Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра информационно-аналитических систем

Филатова Анастасия Алексеевна

Разработка методик и построение алгоритмов моделирования пассажиропотока на коммерческом транспорте Санкт-Петербурга с целью минимизации временных и трудовых затрат

Бакалаврская работа

Научный руководитель:
к.ф.-м.н., доцент Графеева Н.Г.

Рецензент:
Зам. начальника отдела стратегического планирования
СПБ ГКУ «Организатор перевозок» Махмутов В.Р.

Санкт-Петербург
2019

SAINT PETERSBURG STATE UNIVERSITY
Mathematics and Mechanics Faculty
Analytical Informational Systems Department

Filatova Anastasiia

Development of methodologies and construction of algorithms for modeling passenger flow on commercial transport in St. Petersburg in order to minimize time and labor costs

Bachelor's graduation paper

Supervisor:
Ph.D., associate professor
Grafeeva N.G.

Reviewer:
Deputy Head of Department of Strategic Planning
Makhmutov V.R.

Saint-Petersburg
2019

Оглавление

Введение 4
Постановка задачи 7
Обзор литературы 8
Описание и обработка данных 12
Транзакционные данные об оплате проезда смарт-картами 12
Данные наблюдателей 15
Данные об остановках на городских и коммерческих маршрутах 16
Методология 19
Оценка пассажиропотока по данным о похожих городских маршрутах 19
Оценка пассажиропотока по транзакционным данным 22
Оценка пассажиропотока по данным наблюдателей 23
Реализация методик 25
1. Оценка пассажиропотока по данным о похожих городских маршрутах 25
Используемые данные 25
Обработка данных 25
Описание алгоритма поиска похожих маршрутов 26
Применение методики к оценке пассажиропотока 29
2. Оценка пассажиропотока по транзакционным данным 31
Используемые данные 31
Обработка данных 31
Описание алгоритма восстановления мест посадки пассажиров 32
Применение методики к оценке пассажиропотока 37
Анализ применимости описанных методик 38
Заключение 39
Список литературы 40
Приложение 1. Функции, реализующие основные алгоритмы 41

Введение

В современном мире трудно переоценить значимость общественного транспорта в жизни людей. Человек чаще всего уже не может довольствоваться отдельным кварталом, а зачастую даже районом города. Разнесение дома, места работы, магазинов, общественных мест и мест отдыха в пространстве города порождают необходимость в большом количестве перемещений. С учетом того, что далеко не все могут себе позволить личный транспорт или не видят в нем необходимости, основная нагрузка по обеспечению перевозок пассажиров ложится на системы общественного транспорта.

Санкт-Петербург является крупнейшим транспортным узлом в Северо-Западном Федеральном округе и вторым в стране после Москвы. Поэтому для того, чтобы улучшить качество жизни жителей города, крайне важно, чтобы инфраструктура общественного транспорта развивалась и улучшалась, чтобы городская транспортная система была максимально удобной и эффективной.

Особенностью России и Санкт-Петербурга в частности, является широко развитая сеть коммерческого маршрутного транспорта. Маршрутные такси имеют ряд преимуществ перед муниципальным транспортом: они более мобильны, количество машин на маршруте у них больше, чем у многих других видов транспорта. Зачастую маршрутные такси позволяют существенно сократить время, затрачиваемое на дорогу, а стоимость проезда в них сравнима со стоимостью проезда в муниципальном транспорте для тех категорий граждан, которые не имеют социальных льгот. На сегодняшний день все больше людей предпочитают маршрутные такси иному наземному транспорту как раз потому, что они быстры и мобильны, что очень важно как для работающих граждан, так и для отдыхающих и туристов, желающих меньше времени проводить в дороге. В связи с этим возникает необходимость анализа существующей системы коммерческих маршрутов для улучшения ее качества, а в частности, оценка пассажиропотока на коммерческих маршрутах и построение алгоритмов для его моделирования, чему и посвящена моя работа.

Пассажиропотоком в данном исследовании будем называть количество пассажиров, которые совершают поездку в данный момент времени на транспортном средстве, следующем по определенному маршруту, между соседними остановками. Умение качественно оценивать пассажиропоток на коммерческих маршрутах очень важно для повышения эффективности существующей системы. Пассажиропоток может частично отражать коллективную мобильность пассажиров, следующих по определенному маршруту, и качество перевозок с точки зрения комфорта. С точки зрения планирования это говорит о том, сколько людей путешествует или хочет путешествовать по данному маршруту. Эта информация может помочь операторам динамически распределять и планировать маршруты коммерческого транспорта и его расписание с высокой степенью детализации и точности.

В Санкт-Петербурге для городских маршрутов был разработан ряд систем, позволяющих упростить как пользование транспортом, так и анализ существующей транспортной системы. В первую очередь это введение на маршрутах бесконтактной оплаты проезда с использованием бесконтактных смарт-карт (БСК). На данный момент существует большое количество разных видов БСК: электронная карта-кошелек «Подорожник», пенсионные, ученические, студенческие карты и прочие. Информация о транзакциях оплаты проезда смарт-картами на маршрутах учитывается системой электронного контроля оплаты проезда (СЭКОП) и может быть использована для дальнейшего анализа. Также для удобства отслеживания транспортных средств на маршрутах была введена автоматическая система управления городским пассажирским транспортом (АСУГПТ), которая позволяет определять местоположение конкретного транспортного средства в определенный момент времени. Это происходит благодаря специальным датчикам, которые установлены на борту.

Важное отличие между коммерческими и муниципальными маршрутами в Санкт-Петербурге заключается в том, что на коммерческих маршрутах на данный момент не представляется возможным установить датчики АСУГПТ, поэтому

становится невозможным отслеживание коммерческих транспортных средств на маршрутах. Кроме того, на коммерческих маршрутах допустима оплата смарт-картами только в виде электронного кошелька «Подорожник». Поэтому в СЭКОП данные о транзакциях оплаты проезда БСК представлены только в виде транзакций смарт-карт «Подорожник».

В данной работе я исследую и разрабатываю методики, которые позволят оценить пассажиропоток на коммерческом транспорте Санкт-Петербурга, основываясь на данных СЭКОП об оплате пассажирами проезда на маршрутах коммерческого транспорта с использованием бесконтактных смарт-карт «Подорожник» и, при необходимости, минимальных дополнительных наблюдений и обследований.

Постановка задачи

Цель работы

Целью данной работы является исследование и разработка методик оценки пассажиропотока на коммерческих маршрутах Санкт-Петербурга с использованием данных о транзакциях смарт-карт «Подорожник» и, при необходимости, дополнительных данных.

Задачи

- Проанализировать и обработать имеющиеся данные, подготовить их к дальнейшей работе.
- Исследовать необходимость использования дополнительных источников данных для более качественного анализа. Заявить о необходимости дополнительных данных и обследований.
- Разработать методики оценки пассажиропотока на коммерческих маршрутах на основе представленных данных.
- Оценить пассажиропоток на коммерческих маршрутах с помощью разработанных методик.
- Провести анализ результатов.
- Визуализировать результаты.

Обзор литературы

Для изучения существующих алгоритмов, позволяющих решать задачи, схожие с задачами, поставленными в рамках данной работы, был проведен обзор литературы. Возможность использования данных смарт-карт для качественного анализа характеристик транспортных систем, в том числе для анализа пассажиропотока, была подтверждена многочисленными исследованиями на эту тему.

В работах «The potential of public transport smart card data» [1] и «The promises of big data and small data for travel behavior (aka human mobility) analysis» [2] приводятся соображения и доказательства того, что качественно подготовленные данные о транзакциях смарт-карт могут являться хорошим самостоятельным источником данных в задачах анализа транспортных систем.

Работа «Smart Card Data Mining of Public Transport Destination: A Literature Review» [3] представляет собой обзорную статью, посвященную разным подходам в задаче оценивания места выхода пассажира, оплатившего проезд смарт-картой. Модели, рассматриваемые в данной статье, в основном нацелены на использование данных систем контроля оплаты, в которых время и место выхода пассажира в системе не фиксируется. К таким системам как раз и относится СЭКОП, поэтому данные подходы заслуживают пристального внимания.

В работе [3] рассмотрены три модели, которые наиболее широко используются в задаче оценивания места выхода пассажира. Модель, которая наиболее применима в моей работе – это **модель корреспонденций**. Данный подход применяется в основном исследователями из Китая и Америки в анализе транспортных систем городов, где нет зонального деления маршрутов и, как следствие, большинство систем контроля оплаты проезда не фиксируют данные о месте и времени выхода пассажира из транспортного средства.

Одной из ключевых работ по анализу пунктов назначения пассажиров, в основе которой лежит модель корреспонденций, является работа «Origin and

Destination Estimation in New York City with Automated Fare System Data» [4]. В ней авторы используют следующие предположения: станция высадки текущей поездки на определенном виде транспорта является начальной станцией следующей поездки, и большинство пассажиров заканчивают свою последнюю поездку в день на станции, где они начинают свою первую поездку в этот же день.

В еще одной работе, использующей модель корреспонденций – «Spatio-temporal Analysis of Passenger Travel Patterns in Massive Smart Card Data» [5], авторы добавили еще два предположения. Первое – о том, что люди не совершают продолжительных переходов пешком, когда пересаживаются с одного транспорта на другой и что между пользованием транспортом они не используют дополнительно личные машины/ велосипеды/ мотоциклы и прочее.

Еще одна модель, которая используется для задачи оценивания места и времени выхода пассажиров – это **вероятностная модель.** В работе «OD Matrix Estimation Method of Public Transportation Flow Based on Passenger Boarding and Alighting» [6], в основе которой лежит вероятностная модель, авторы используют матрицы отправления – назначения. В данной работе для каждой станции рассчитывается вероятность, с которой для конкретного пассажира (заданного идентификатором смарт-карты) данная остановка будет конечной с учетом того, сколько он уже едет. Также авторами бы сделан интересный вывод, что расстояние, на которое пассажиры путешествуют, подчиняется распределению Пуассона. Стоит отметить, что так же, как и в модели корреспонденций, для использования данной модели необходимо сначала восстановить данные о местах посадки пассажиров. Еще одна особенность заключается в том, что при большей сложности реализации данный подход дает сравнимую точность с методом корреспонденций, поэтому его использование применительно к моей задаче не совсем оправдано.

Третий описанный подход к решению данной задачи – это использование **глубокого обучения и нейронных сетей**. В статье «Estimation a Transit Route OD Matrix Using On/off Data: An Application of Modified BP Artificial Neural Network»

[7], авторы использовали нейронную сеть для того, чтобы предсказывать остановки, на которых пассажиры будут выходить, по входным данным о том, где и когда пассажиры вошли. В работе «Deep-learning Architecture to Forecast Destinations of Bus Passengers from Entry-only Smart-card Data» [8], кроме информации о транзакциях смарт-карт в качестве входных данных, использовались также характеристики местности и особенности конкретных маршрутов и транспортных сетей. Данный подход может давать точные результаты, но только в том случае, если корректных данных о фактическом пассажиропотоке на коммерческом транспорте достаточно много для правильного обучения нейронной сети. В условиях моей работы такие данные не могут быть предоставлены в достаточном количестве.

В результате обзора ряда работ, исследующих разные подходы к решению задачи определения места выхода пассажиров из транспортного средства, мной был сделан вывод, что модель корреспонденций подходит для решения моей задачи больше всего. Она относительно проста в реализации, но дает достаточную точность и ключевые факторы, используемые в ней для предсказания места выхода пассажира – это место и время его посадки, а также информация о допустимом расстоянии, которое пассажир может пройти пешком во время пересадки. Нюанс использования данной модели применительно к имеющимся данным о транзакциях смарт-карт на коммерческих маршрутах – то, что в данных нет информации о местах посадки, и ее придется восстанавливать.

Восстановлению данных о месте посадки пассажиров посвящена работа «Extraction bus transit boarding stop information using smart card transaction data» [9]. В ней описываются методы восстановления информации о месте посадки пассажира, оплатившего проезд смарт-картой, с использованием информации о транзакциях на транспортном средстве и базовой информации о маршруте. В связи с указанной выше необходимостью получения информации о местах посадки пассажиров, методы, описанные в данной статье, могут быть полезны в моей работе.

Наконец, работу «Using Smart Card Technologies to Measure Public Transport Performance: Data Capture and Analysis» [10] можно использовать как пример качественной обработки данных для дальнейшего анализа.

В результате обзора литературы мной также был сделан вывод, что для проведения качественного анализа пассажиропотока на основе транзакционных данных, необходимо получить также информацию об остановках, на которых пассажиры заходили в маршрутное транспортное средство и добавить эту информацию к уже имеющимся данным о времени совершаемых транзакций оплаты проезда смарт-картами.

