получения данных для построения модели можно использовать при планировании улучшений существующих дорог или при моделировании новых районов города с целью анализа влияния новых транспортных узлов на существующую систему. Применение этой модели позволяет городским планировщикам и инженерам принимать обоснованные решения, повышающие эффективность транспортной инфраструктуры. Это также способствует улучшению качества жизни горожан, снижению времени в пути и уменьшению вредных выбросов благодаря более рациональному распределению транспортных потоков.



## Заключение

В ходе работы были собраны данные о населении и транспортной системе города Петрозаводска. Данные включали информацию о местоположении зданий, их площади, количестве этажей и типе здания, а также информацию о местоположении перекрёстков, и другие показатели, влияющие на работу транспортной сети. Далее эти данные были проанализированы и обработаны для получения удобных значений для расчёта математической модели. Также были рассмотрены несколько подходов к построению матрицы корреспонденции: гравитационный и энтропийный. В результате был выбран гравитационный подход, потому что для энтропийного невозможно получить априорные вероятности, необходимые для расчёта модели.

Следующим этапом был расчёт матрицы корреспонденции и визуализация результатов на карте транспортной системы города Петрозаводска. Этот процесс включал в себя использование картографических данных для определения основных узлов и соединений в транспортной сети, что позволило создать более точную и информативную модель. Визуализация данных на карте выявила основные направления транспортных потоков, очаги перегруженности и возможные зоны улучшения.

Полученные результаты позволяют увидеть взаимосвязи и взаимодействия между узлами транспортной сети города, которые в нашем случае были представлены перекрёстками. Анализ этих взаимодействий помогает понять, каким образом можно оптимизировать перемещение между различными частями города, улучшить логистику и сократить время в пути для жителей.

Данная работа может помочь в моделировании будущих изменений транспортной системы или в улучшении существующих проблем с пропускной способностью дорог. Предложенные методы и модели могут быть использованы городскими планировщиками и инженерами для разработки стратегий по уменьшению заторов и повышению эффективности городской транспортной системы. Кроме того, результаты анализа могут быть полезны для принятия обоснованных решений о необходимости строительства новых дорог или реконструкции существующих, а также для внедрения новых технологий, таких как интеллектуальные транспортные системы.

Таким образом, представленная работа не только способствует текущему пониманию состояния транспортной системы Петрозаводска, но и предоставляет инструменты для ее дальнейшего улучшения и развития, что в конечном итоге поможет повысить качество жизни горожан и сделать городские путешествия более комфортными и эффективными.

## Список литературы

- [1] Д. Е. Намиот, М. Н. Некраплённая, О. Н. Покусаев, А. Е. Чекмарев «Матрицы корреспонденций и анализ пассажирских потоков». International Journal of Open Information Technologies ISSN: 2307-8162 vol. 8, no.4 (8), 2020.
- [2] [URL] Open Street Map <https://www.openstreetmap.org/about>
- [3] [URL] bbbike <https://www.bbbike.org/>
- [4] «Жилищный кодекс Российской Федерации» от 29.12.2004 N 188-ФЗ, Статья 50
- [5] Закон Республики Карелия от 25 февраля 1997 г. N 176-ЗРК «О государственном и муниципальном жилищных фондах и их использовании», Статья 35
- [6] «Методические рекомендации по обеспечению санитарно-эпидемиологических требований к условиям труда», МР 2.2.0244-21, 2021
- [7] СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
- [8] СанПиН 2.4.1.3049-13 «Санитарно-эпидемиологические требования к устройству, содержанию и организации режима работы дошкольных образовательных организаций»
- [9] СанПиН 2.1.3.2630-10 «Санитарно-эпидемиологические требования к организациям, осуществляющим медицинскую деятельность»
- [10] [URL] Формат алгоритма кодированной ломаной линии в Google <https://developers.google.com/maps/documentation/utilities/polylinealgorithm>
- [11] [URL] Google Directions API <https://developers.google.com/maps/documentation/directions/overview>

- [12] [URL] Project OSRM <https://project-osrm.org/>
- [13] [URL] Docker <https://www.docker.com/>
- [14] I Ekowicaksono, F Bukhari, and A Aman «Estimating Origin-Destination Matrix of Bogor City Using Gravity Model». IOP Conf. Series: Earth and Environmental Science 31 (2016) 012021
- [15] Я. А. Селиверстов, С. А. Селиверстов «Методы и модели построения матриц транспортных корреспонденций». Научно-технические ведомости СПбГПУ 2' (217) – 3' (222) Информатика. Телекоммуникации. Управление
- [16] Б. Я. Советов, А. В. Сикерин Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина) «Гравитационная и энтропийная модели потоков при территориальном планировании развития транспортной системы». Известия СПбГЭТУ «ЛЭТИ» № 8/2016