Описание и обработка данных

Первоначально данные для анализа были предоставлены СПб ГКУ «Организатор перевозок» в виде нескольких csv-файлов, в которых находилась информация из СЭКОП о транзакциях оплаты проезда на коммерческих маршрутах с использованием смарт-карты «Подорожник.

Транзакционные данные об оплате проезда смарт-картами

Данные об оплате проезда на коммерческих маршрутах с использованием смарт-карты «Подорожник» были собраны в период с 1 августа 2018 года по 24 октября 2018 года. Они были представлены в виде 6 файлов формата csv, которые содержали следующую информацию: номер смарт-карты, закодированный тип карты, дата и время транзакции с точностью до секунд, код перевозчика, идентификатор маршрута в СЭКОП, бортовой номер транспортного средства, идентификатор считывающего устройства, номер поездки, идентификатор кондуктора и дату транзакции. Для удобства дальнейшей работы данные о транзакциях были избавлены от некорректных записей и объединены в общий файл формата csv. Данный файл стал содержать 1273428 записей о 64-ех коммерческих маршрутах. Из него также были убраны такие данные, как закодированный тип карты, код перевозчика, идентификатор считывающего устройства и идентификатор кондуктора, т.к. тип карты во всех транзакциях единый, а остальные параметры однозначно задавались бортовым номером транспортного средства. В итоговый файл к идентификаторам маршрутов в СЭКОП также были добавлены номера маршрутов. Вид итогового файла показан на **рис. 1**.

CARD_NUM	TRANSACT_DAY	TRANSACT_MONTH	TRANSACT_YEAR	ID_ROUTE	CARRIER_BOARD_NUM	TRANSACT_TIME	ROUTE_NUM
36125888012291588	1	8	2018	413	123	09:00:09	191
36127155834736132	1	8	2018	413	123	09:36:11	191
36124337533755652	1	8	2018	413	123	09:43:19	191

Рис. 1. Вид итогового файла, содержащего все транзакции в период август – октябрь 2018 года

Для разработки методик, позволяющих качественно оценить пассажиропоток на коммерческих маршрутах, был проведен разведочный анализ данных с целью выявления интересных закономерностей в транзакционных данных. Ниже представлены диаграммы, описывающие найденные закономерности.

В транзакционных данных присутствует достаточно много маршрутов, на которых оплата проезда смарт-картами является очень популярной. На двенадцати из них максимальное количество транзакций оплаты проезда смарт-картами за сутки составляет более 500. Эти маршруты представлены на **рис.2**.

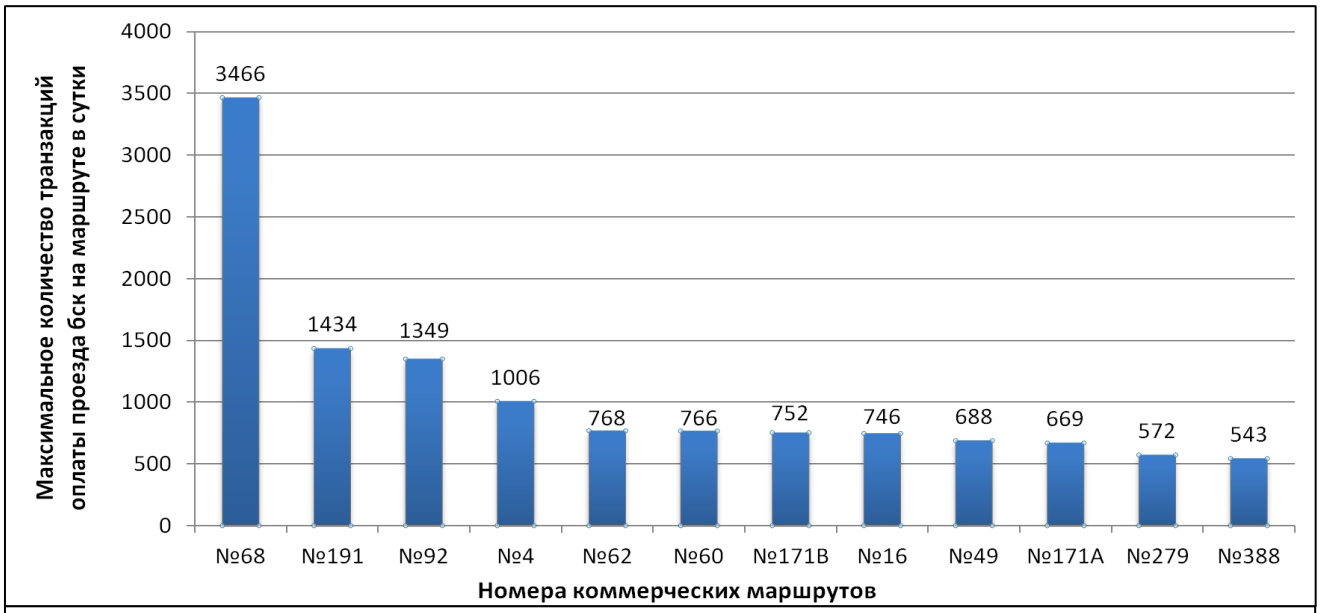

Рис. 2. Двенадцать наиболее популярных коммерческих маршрутов среди пассажиров, оплачивающих проезд смарт-картами «Подорожник»

В будние и выходные дни транзакции распределяются по-разному. Как можно увидеть на **рис. 3** и **рис. 4**, в будние дни характерно увеличение количества транзакций в утренние и вечерние часы, в то время как в выходные большинство транзакций производится днем.

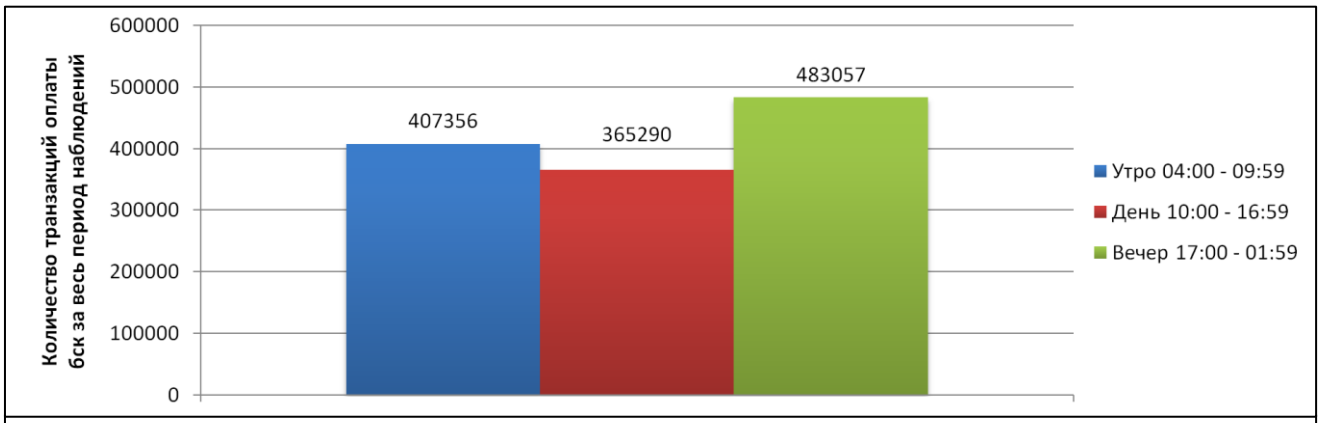

Рис. 3. Распределение общего количества транзакций оплаты проезда смарт-картами в будние дни

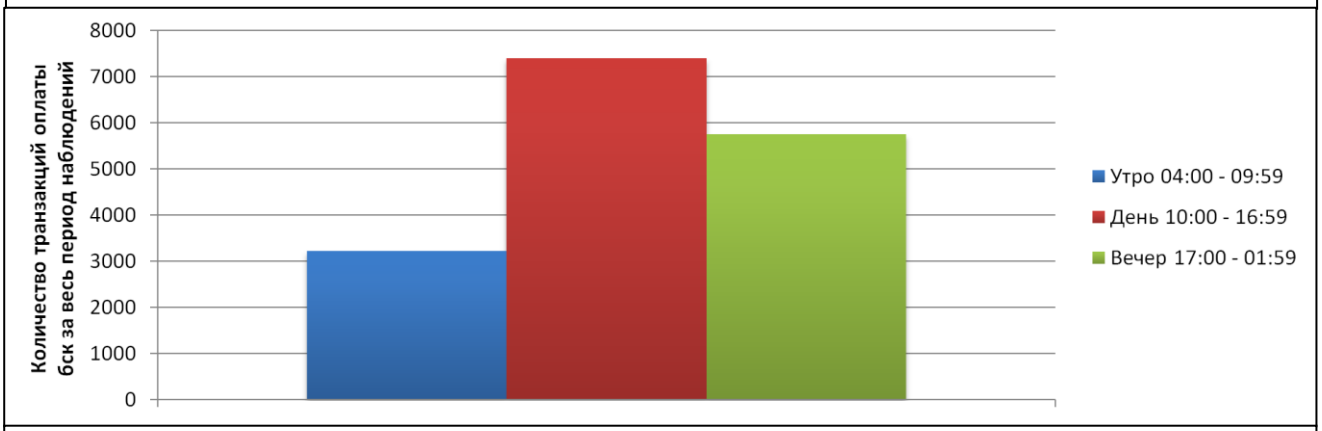

Рис. 4. Распределение общего количества транзакций оплаты проезда смарт-картами в выходные

Для получения более детальной информации относительно распределения транзакций были построены графики, показывающие распределение по часам среднего количества людей, которые оплачивали проезд смарт-картами, отдельно в будние и выходные дни. Результаты представлены на **рис. 5** и **рис. 6**. Из данных графиков видно, что в будние дни наибольшее количество транзакций приходится на часы, когда пассажиры едут на работу и с работы домой, а в выходные транзакции практически равномерно распределяются с 9 утра до 9 вечера.

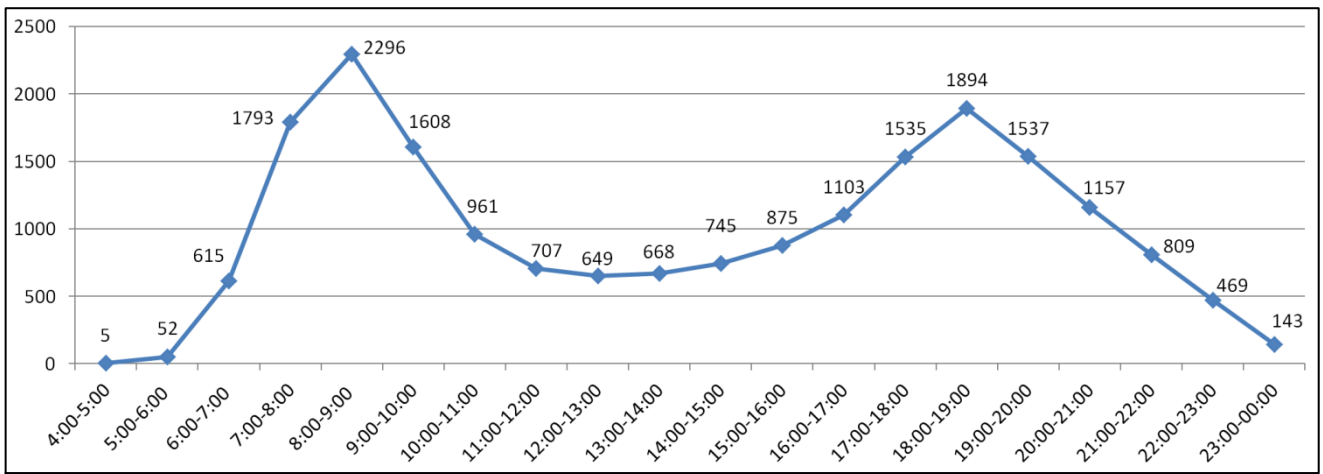

Рис. 5. Распределение по часам среднего количества транзакций в будние дни

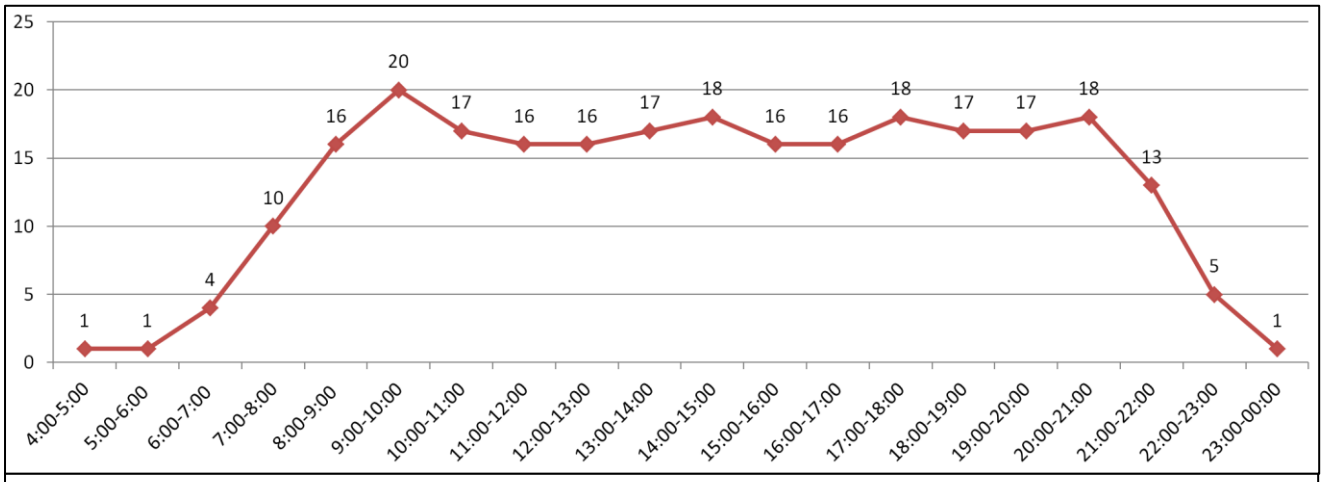

Рис. 6. Распределение по часам среднего количества транзакций в выходные

После проведения разведочного анализа данных и разработки методик оценки пассажиропотока были запрошены дополнительно следующие данные: информация об остановках на маршрутах, данные наблюдателей, корреспонденции и информация о пассажиропотоке на городском транспорте за указанный период, а также полный список остановок и кластеры остановок.

Данные наблюдателей

Данные наблюдателей были представлены в виде большого количества файлов формата xlsx, разделенных на файлы. В них содержалась информация разного уровня детализации о количестве пассажиров на транспортных средствах, следующих по указанным маршрутам, в определенные даты и время. Эта информация была собрана вручную пассажирами-наблюдателями. Наиболее полной и удобной для обработки и дальнейшего использования в моем исследовании оказалась информация от наблюдателей, которые находились внутри транспортного средства и записывали количество пассажиров, входящих и выходящих на определенных остановках. Остальная информация содержала существенные пробелы, которые невозможно было восстановить.

Для удобства использования отобранная информация от наблюдателей была приведена в единый вид, и для каждого описываемого транспортного средства по данным о количестве входящих и выходящих пассажиров было вычислено общее

количество пассажиров в салоне в каждый момент времени следования транспортного средства по маршруту. Пример обработанных данных наблюдателей можно увидеть на **рис. 7**.

Результаты обследования пассажиропотока
на маршруте К-92
(обратное направление)
Ул. Химиков - Ст.метро Ладожская

Дата обследования: **15.10.2018** **ТС №: У 715 ХО**
Время обследования: **08:15 - 08:56**

№ п/п	Остановка	Вошло	Вышло	Наполняемость
1	Ул.Химиков,26			0
2	Ш.Революции			0
3	Ул. Комунны/Ириновский пр.	2		2
4	Ул. Коммуны (уг. Ударников)	8		10
5	Пр. Наставников	14		24
6	Пр. Ударников,29	4	1	27
7	Индустриальный пр.	11	1	37
8	Пр. Ударников	6		43
9	Поликлиника 103		1	42
10	Ул.Передовиков		1	41
11	Ст.метро Ладожская		41	0
ИТОГО:		**45**	**45**	

Рис. 7. Пример обработанного отчета наблюдателя

Данные об остановках на городских и коммерческих маршрутах

Данные об остановках на маршрутах были представлены в виде набора записей о посещении транспортным средством, следующим по указанному маршруту в указанную дату, остановок с указанием их идентификаторов, направления движения транспортного средства, порядкового номера остановки на маршруте и расстояния до следующей остановки (**рис. 8**).

ID_ROUTE	ID_ROUTE_ASUGPT	ID_TRANSPORT	ROUTE_NUMBER	ROUTE_NAME	STOP_NUMBER	DIRECTION	ID_STOP	ID_NEXT_STOP	DISTANCE	DDATE
15262	240	1	153	АВТОБУСНАЯ СТАНЦИЯ "ПР. КУЛЬТУРЫ" - ХАСАНСКАЯ ...	16	1	3247	2352	0,6	03.10.2018
15262	240	1	153	АВТОБУСНАЯ СТАНЦИЯ "ПР. КУЛЬТУРЫ" - ХАСАНСКАЯ ...	17	1	2352	1892	0,39	03.10.2018
15262	240	1	153	АВТОБУСНАЯ СТАНЦИЯ "ПР. КУЛЬТУРЫ" - ХАСАНСКАЯ ...	18	1	1892	2232	0,51	03.10.2018

Рис. 8. Вид файла, содержащего информацию о посещении транспортными средствами остановок

Первичный анализ показал, что в данном документе корректно представлена информация только о маршрутах городского транспорта, а все записи, соответствующие коммерческим маршрутам, содержат неполную информацию об остановках. По результатам первичного анализа было принято решение обработать имеющиеся данные для получения списков остановок на городских маршрутах в прямом и обратном направлениях в период 1 августа – 24 октября 2018 года. Данные об остановках коммерческих маршрутов необходимо было собрать вручную.

В результате для всех городских маршрутов был подготовлен документ, содержащий номер маршрута и тип транспорта, а также количество остановок и списки идентификаторов остановок в двух направлениях (**рис. 9**).

route_number	id_transport	num1	stops_in_direction1	num2	stops_in_direction2
63	1	24	2751,3917,1322,23734,18338,18339,18344,23494,1...	24	14794,3028,1546,1439,3836,18405,18406,18407,18...
64	1	24	22854,19643,1840,3254,1701,20281,20284,19659,1...	29	20307,3552,2800,3624,2863,22739,2747,1693,3298...
72	1	31	22854,3895,14794,3028,1546,1439,1438,1523,2367...	31	3256,18050,22206,14813,3407,2552,3581,1612,283...
89	1	11	23035,20123,20125,20128,20130,20148,20149,2015...	13	20044,18180,18181,20047,20266,23454,20152,2015...
9	1	18	22735,22734,23100,3457,24980,24981,2567,4499,2...	19	1790,1865,3443,3404,2221,2009,4501,3474,2933,3...

Рис. 9. Итоговый вид файла с информацией об остановках на городских маршрутах

Среди коммерческих маршрутов были отобраны 9 наиболее интересных с точки зрения анализа. Они представляют интерес, т.к. по этим маршрутам были собраны данные наблюдателей и для них имелось большое количество информации о транзакциях оплаты проезда «Подорожником», что позволило в дальнейшем сравнить разные методики оценки пассажиропотока и выявить оптимальную. Для этих девяти маршрутов вручную с помощью справочников, общего списка остановок и электронных онлайн-карт была собрана информация об остановках в прямом и обратном направлениях и расстоянии между остановками. Итоговая информация также была помещена в отдельный файл (**рис. 10**).

route_number	num1	stops_in_direction1	num2	stops_in_direction2
191	38	20777,3391,3471,3711,3713,7070,4603,3717,2675,...	34	21667,3463,14949,1351,1531,2186,2223,1788,4376...
68	19	20044,2263,15310,3416,2350,15321,2197,20058,25...	19	20027,20028,20030,20035,20037,20040,1830,3540,...
1	14	24748,2909,3303,16431,1632,16501,1792,22153,45...	12	3201,2778,2127,3716,4517,22153,1792,22353,1632...
184	17	20044,2263,15310,3416,2350,15321,2197,20058,25...	17	28233,23019,14743,3489,1830,3540,2211,2293,254...
226	36	22739,22739,1542,2893,3298,16510,2583,2086,238...	35	27893,27888,27885,22084,22085,22086,22087,2222...
92	13	15952,1847,4116,3485,1556,3116,1921,3114,3749,...	11	22798,2475,3421,3114,1921,3116,1556,3485,4116,...
55	7	4502,1835,1591,4537,1360,2641,1548	7	1548,24981,1360,4499,2798,3022,4502
67	12	1361,3501,3577,2904,3302,2866,3163,2685,2684,3...	12	18397,25347,1473,1474,1971,1662,1707,4769,1706...
7	8	3201,2957,2652,1722,2858,2984,16362,18467	6	18467,16362,2984,2858,1722,2957,3201

Рис. 10. Итоговый вид файла с информацией об остановках на коммерческих маршрутах

Методология

После анализа и первичной обработки данных мной были разработаны несколько методик оценки пассажиропотока на коммерческом транспорте Санкт-Петербурга. Описанные методики учитывают нюансы, связанные с коммерческими маршрутами, такие как отсутствие информации о местоположении транспортных средств, следующих по маршруту.

Оценка пассажиропотока по данным о похожих городских маршрутах

В основе данной методики лежит идея поиска для коммерческих маршрутов похожих городских и вычисление пассажиропотока на основе соответствующих данных о городском транспорте.

Городские маршруты Санкт-Петербурга и их основные характеристики уже достаточно подробно изучены, в том числе имеется возможность рассчитать пассажиропоток на городских маршрутах с использованием метода корреспонденций и данных АСУГПТ о местоположении транспортных средств на маршрутах. Методики вычисления пассажиропотока на городском транспорте Санкт-Петербурга описаны в работе [11]. Поэтому возникает идея использовать уже имеющиеся данные о пассажиропотоке на городском транспорте для расчета пассажиропотока на коммерческих маршрутах.

Суть методики заключается в следующем. Зная вместимость транспортных средств, следующих по городским маршрутам, и пассажиропоток на них, а также вместимость коммерческих транспортных средств, следующих по похожим маршрутам, можно оценить пассажиропоток на коммерческих маршрутах.

Метрика «похожести» маршрутов

Введем определение похожих маршрутов. Для удобства назовем промежуток между двумя остановками на маршруте перегоном. Тогда маршруты будем считать тем более похожими, чем больше у них общих перегонов и чем длиннее цепочки таких перегонов.

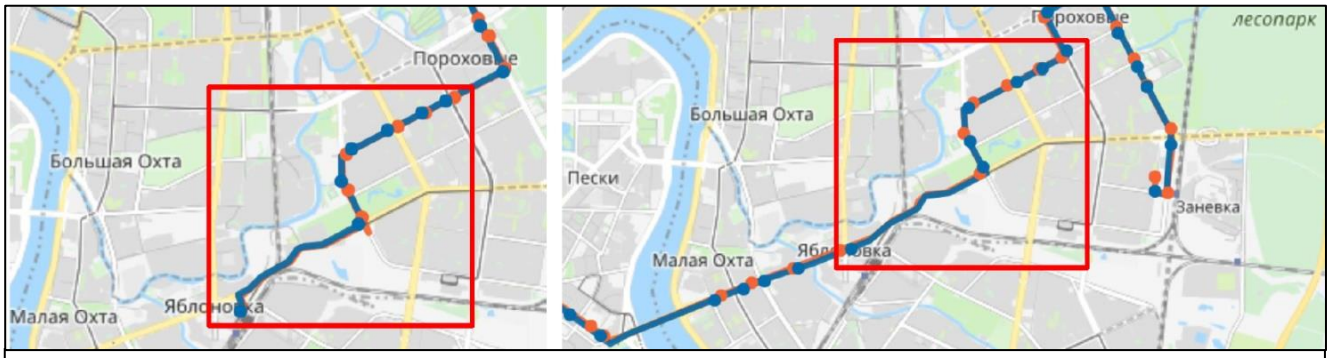

Рис. 11. Пример похожих маршрутов. Слева коммерческий маршрут №92, справа – часть городского автобусного маршрута №27. Цепочка общих перегонов выделена красным.

Но особенностью транспортной системы Санкт-Петербурга является то, что у разных видов городского транспорта зачастую разные остановки, которые, тем не менее, находятся рядом друг с другом. Такие остановки можно объединить в кластеры (например, одному кластеру будут принадлежать все остановки, расположенные около выбранной станции метро). Поэтому корректно будет определить общие перегоны на маршрутах как такие перегоны, у которых начальные остановки принадлежат одному кластеру и конечные остановки принадлежат одному кластеру (**рис. 12**).

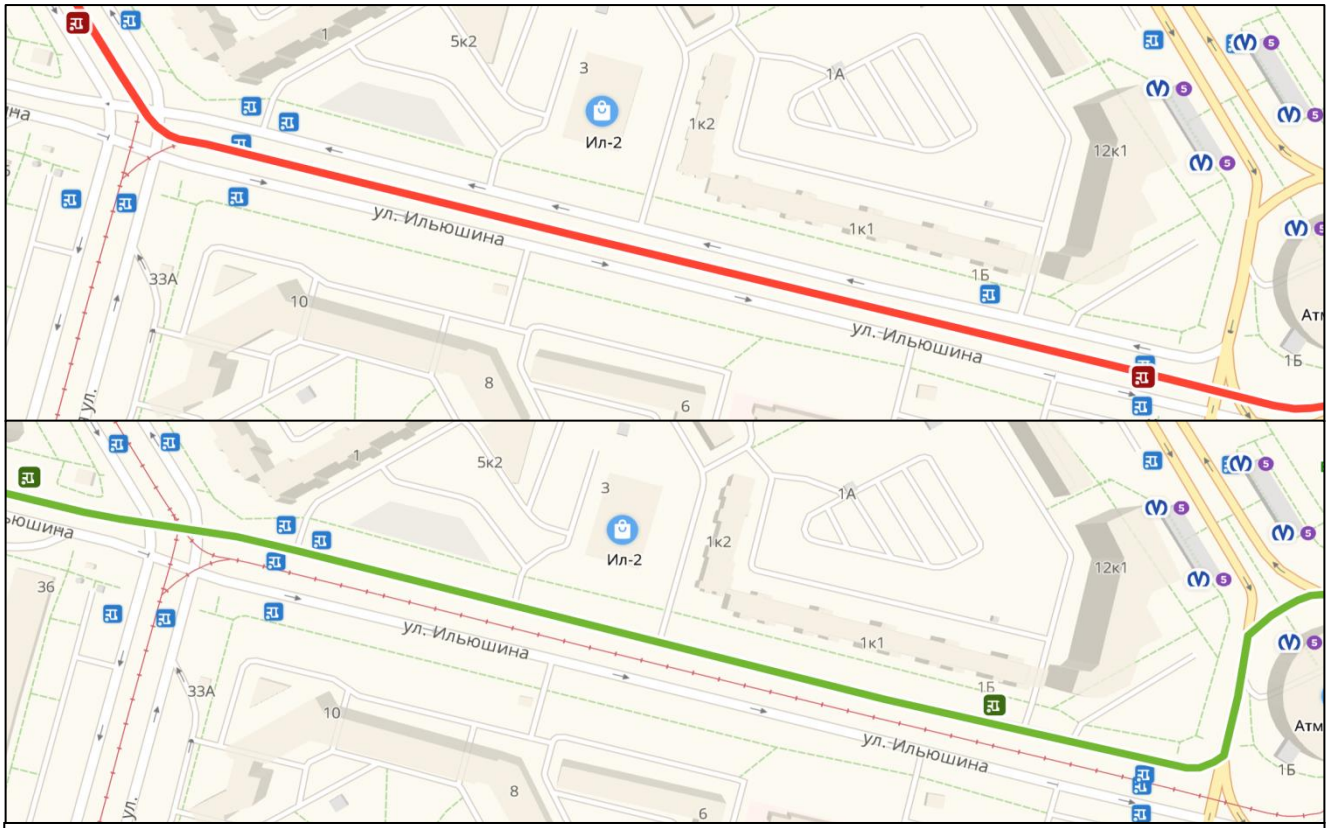

Рис. 12. Пример перегонов, которые должны быть расценены как общие. Сверху – часть трамвайного маршрута №55. Снизу – часть автобусного маршрута № 126.

Обоснование применимости данной методики

Применимость данной идеи подтверждается следующими наблюдениями. Распределение по часам среднего количества транзакций оплаты на коммерческих маршрутах в будние и выходные дни (**рис. 5, 6**) совпадает с соответствующими распределениями на городском транспорте среди пассажиров, не имеющих социальных льгот (**рис. 13, 14**), которые представлены в работе [11].

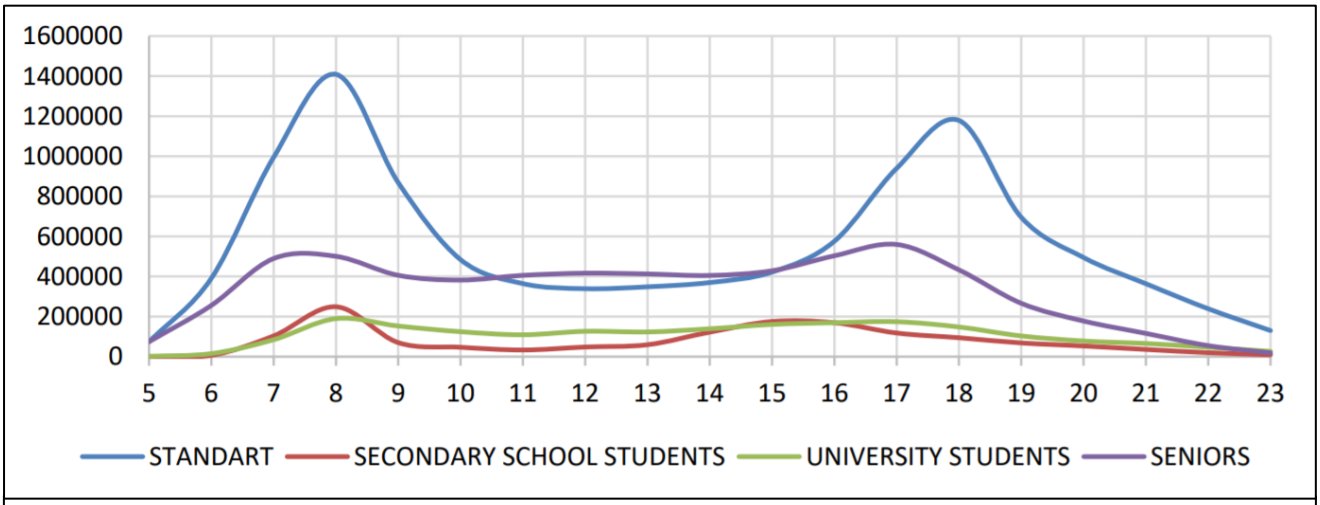

Рис. 13. Распределение пассажиропотока на городском транспорте по часам в будние дни для разных категорий пассажиров. Взято из работы [11]

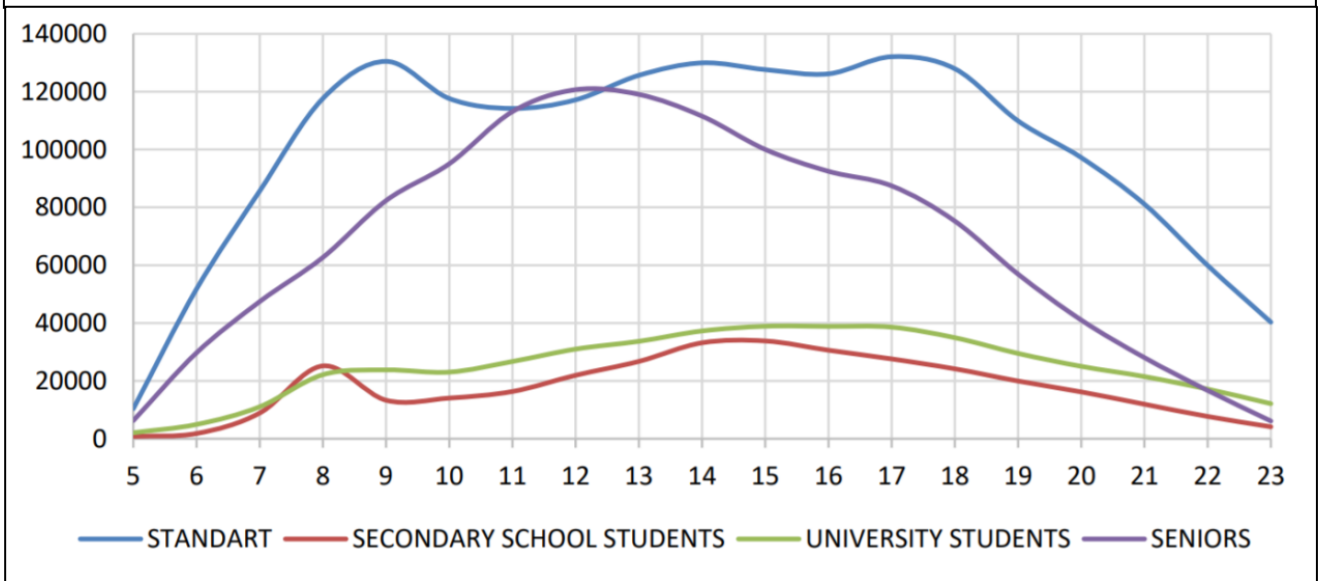

Рис. 14. Распределение пассажиропотока на городском транспорте по часам в выходные дни для разных категорий пассажиров. Взято из работы [11]

Из представленных графиков видно, что для коммерческого и городского транспорта характерно одинаковое распределение пассажиропотока среди пассажиров, не имеющих социальных льгот.

По будням, в часы, когда пассажиры едут на работу и с работы, наблюдается резкое увеличение количества пассажиров и на городских и на коммерческих маршрутах, а по выходным пассажиропоток и там и там распределяется равномерно между 9 и 20-21 часами. Из этого можно сделать вывод, что пассажиропоток на коммерческом транспорте распределяется соответственно пассажиропотоку на городском транспорте и по наполненности транспортных средств на городских маршрутах можно судить о наполненности их на похожих коммерческих маршрутах и, как следствие, рассчитать пассажиропоток на коммерческом транспорте.

Оценка пассажиропотока по транзакционным данным

В основе данного подхода лежит идея применимости большого количества транзакционных данных об оплате проезда с использованием смарт-карт «Подорожник», полученных СЭКОП на коммерческих маршрутах, для оценки количества пассажиров, совершающих поездки по этим маршрутам.

Основной сложностью в работе с такими данными является отсутствие информации о местах посадки и высадки пассажиров, следующих по маршруту, т.к., как было описано выше, на коммерческих маршрутах нет gps-датчиков, позволяющих отслеживать местоположение транспортных средств. Для вычисления места посадки используется следующее предположение: оплата проезда на коммерческих маршрутах происходит в начале поездки. Использование данного предположения оправдано тем, что в Санкт-Петербурге оно выполняется почти для всего коммерческого транспорта.

Методика заключается в следующем. По указанным транзакционным данным, информации о маршрутах, полученной в результате ручного анализа, и данных наблюдателей, для каждой поездки по исследуемым маршрутам можно найти остановки, на которых пассажиры садились в транспортные средства, а на основании данных о корреспонденциях – где они выходили. После этого с использованием полученных данных необходимо построить матрицы отправления назначения для каждого маршрута и по ним рассчитать

пассажиропоток среди людей, оплативших проезд смарт-картами. Для восстановления общего количество пассажиров, следующих по маршрутам, по данным наблюдателей рассчитываются коэффициенты, показывающие разницу между реальным количеством пассажиров и теми, кто оплачивает проезд по карточкам. Данные коэффициенты различны для будней и выходных дней, а также для различных временных промежутков (идет деление на утро, день и вечер).

Оценка пассажиропотока по данным наблюдателей

Идея использования данных наблюдателей для оценки пассажиропотока, в том числе для коммерческих маршрутов, не нова. Но для получения качественных интерпретируемых результатов, которые обладают способностью к обобщению, необходимо провести улучшение данного метода. Требуется определить, на какие маршруты оптимально ставить наблюдателей, в каком виде проводить наблюдения, а также как обеспечить постоянный приток наблюдателей.

После проведения анализа имеющихся данных наблюдателей и характеристик коммерческих маршрутов в Санкт-Петербурге, я выделила несколько важных идей, которые позволят облегчить процедуру поиска наблюдателей, уменьшат число требуемых наблюдений и позволят получить результаты, которые можно будет с некоторой степенью точности обобщить на ряд других коммерческих.

- Кластеризовать коммерческие маршруты по «похожести». Кроме уже описанных характеристик похожих маршрутов можно еще указать наличие одинаковых станций метро на маршруте.
- Выделить «ключевых представителей» для каждого кластера – маршруты, по которым можно будет оценить обстановку сразу в некотором микрорайоне.
- Разделить наблюдения по будням и выходным дням, а также по временным промежуткам (утро 7:00 – 10:00, день 12:00 – 15:00, вечер 17:00 – 20:00 и поздний вечер 21:00 – 00:00).

Данные идеи позволят составить достаточно полную картину о пассажиропотоке на коммерческом транспорте в Санкт-Петербурге.

Оптимальным вариантом проведения наблюдений является регистрация наблюдателем, находящимся в транспортном средстве количества входящих и выходящих пассажиров на каждой остановке маршрута в заранее подготовленной шаблонной форме. Это позволит свести к минимуму количество ошибок, допускаемых при наблюдении, а также облегчит последующую обработку полученных данных.

Важным аспектом также является поиск и привлечение наблюдателей. Для поиска наблюдателей я могу рекомендовать платформу Яндекс Толока. Преимуществами использования данной платформы является большое количество пользователей, готовых выполнять задания, умеренные гонорары, которые необходимо выплачивать за выполнение заданий, возможность использования шаблонов для создания полевых заданий. А также автоматизированный сбор статистики и решений.

Помимо основных наблюдателей, задача которых целенаправленно ездить на транспортных средствах и вести статистику количества пассажиров, к наблюдениям можно также привлекать пассажиров, которые пользуются коммерческим транспортом в рамках своих ежедневных перемещений. Такие исполнители могут, например, вести статистику по пути на работу и с работы, записывая количество пассажиров, проезжающих с ними в одном транспортном средстве. Данные задания можно также размещать на Яндекс Толоке. Их особенностью является малая стоимость, при том, что количество исполнителей, готовых за него взяться, будет достаточно большим (т.к. такие задания не предполагают большого количества накладных ресурсов для исполнителей).

В качестве меры по привлечению наблюдателей также можно анонсировать задания на сайте «Организатора перевозок», а также на других сайтах и в приложениях, которые занимаются построением транспортных маршрутов в Санкт-Петербурге (Яндекс Транспорт, Wikiroutes и т.п.).

Реализация методик

В этом разделе описываются особенности реализации указанных выше методик и применения их к обработке имеющихся данных, а также полученные результаты. Для каждой из трех методик описаны используемые данные, необходимые дополнительные этапы предобработки этих данных и алгоритмы, позволяющие оценить пассажиропоток.

1. Оценка пассажиропотока по данным о похожих городских маршрутах

Основной задачей для реализации данной методики становится разработка алгоритма поиска городских маршрутов, которые похожи на анализируемые коммерческие.

Используемые данные

1. Информация об остановках в прямом и обратном направлении на городских и коммерческих маршрутах.
2. Информация о пассажиропотоке на городском транспорте в требуемые даты и время.
3. Информация о пассажировместимости транспортных средств, следующих по интересуемым коммерческим и похожим городским маршрутам.

Обработка данных

С учетом описанной выше особенности транспортной системы Санкт-Петербурга (см. описание данной методики в разделе «Методология»), первое, что было сделано в рамках подготовки данных, это присвоение остановкам на маршрутах информации о кластерах, которым они принадлежат. Это действие является необходимым для корректного нахождения общих перегонов. Информация о пассажировместимости транспортных средств собиралась вручную.

Описание алгоритма поиска похожих маршрутов

На вход алгоритму передается номер исследуемого коммерческого маршрута, тип вывода результата и настраиваемый параметр. Типов вывода результата предусмотрено два. Первый – вывод указанного количества наиболее похожих городских маршрутов (тогда настраиваемый параметр – это требуемое количество), а второй – вывод городских маршрутов, у которых процент совпадения с коммерческим не меньше переданного значения (в процентах). Процент совпадения рассчитывается как отношение количества совпадающих перегонов к общему количеству перегонов на коммерческом маршруте.

Далее происходит поочередное сравнение коммерческого маршрута со всеми городскими маршрутами. В результате каждого такого сравнения составляется список общих цепочек перегонов коммерческого и текущего городского маршрутов. По окончании сравнения выбираются те городские маршруты, у которых больше всего совпадающих перегонов с исследуемым коммерческим маршрутом.

Вспомогательный алгоритм нахождения всех непересекающихся последовательностей общих перегонов у двух маршрутов

Основной интерес представляет реализация лежащего в основе алгоритма поиска всех общих последовательностей перегонов у двух сравниваемых маршрутов. Для этого был разработан вспомогательный алгоритм.

На вход вспомогательному алгоритму передаются массивы кластеров остановок на двух анализируемых маршрутах, а сам он состоит из следующих шагов.

1. **шаг:** Находится максимальная общая последовательность перегонов у двух маршрутов и добавляется в массив цепочек общих перегонов.
2. **шаг:** Найденная общая цепочка перегонов «вырезается» из обоих маршрутов и производится «склейка» оставшихся частей маршрутов.

3 шаг: Действия 1 и 2 повторяются на новых цепочках, пока одна из них не станет длины 0 или пока у них не закончатся общие перегоны.

В итоге в результирующем массиве общих перегонов будут лежать все общие цепочки перегонов (отсортированные по убыванию длины).

Корректность вспомогательного алгоритма

Утверждение. В результате работы такого алгоритма для каждой пары маршрутов мы получим корректный список всех непересекающихся цепочек общих перегонов.

Доказательство. Для доказательства корректности необходимо показать, что при выполнении операций «вырезания» и «склейки» частей маршрута у нас не образуется новых некорректных общих цепочек перегонов и не произойдет незапланированное удаление имеющихся.

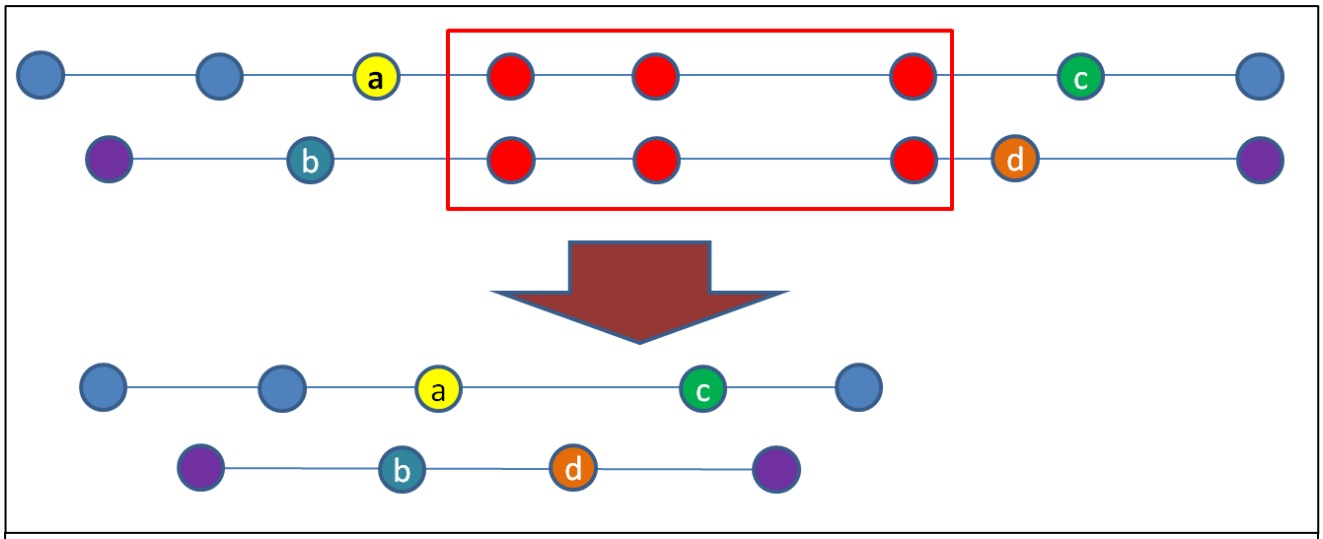

Рис. 15. Иллюстрация к доказательству корректности вспомогательного алгоритма

На **рис. 15** изображены две цепочки перегонов на разных маршрутах – сверху до «вырезания» найденной алгоритмом на шаге 1 цепочки общих перегонов (на рисунке выделена красным), а снизу – после операции «склейки». Т.к. найденная цепочка общих перегонов была максимальной по длине, отсюда следует, что остановки a и b принадлежали разным кластерам. Аналогично остановки c и d также принадлежали разным кластерам. После проведения операции «склейки» образуются новые некорректные перегоны a-c и b-d

(некорректные в том плане, что изначально их не существовало). Но с учетом сказанного выше, они не будут распознаны как общие по определению общих перегонов, т.к. их начальные и конечные остановки не принадлежат соответственно одним кластерам. Значит, новые образовавшиеся перегоны в дальнейшем не войдут ни в одну найденную цепочку общих перегонов, поэтому новых некорректных общих цепочек перегонов не образуется. Вместе с этим, все цепочки перегонов до остановок a и b и после остановок c и d сохранятся в таком же виде, а значит, что существующие на них общие перегоны в дальнейшем будут рассмотрены и обнаружены. **Ч.т.д.**

Обработка результатов вспомогательного алгоритма

После получения для каждого городского маршрута списка общих с исследуемым коммерческим маршрутом перегонов, происходит сортировка городских маршрутов по убыванию количества совпадающих перегонов. Также на этом этапе выполняется анализ совпадающих направлений – вычисляется, какое направление движения на коммерческом маршруте (прямое или обратное) совпадает с каким направлением движения на каждом из городских. Результатом работы алгоритма становится структурированный вывод информации о требуемом количестве похожих маршрутов с указанием совпадающих направлений движения и процента совпадения.

Пример работы алгоритма поиска похожих маршрутов

Пример работы алгоритма с первым типом вывода результата. На вход передается номер исследуемого коммерческого маршрута, тип вывода результата “number”, что означает, что нас интересует определенное количество наиболее похожих маршрутов и численное значение этого требуемого количества (**рис. 16**).

```
find_most_similar_routes(68, 'number', 5)
```

```
Для коммерческого маршрута № 68 найдены следующие наиболее похожие:

Прямое направление маршрута №68 совпадает с обратным направлением автобусного маршрута № 68, процент совпадения:  100 %

Прямое направление маршрута №68 совпадает с прямым направлением троллебусного маршрута № 37, процент совпадения:  83 %

Прямое направление маршрута №68 совпадает с прямым направлением автобусного маршрута № 130, процент совпадения:  67 %

Прямое направление маршрута №68 совпадает с прямым направлением троллебусного маршрута № 46, процент совпадения:  44 %

Обратное направление маршрута №68 совпадает с прямым направлением троллебусного маршрута № 44, процент совпадения:  44 %
```

Рис. 16. Пример результата работы алгоритма. Вывод пяти наиболее похожих маршрутов

Пример работы алгоритма со вторым типом вывода результата. На вход передается номер исследуемого коммерческого маршрута, тип вывода результата “percent”, что означает, что нас интересует определенный пороговый процент похожести и численное значение этого порогового процента **(рис. 17)**.

```
find_most_similar_routes(67, 'percent', 50)
```

```
Для коммерческого маршрута № 67 найдены следующие наиболее похожие:

Обратное направление маршрута №67 совпадает с обратным направлением автобусного маршрута № 67, процент совпадения:  91 %

Прямое направление маршрута №67 совпадает с прямым направлением автобусного маршрута № 70, процент совпадения:  82 %

Обратное направление маршрута №67 совпадает с прямым направлением автобусного маршрута № 71, процент совпадения:  82 %

Обратное направление маршрута №67 совпадает с прямым направлением автобусного маршрута № 65, процент совпадения:  82 %

Прямое направление маршрута №67 совпадает с прямым направлением автобусного маршрута № 67Б, процент совпадения:  73 %
```

Рис. 17. Пример результата работы алгоритма. Вывод всех городских маршрутов, процент совпадения которых с указанным коммерческим не менее 50

Применение методики к оценке пассажиропотока

Для проведения комплексного исследования пассажиропотока на коммерческом маршруте с использованием описанного алгоритма, необходимо выполнить следующие шаги.

1 **Шаг:** Выбрать до пяти городских маршрутов, которые больше всего похожи на рассматриваемый коммерческий и, по возможности, образуют полное покрытие его перегонов.

2 **Шаг:** Рассмотреть пассажиропоток на выбранных городских маршрутах в следующие временные промежутки:

   a. В будние дни (пн-пт) утром (с 7:00 до 11:00), днем (с 11:00 до 17:00), вечером (с 17:00 до 21:00) и поздним вечером (после 21:00).
   b. В выходные дни (сб-вс) утром (с 7:00 до 11:00), днем (с 11:00 до 21:00) и поздним вечером (после 21:00).

3 **Шаг:** Вычислить коэффициенты загруженности городского транспорта в эти временные промежутки, посчитав отношение реального количества пассажиров к максимальному значению пассажировместимости транспортных средств, следующих по маршрутам.

4 **Шаг:** Спрогнозировать пассажиропоток на коммерческих маршрутах. Пассажиропоток будет рассчитан для каждого перегона на маршруте в каждый исследуемый промежуток времени следующим образом:

   a. Находим значения коэффициентов загруженности похожих городских маршрутов на этом перегоне в этот временной промежуток (для тех маршрутов, у которых этот перегон также входит в состав маршрута).
   b. Усредняем значение полученных коэффициентов. Усреднять следует с использованием весовых коэффициентов, отдавая предпочтение коэффициентам загруженности на маршрутах с наибольшим процентом совпадения. Коэффициенты предлагается вычислить следующим образом:

$$coeff_i = \frac{similarity\_percent(route_i)}{\sum_{route}^{all\ routes} similarity\_percent(route)}$$, где

      - $all\ routes$ – это все городские маршруты, отобранные на шаге 1
      - $similarity\_percent(route_i)$ – процент совпадения i-ого городского маршрута с исследуемым коммерческим.

   c. Умножаем полученный коэффициент на значение максимальной вместимости коммерческого транспортного средства.

5 **Шаг:** Повторим шаг 4 для всех перегонов на интересуемом маршруте и для всех интересуемых временных промежутков и получим статистику пассажиропотока на коммерческом маршруте.

2. Оценка пассажиропотока по транзакционным данным

Основной задачей для реализации данной методики становится разработка алгоритма вычисления мест посадки пассажиров по транзакциям оплаты проезда смарт-картами, т.к. из-за отсутствия данных АСУГПТ на коммерческих маршрутах, автоматически место совершения транзакции в системе не фиксируется.

Базовое предположение, лежащее в основе данной методики: транзакции оплаты совершаются сразу при входе пассажира в транспортное средство. Оно выполняется на преимущественном большинстве коммерческих маршрутов Санкт-Петербурга.

Используемые данные

1. Информация о транзакциях оплаты проезда смарт-картами на коммерческих маршрутах.
2. Информация об остановках на коммерческих маршрутах, расстоянии между ними и времени преодоления этого расстояния, средней скорости транспортных средств на маршруте.
3. Информация наблюдателей о пассажиропотоке на коммерческих маршрутах.

Обработка данных

Информация о транзакциях оплаты проезда смарт-картами была детально исследована на корректность. В результате проведенного анализа в данных о транзакциях были обнаружены некорректные записи. На коммерческом маршруте №68 количество дневных транзакций оплаты по картам у транспортного средства с бортовым номером 0 аномально большое. Потенциальной причиной

возникновения некорректных данных можно считать сбой в системе в октябре 2018 года, который привел к потере информации о бортовых номерах у ряда транспортных средств, следующих по маршруту №68 и в результате произошло объединение данных о нескольких транспортных средствах (с новым общим бортовым номером 0). Т.к. разделить такие данные не представляется возможным, их нельзя учитывать при расчете пассажиропотока по этой методике.

Информация о расстоянии между остановками, времени их преодоления и средней скорости транспортных средств на маршрутах была собрана вручную и добавлена к уже имеющейся информации об остановках на маршрутах и их кластерах.

Описание алгоритма восстановления мест посадки пассажиров

Созданный алгоритм базируется на идеях, описанных в статье [9] и адаптирован под особенности системы коммерческого транспорта Санкт-Петербурга.

Алгоритм работает с общим списком всех транзакций на коммерческих маршрутах и заранее известным списком интересуемых маршрутов. Для всех транзакций на интересуемых маршрутах за весь период исследования алгоритм восстанавливает номер поездки, направление движения транспортного средства и идентификатор остановки, на которой садился пассажир, совершивший эту транзакцию.

Алгоритм состоит из следующих шагов.

1 **Шаг:** На первом этапе происходит разделение транзакций по поездкам. Каждой отдельной поездке назначается свой идентификатор. Отделение различных поездок друг от друга происходит по принципу, описанному на блок-схеме (**рис. 18**). К разным поездкам будут отнесены записи, временная разница между которыми больше 12 минут, а также запись будет считаться началом новой поездки, если с начала предыдущей прошло времени больше, чем требуется транспортному средству, чтобы преодолеть маршрут по

расписанию. Здесь параметр 12 минут был выбран практически на основе анализа имеющихся данных как оптимальный разделитель.

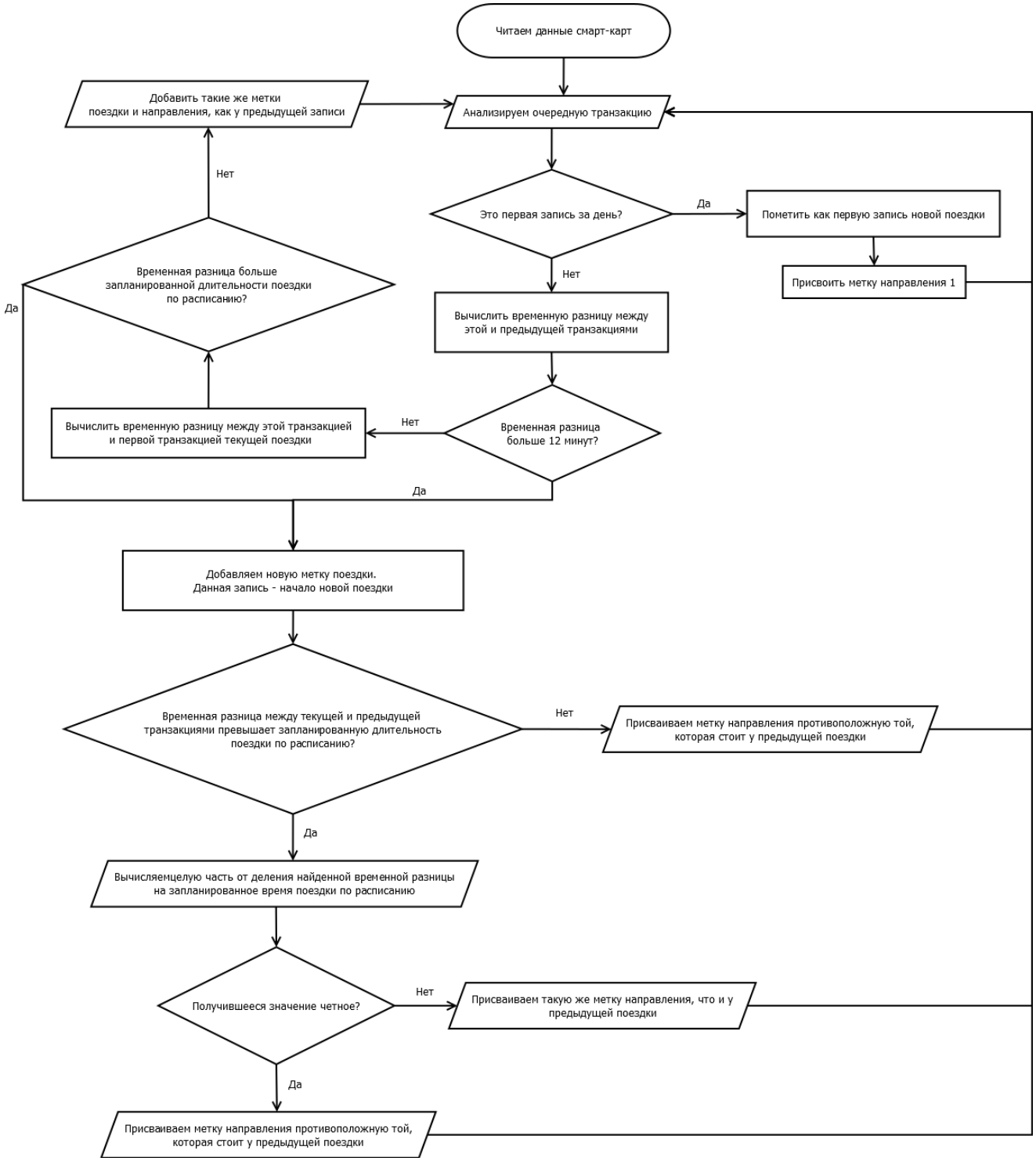

Рис. 18. Блок-схема алгоритма определения номера поездки и направления движения

2 **Шаг:** На втором этапе производится кластеризация записей. Маленький временной промежуток между транзакциями означает, что совершившие их пассажиры садились в одном месте, а значит такие записи нужно отнести к одному кластеру. Более подробно данный этап описан на блок-схеме (**рис. 19**).

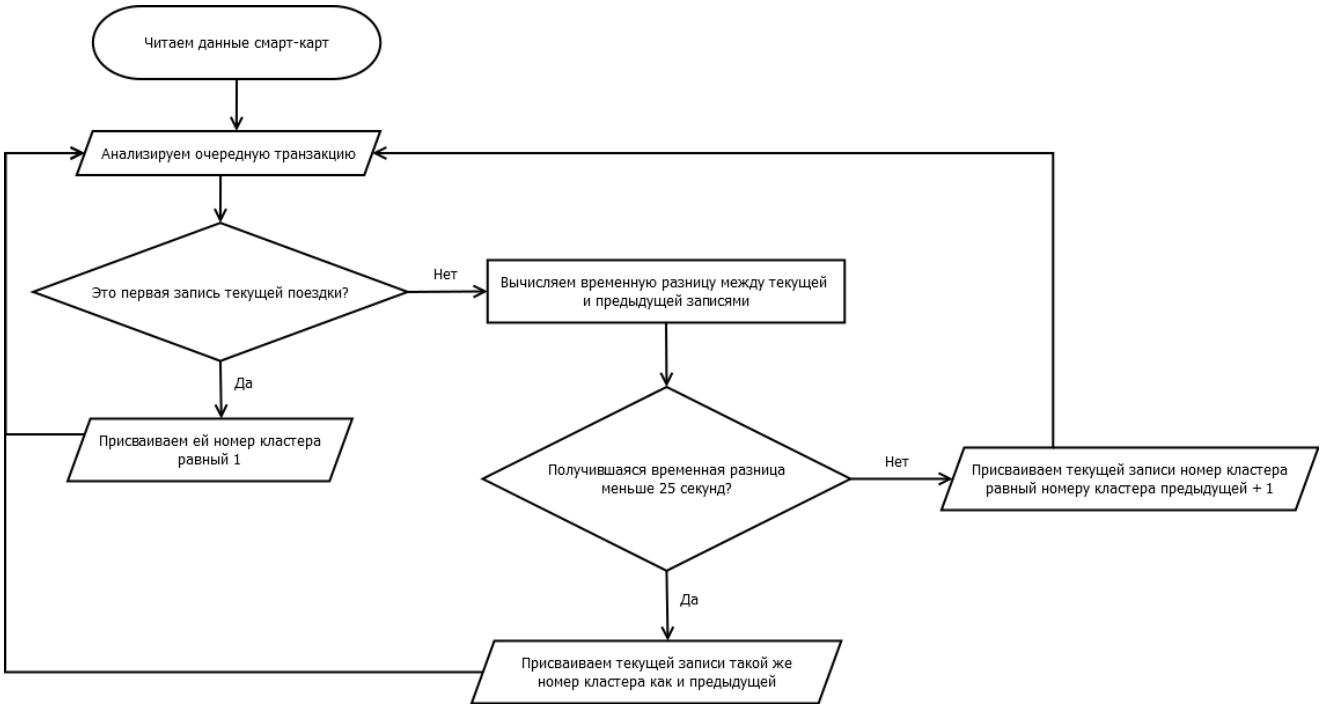

Рис. 19. Блок-схема алгоритма кластеризации записей

3 **Шаг:** Целью третьего этапа является соотнесение полученных кластеров и остановок на маршруте. В результате для каждой транзакции должна появиться информация об остановке, на которой входил совершивший ее пассажир.

Алгоритм соотнесения кластеров и остановок, который реализуется на данном шаге, выглядит следующим образом:

a. Первому кластеру каждой поездки ставится в соответствие первая остановка маршрута (это базовое предположение, его можно изменить, если станет известна дополнительная информация о том, как это можно вычислить более точно).

b. Обозначим за $t_{n(n+1)}$ временную разницу между первой записью в $n+1$ кластере и последней записью в n кластере. Пусть кластер n был соотнесен с остановкой i. Обозначим за $a_{i(i+1)}$ время прохождения транспортным средством по маршруту от остановки i до остановки $i+1$.

c. Если $t_{n(n+1)} < a_{i(i+1)}$, то соотнесем кластер $n+1$ с остановкой $i+1$.

d. Если$t_{n(n+1)} > a_{i(i+1)}$, то сравним $t_{n(n+1)}$ и $a_{i(i+2)}$. Если снова больше, то будем сравнивать $t_{n(n+1)}$ с $a_{i(i+k)}$ до тех пор, пока не будет выполнено $t_{n(n+1)} < a_{i(i+k)}$. После этого соотнесем кластер $n+1$ с остановкой $i+1$.

Для наглядности описанный алгоритм изображен на **рис. 20**.

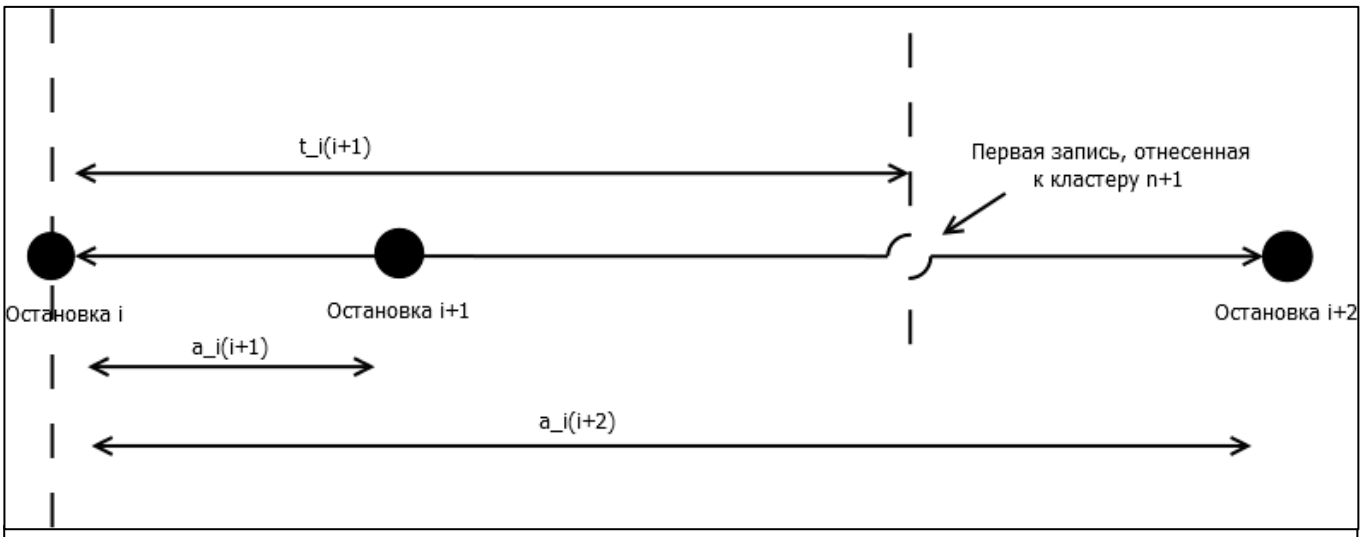

Рис. 20. Иллюстрация работы алгоритма, соотносящего кластеры записей и остановки на маршруте

4 Шаг: На четвертом шаге происходит корректировка полученной информации. Необходимость корректировки связана со следующей особенностью коммерческих транспортных средств в Санкт-Петербурге. Очень часто водители, завершив поездку (особенно у метро) и высадив людей, начинают сразу запускать новых пассажиров, которые сразу оплачивают проезд. Таким образом, становится невозможным отследить начало новой поездки на этапе 1. Но после того, как записям на третьем шаге присваиваются идентификаторы остановок это становится возможным.

Характерной особенностью таких записей является скопление записей с разными идентификаторами поездок, но одинаковыми идентификаторами остановок (**рис. 21**).

CARRIER_BOARD_NUM	TRANSACT_TIME	ROUTE_NUM	TRIP_ID	DIRECTION	BOARDING_CLUSTER	BOARDING_STOP_NUM	BOARDING_STOP_ID
67	08:55:00	92	3	1	5	10	15952
67	08:57:00	92	3	1	6	10	15952
67	08:57:00	92	3	1	6	10	15952
67	08:57:00	92	3	1	6	10	15952
67	08:57:00	92	3	1	6	10	15952
67	08:57:00	92	3	1	6	10	15952
67	09:07:00	92	4	0	1	0	15952
67	09:10:00	92	4	0	2	1	1847
67	09:11:00	92	4	0	3	2	4116
67	09:12:00	92	4	0	4	3	3485

Рис. 21. Пример скопления транзакций, совершенных на одной остановке, но отнесенных к разным поездкам. Идентификатор 15952 соответствует автобусной остановке «Ст.м. Ладожская»

Корректировка, которая вносится в данные – присвоение всем таким записям нового идентификатора поездки, т.к. фактически все подобные транзакции относятся к началу следующей поездки. Результат корректировки показан на **рис. 22**. Он же и является итоговым результатом работы алгоритма.

CARRIER_BOARD_NUM	TRANSACT_TIME	ROUTE_NUM	TRIP_ID	DIRECTION	BOARDING_CLUSTER	BOARDING_STOP_NUM	BOARDING_STOP_ID
67	08:55:00	92	4	0	1	0	15952
67	08:57:00	92	4	0	1	0	15952
67	08:57:00	92	4	0	1	0	15952
67	08:57:00	92	4	0	1	0	15952
67	08:57:00	92	4	0	1	0	15952
67	08:57:00	92	4	0	1	0	15952
67	09:07:00	92	4	0	1	0	15952
67	09:10:00	92	4	0	2	1	1847
67	09:11:00	92	4	0	3	2	4116
67	09:12:00	92	4	0	4	3	3485

Рис. 22. Результат корректировки данных. Теперь все такие записи относятся к началу новой поездки.

Применение методики к оценке пассажиропотока

Для проведения исследования пассажиропотока на коммерческом маршруте с использованием описанного алгоритма, необходимо выполнить следующие шаги.

1. **Шаг:** Загрузить полученные в результате выполнения описанного алгоритма данные о транзакциях и местах посадки пассажиров на транспортных средствах, следующих по интересуемому маршруту.
2. **Шаг:** Используя полученные данные, а также существующий метод корреспонденций и имеющиеся данные о корреспонденциях на всем пассажирском транспорте Санкт-Петербурга рассчитать места выхода пассажиров, которые оплатили проезд смарт-картами.
3. **Шаг:** Рассчитать реальное количество вошедших и вышедших на каждой остановке пассажиров, используя коэффициенты различия между реальным количеством пассажиров и количеством пассажиров, оплативших проезд смарт-картами.
4. **Шаг:** Вычислить количество пассажиров, которые ехали между каждой парой остановок по данным о входе и выходе.

Анализ применимости описанных методик

Из трех рассмотренных методик наиболее точной и надежной является оценка пассажиропотока на основе данных наблюдателей. Она позволяет получить 100% верные характеристики пассажиропотока как на городском, так и на коммерческом транспорте. Тем не менее, у данной методики очень большие накладные расходы, т.к. требуется привлечение сторонних наблюдателей. В своей работе я описала, как можно оптимизировать накладные расходы, уменьшить требуемое количество наблюдений и привела ряд идей по привлечению наблюдателей, которые позволят упростить данный процесс.

Методика оценки пассажиропотока на основе данных о пассажиропотоке на похожих городских маршрутах также достаточна точна и надежна, т.к. в ее основе лежит использование проверенных данных о пассажиропотоке на городском транспорте. Алгоритм поиска похожих маршрутов хорошо протестирован и выдает корректные списки городских маршрутов, у которых большая часть перегонов совпадает с исследуемым коммерческим.

Использование методики оценки пассажиропотока на коммерческом транспорте по транзакционным данным об оплате проезда, требует корректности данных о транзакциях. В ходе исследования были выявлены и описаны ошибки, возникающие при сборе таких данных. Для получения корректного результата оценки пассажиропотока необходима качественная предобработка имеющихся данных.

Отдельно стоит отметить, что качественный сбор транзакционных данных об оплате проезда смарт-картами и составление электронного справочника с полным перечнем остановок на коммерческих маршрутах позволят значительно улучшить качество описанных выше методик и автоматизировать этапы, которые выполнялись вручную.

Заключение

В работе рассмотрена актуальная проблема в области оценки характеристик пассажирского транспорта – оценка пассажиропотока на коммерческих маршрутах Санкт-Петербурга. Для решения данной проблемы были разработаны и описаны три методики. Описание включает в себя перечень требуемых данных и способы их предобработки, алгоритмы, лежащие в основе методик и особенности их реализации. Разработанные методики были представлены на всероссийской конференции «Список 2019». Все разработанные и описанные алгоритмы были реализованы и протестированы. Исходные коды алгоритмов и данные доступны по ссылке: https://github.com/nast1415/vkr2019. Надеюсь, что описанные методики окажутся полезными, будут применяться для анализа пассажиропотока на коммерческом транспорте Санкт-Петербурга и позволят оптимизировать существующую транспортную систему.

Список литературы

1. M. Bagchi and P. R. White, «The potential of public transport smart card data», 2005.
2. C. Chen, J.Ma, Y. Susilo, Y. Liu, andM.Wang, «The promises of big data and small data for travel behavior (aka human mobility) analysis», 2016
3. Tian Li, Dazhi Sun, Peng Jing, Kaixi Yang «Smart Card Data Mining of Public Transport Destination: A Literature Review», 2018
4. Barry, J.J.; Newhouser, R.; Rahbee, A.; Sayeda, S. «Origin and Destination Estimation in New York City with Automated Fare System Data», 2002
5. Zhao, J.; Qu, Q.; Zhang, F.; Xu, C.; Liu, S. «Spatio-temporal Analysis of Passenger Travel Patterns in Massive Smart Card Data», 2017
6. Dou, H.; Liu, H.; Yang, X. «OD Matrix Estimation Method of Public Transportation Flow Based on Passenger Boarding and Alighting», 2007
7. Jie, Y.U.; Yang, X.G. «Estimation a Transit Route OD Matrix Using On/off Data: An Application of Modified BP Artificial Neural Network», 2006
8. Jung, J.; Sohn, K. «Deep-learning Architecture to Forecast Destinations of Bus Passengers from Entry-only Smart-card Data», 2017
9. Chen Z.; Fan W. «Extracting bus transit boarding stop information using smart card transaction data», 2018
10. Pau Segarra Algueró «Using Smart Card Technologies to Measure Public Transport Performance: Data Capture and Analysis», 2013
11. Natalia Grafeeva, Elena Mikhailova, Elena Nogova, Innokenty Tretyakov, Passanger Traffic Analysys Based on St. Petersburg Public Transport, 17th International Multidisciplinary Scientific GeoConference: Informatics, Geoinformatics and Remote Sensing, Issue 21, SGEM 2017; Albena; Bulgaria; Volume 17, 2017, Pages 509-516

Приложение 1. Функции, реализующие основные алгоритмы

В данном приложении приводится код функций, реализующих два основных алгоритма, описываемых в данной работе: алгоритма поиска похожих маршрутов и алгоритма восстановления места посадки пассажиров.

Алгоритм поиска похожих маршрутов

Функция lcs, которая находит максимальную общую последовательность перегонов и две вспомогательные функции к ней.

```
def sequential_slice(iterable, length):
    pool = tuple(iterable)
    assert 0 < length <= len(pool)
    tails = (pool[s:] for s in range(length))
    return zip(*tails)

def sequence_in_list(sequence, lst):
    pool = tuple(sequence)
    return any((pool == s for s in sequential_slice(lst, len(pool))))

def lcs(str1, str2):
    a = str1.split(',')
    b = str2.split(',')

    if len(a) > len(b):
        a, b = b, a
    for l in reversed(range(1, len(a)+1)):
        seq = [subseq for subseq in sequential_slice(a, l) if sequence_in_list(subseq, b)]
        if seq:
            break
    return seq
```

Функция, возвращающая полный список непересекающихся последовательностей общих перегонов.

```
def create_list_of_subroutes(str1, str2):
    res = 1
    all_subroutes = []
    while len(str1) > 0 and len(str2) > 0 and res != 0:
        seq = lcs(str1, str2)
        # Если нет общих перегонов или остались только общие остановки, мы завершаем работу
        # (больше нет общих перегонов)
        if len(seq) == 0 or len(seq[0]) == 1:
            res = 0
```

```
    # Иначе мы добавляем наибольший общий в итоговый список всех общих и удаляем из обоих маршрутов
    # эту последовательность
    else:
        all_subroutes.append(seq[0])
        substr = seq[0][0]
        for i in seq[0][1:]:
            substr += "," + i
        # удаление общей подпоследовательности из маршрутов
        new_str1_array = str1.split(substr)
        if len(new_str1_array) == 1:
            str1 = str1.split(substr)[0]
            str1 = str1.replace(',,',',')
        else:
            str1 = str1.split(substr)[0] + str1.split(substr)[1]
            str1 = str1.replace(',,',',')

        new_str2_array = str2.split(substr)
        if len(new_str2_array) == 1:
            str2 = str2.split(substr)[0]
            str2 = str2.replace(',,',',')
        else:
            str2 = str2.split(substr)[0] + str2.split(substr)[1]
            str2 = str2.replace(',,',',')
amount_of_similar = 0 # Переменная, в которую мы запишем количество общих перегонов (чтобы в дальнешем сортировать)
for i in range(len(all_subroutes)):
    amount_of_similar += len(all_subroutes[i]) - 1
return all_subroutes, amount_of_similar
```

Функция, которая возвращает указанное количество городских маршрутов, которые больше всего похожи на наш.

```
def most_similar(comm_route, number_of_similar):
    comm_clusters1 = comm_df[comm_df['route_number'] == comm_route]['clusters_in_direction1'].values[0]
    comm_clusters2 = comm_df[comm_df['route_number'] == comm_route]['clusters_in_direction2'].values[0]

    # Создаем zip с парами номер маршрута - тип транспорта для использования в алгоритме
    all_routes_names = all_routes_df['route_number'].values
    all_routes_types = all_routes_df['id_transport'].values
    all_routes = zip(all_routes_names, all_routes_types)

    choosen_direction = 1
    direction_of_result = 1

    similarity_arr = []
    max_subroute = []
    for route_name, route_type in all_routes:
        if comm_clusters1 != '-1':
            all_routes_this_type = all_routes_df[all_routes_df['id_transport'] == route_type]
            current_clusters_1 =
                all_routes_this_type[all_routes_this_type['route_number'] == route_name]['clusters_in_direction1'].values[0]
            current_clusters_2 =
                all_routes_this_type[all_routes_this_type['route_number'] == route_name]['clusters_in_direction2'].values[0]
            subroutes1, amount1 = create_list_of_subroutes(comm_clusters1, current_clusters_1)
            subroutes2, amount2 = create_list_of_subroutes(comm_clusters1, current_clusters_2)
            if amount2 > amount1:
                subroutes_1 = subroutes2
                amount_1 = amount2
                direction_of_result = 2
            else:
                subroutes_1 = subroutes1
                amount_1 = amount1
                direction_of_result = 1
```

```
        all_routes_this_type = all_routes_df[all_routes_df['id_transport'] == route_type]
        current_clusters_1 =
            all_routes_this_type[all_routes_this_type['route_number'] == route_name]['clusters_in_direction1'].values[0]
        current_clusters_2 =
            all_routes_this_type[all_routes_this_type['route_number'] == route_name]['clusters_in_direction2'].values[0]
        subroutes3, amount3 = create_list_of_subroutes(comm_clusters2, current_clusters_1)
        subroutes4, amount4 = create_list_of_subroutes(comm_clusters2, current_clusters_2)
        if amount4 > amount3:
            subroutes_2 = subroutes4
            amount_2 = amount4
            direction_of_result = 2
        else:
            subroutes_2 = subroutes3
            amount_2 = amount3
            direction_of_result = 1
    if amount_2 > amount_1:
        subroutes = subroutes_2
        amount = amount_2
        number_of_stops = len(comm_clusters2.split(','))
        choosen_direction = 2
    else:
        subroutes = subroutes_1
        amount = amount_1
        number_of_stops = len(comm_clusters1.split(','))
        choosen_direction = 1

    if len(subroutes) == 0:
        similarity_arr.append([(route_name, route_type), 0, []])
    else:
        similarity_arr.append([(route_name, route_type), amount, subroutes,
                               number_of_stops, choosen_direction, direction_of_result])
most_similar = sorted(similarity_arr, key=lambda x: x[1], reverse=True)[:number_of_similar]
return most_similar
```

Основная функция, которая возвращает указанное количество городских маршрутов, похожих на переданный коммерческий.

```
def find_most_similar_routes(comm_route, type_of_result, threshold):
    if type_of_result == "percent":
        number_of_similar = 5
    else:
        number_of_similar = threshold
    similar = most_similar(comm_route, number_of_similar)
    print('Для коммерческого маршрута № '+ str(comm_route) + ' найдены следующие наиболее похожие:')
    print()
    for route in similar:
        route_pair = route[0]
        route_name = route_pair[0]
        route_type = route_pair[1]
        number_of_stops = route[3]
        choosen_direction = route[4]
        direction_of_result = route[5]
        similarity_percent = round(route[1] * 100 / (number_of_stops - 1))
        if type_of_result == "percent":
            if similarity_percent < threshold:
                break
        direction = ""
        if choosen_direction == 1:
            ch_direction = "Прямое направление "
        else:
            ch_direction = "Обратное направление "
        if direction_of_result == 1:
            res_direction = "прямым направлением "
        else:
            res_direction = "обратным направлением "
        type_string = ""
        if route_type == 1:
            type_string = "автобусного "
        elif route_type == 2:
            type_string = "трамвайного "
        else:
            type_string = "троллебусного "
        print(ch_direction + "маршрута №" + str(comm_route) + " совпадает с " + res_direction + type_string +
              "маршрута № " + route_name + ", процент совпадения: ", similarity_percent, "%")
        print()
```

Алгоритм восстановления мест посадки пассажиров

Вспомогательные функции к основной функции алгоритма. Функция get_route_df_by_date принимает на вход информацию о номере маршрута и интересуемую дату и составляет dataset с информацией о транзакциях на указанном маршруте в указанную дату. Функция check_data возвращает true/false в зависимости от того, есть ли транзакционные записи об указанном маршруте в указанную дату. И функция get_sheduled_trip_time возвращает по номеру маршрута время его поездки по расписанию.

```
def get_route_df_by_date(route_num, tr_date):
    route_df = pd.read_csv('transactions_by_routes/' + route_num + '_route.csv')
    route_df = route_df[route_df['TRANSACT_MONTH'] == int(tr_date.split('.')[1])]
    route_df = route_df[route_df['TRANSACT_DAY'] == int(tr_date.split('.')[0])]
    route_df = route_df[route_df['CARRIER_BOARD_NUM'] <= 10000]
    route_df = route_df.sort_values(by=['TRANSACT_YEAR', 'TRANSACT_MONTH', 'TRANSACT_DAY', 'CARRIER_BOARD_NUM', 'TRANSACT_TIME'])
    route_df = route_df.reset_index(drop=True)
    return route_df[['CARD_NUM', 'TRANSACT_DAY', 'TRANSACT_MONTH', 'TRANSACT_YEAR', 'CARRIER_BOARD_NUM', 'TRANSACT_TIME',
                     'ROUTE_NUM']]


def check_date(route_num, tr_date):
    route_df = get_route_df_by_date(route_num, tr_date)
    return route_df['ROUTE_NUM'].count() > 0


def get_scheduled_trip_time(route_num):
    return comm_df[comm_df['route_number'] == int(route_num)]['scheduled_trip_time'].values[0]
```

Функция, которая реализует непосредственно сам алгоритм. Она принимает на вход номер маршрута и дату и состоит из четырех этапов, описанных в разделе «Реализация».

```
def get_boarding_locations_by_day(route_num, tr_date):
    cur_comm_df = comm_df[comm_df['route_number'] == int(route_num)]
    route_df = get_route_df_by_date(route_num, tr_date)
    board_nums_array = route_df['CARRIER_BOARD_NUM'].unique()
    for board_num in board_nums_array:
        board_df = route_df[route_df['CARRIER_BOARD_NUM'] == board_num]
        tr_amount = board_df.shape[0]
        tr_times = board_df['TRANSACT_TIME'].values
```

```
# Этап 1 - разделение поездок по времени между транзакциями
board_trips = [1]
board_directions = [0]
trip_id = 1
direction = 0
first_note_of_the_trip = 0
for i in range(1, tr_amount):
    sheduled_trip_duration = get_scheduled_trip_time(route_num)
    if (get_times_delta(tr_times[i - 1], tr_times[i]) >= coarse_trips_pivot) or
        get_times_delta(tr_times[first_note_of_the_trip], tr_times[i]) >= sheduled_trip_duration:
        trip_id += 1
        first_note_of_the_trip = i
        board_trips.append(trip_id)
        if get_times_delta(tr_times[i - 1], tr_times[i]) >= sheduled_trip_duration:
            dir_coeff = get_times_delta(tr_times[i - 1], tr_times[i]) // sheduled_trip_duration
            if dir_coeff % 2 == 0:
                direction = (direction + 1) % 2
                board_directions.append(direction)
            else:
                board_directions.append(direction)
        else:
            direction = (direction + 1) % 2
            board_directions.append(direction)
    else:
        board_trips.append(trip_id)
        board_directions.append(direction)

preliminary_trip_id_column = np.array(board_trips)
preliminary_direction_column = np.array(board_directions)
```

```
# Этап 2 - кластеризация записей
tr_times = board_df['TRANSACT_TIME'].values
board_clusters = [1]
cluster_id = 1
cur_trip_num = preliminary_trip_id_column[0]
for i in range(1, tr_amount):
    trip_id = preliminary_trip_id_column[i]
    if trip_id == cur_trip_num:
        if get_times_delta(tr_times[i - 1], tr_times[i]) < clusters_pivot:
            board_clusters.append(cluster_id)
        else:
            cluster_id += 1
            board_clusters.append(cluster_id)
    else:
        cur_trip_num = trip_id
        cluster_id = 1
        board_clusters.append(cluster_id)

clusters_column = np.array(board_clusters)
```

```
# Этап 3 - соотнесение кластеров и информации об остановках
tr_times = board_df['TRANSACT_TIME'].values
boarding_stations_nums = [0]
station_num = 0
cur_trip_num = preliminary_trip_id_column[0]
cur_direction = preliminary_direction_column[0]
cur_cluster = clusters_column[0]

if cur_direction == 0:
    stops_times = cur_comm_df['times_for_stops1'].values[0].split(',')
    route_stops_arr = cur_comm_df['stops_in_direction1'].values[0].split(',')
else:
    stops_times = cur_comm_df['times_for_stops2'].values[0].split(',')
    route_stops_arr = cur_comm_df['stops_in_direction2'].values[0].split(',')

boarding_stations_id = [route_stops_arr[0]]
station_id = route_stops_arr[0]
```

```
for i in range(1, tr_amount):
    trip_id = preliminary_trip_id_column[i]
    direction_id = preliminary_direction_column[i]
    cluster_id = clusters_column[i]
    if trip_id == cur_trip_num:
        if cluster_id != cur_cluster:
            cur_cluster = cluster_id
            delta = get_times_delta(tr_times[i - 1], tr_times[i])
            time_between_stations = 0
            finded = 0
            for j in range(station_num, len(stops_times)):
                time_between_stations += int(stops_times[j])
                if delta < time_between_stations:
                    finded = 1
                    station_num = j + 1
                    station_id = route_stops_arr[j + 1]
                    boarding_stations_nums.append(station_num)
                    boarding_stations_id.append(station_id)
                    break
            if finded == 0:
                station_num = len(stops_times)
                station_id = route_stops_arr[len(stops_times)]
                boarding_stations_nums.append(station_num)
                boarding_stations_id.append(station_id)
        else:
            boarding_stations_nums.append(station_num)
            boarding_stations_id.append(station_id)
```

```
    else:
        cur_trip_num = trip_id
        cur_direction = direction_id
        curr_cluster = cluster_id

        if cur_direction == 0:
            stops_times = cur_comm_df['times_for_stops1'].values[0].split(',')
            route_stops_arr = cur_comm_df['stops_in_direction1'].values[0].split(',')
        else:
            stops_times = cur_comm_df['times_for_stops2'].values[0].split(',')
            route_stops_arr = cur_comm_df['stops_in_direction2'].values[0].split(',')

        station_num = 0
        station_id = route_stops_arr[0]
        boarding_stations_nums.append(station_num)
        boarding_stations_id.append(station_id)


stops_num_column = np.array(boarding_stations_nums)
stops_id_column = np.array(boarding_stations_id)
```

```
# Этап 4 - корректировка информации о поездках
tr_times = board_df['TRANSACT_TIME'].values
tr_num = tr_times.shape[0]

trip_nums = preliminary_trip_id_column
directions = preliminary_direction_column
stops_id = stops_id_column
stops_numbers = stops_num_column
clusters = clusters_column

curr_trip = trip_nums[tr_num - 1]
curr_dir = directions[tr_num - 1]
curr_number = stops_numbers[tr_num - 1]
curr_cluster = clusters[tr_num - 1]

new_trips_ids = [curr_trip]
new_directions = [curr_dir]
new_clusters = [curr_cluster]
new_numbers = [curr_number]

pointer = tr_num - 1
```

```
while pointer > 0:
    if (trip_nums[pointer] != trip_nums[pointer - 1]) and (stops_id[pointer] == stops_id[pointer - 1]):
        while stops_id[pointer] == stops_id[pointer - 1]:
            new_trips_ids.append(curr_trip)
            new_directions.append(curr_dir)
            new_clusters.append(curr_cluster)
            new_numbers.append(curr_number)
            pointer -= 1
    else:
        curr_trip = trip_nums[pointer - 1]
        curr_dir = directions[pointer - 1]
        curr_cluster = clusters[pointer - 1]
        curr_number = stops_numbers[pointer - 1]

        new_trips_ids.append(curr_trip)
        new_directions.append(curr_dir)
        new_clusters.append(curr_cluster)
        new_numbers.append(curr_number)
        pointer -= 1
```

```
# Добавление информации, полученной на 1-4 этапах в dataframe
new_trips_column = pd.Series(np.array(list(reversed(new_trips_ids))), index=board_df.index)
board_df.loc[:, 'TRIP_ID'] = new_trips_column

new_dirs_column = pd.Series(np.array(list(reversed(new_directions))), index=board_df.index)
board_df.loc[:, 'DIRECTION'] = new_dirs_column

boarding_clusters = pd.Series(np.array(list(reversed(new_clusters))), index=board_df.index)
board_df.loc[:, 'BOARDING_CLUSTER'] = boarding_clusters

boarding_stops_nums = pd.Series(np.array(list(reversed(new_numbers))), index=board_df.index)
board_df.loc[:, 'BOARDING_STOP_NUM'] = boarding_stops_nums

boarding_stops_ids = pd.Series(stops_id_column, index=board_df.index)
board_df.loc[:, 'BOARDING_STOP_ID'] = boarding_stops_ids

# Сохраняем информацию
board_df.to_csv('transactions_by_routes/' + route_num + '_route_' +  str(board_num) + '_board_itog.csv')
```