Санкт-Петербургский государственный университет

На правах рукописи

Герштейн Аркадий Михаил

Программные инструменты для построения безопасных маршрутов транспорта

Научная специальность 2.3.5.
Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ

на соискание ученой степени кандидата физико-математических наук

Научный руководитель:

Доктор физико-математических наук,

профессор

Терехов Андрей Николаевич

Санкт-Петербург

2024

Содержание

Введение

- До встречи, - Гарри вместо того, чтобы пойти направо, повернул налево, выбирая более длинную, но безопасную дорогу к совам.

Джоан Роулинг, «Гарри Поттер и Орден Феникса»

В данной работе создан программный комплекс, позволяющий проложить более безопасные маршруты для автотранспорта. Необходимость создания такого комплекса обусловлена тем, что аварии на транспорте приводят к колоссальным людским потерям и травмам. Согласно данным WHO (World Health Organization) [https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries] за год около 1.3 миллиона человек во всем мире погибает в дорожных авариях и примерно 50 миллионов получают травмы различной степени тяжести.

Поэтому задача уменьшения числа дорожно-транспортных происшествий (ДТП) и, следовательно, понижения смертности и травматичности в ДТП представляется весьма важной и актуальной. Об этом также свидетельствуют многочисленные работы по теме (см. Главу 1, посвященную обзору литературы). Здесь же, во Введении, рассмотрим наиболее очевидные подходы к решению этой задачи, примененные в данной работе.

Первое, что приходит в голову — собрать данные о ДТП за несколько лет (координаты, степень тяжести и пр.), выделить из этих данных кластеры — места особой концентрации ДТП и затем обходить эти кластеры при построении более безопасного маршрута.

Чтобы продвинуться дальше, необходимо в общих чертах понять, как строится маршрут следования Транспортного средства (ТС). Если в реальности существуют отрезки дорог и перекрестки, то всевозможные навигаторы используют другое, более абстрактное представление дорог в городе — дорожную сеть, состоящую из ребер и вершин и называемую графом [1]. Ребра заменяют в расчетах сегменты дороги, а вершины обозначают переход от одного сегмента к другому (в результате, например, смены направления). Вершины также присутствуют на

перекрестках дорог, тогда вершина помечает место, где встречаются несколько сегментов дороги.

Дорожная сеть представляет собой *направленный* граф, поскольку многие ребра пропускают транспорт только в одну сторону. Есть также в дорожной сети двунаправленные ребра, по которым разрешено двигаться в обе стороны.

Пусть теперь каждое ребро дорожного графа обладает атрибутом, например, длиной ребра и нам известны начальная и конечная вершины маршрута. Тогда один из алгоритмов маршрутизации на дорожном графе (например, алгоритм Дейкстры [2]) способен построить маршрут минимальной длины из начальной вершины в конечную. Такой маршрут, возможно, окажется самым быстрым, но алгоритм маршрутизации никак не учитывает его безопасность. Если же взять в качестве атрибута грани число случившихся на соответствующем сегменте дороги ДТП, подсчитанное за несколько предшествующих лет, то такой маршрут хоть и будет самым безопасным, поскольку минимизирует число ДТП вдоль маршрута, вряд ли будет самым коротким[1]. Очевидно, необходим подход, который позволил бы построить компромиссный маршрут — более безопасный, и одновременно не очень протяженный (по сравнению с маршрутом минимальной длины). В Главе 1 разбираются различные подходы, позволяющие достичь этой цели, мы же здесь во Введении расскажем о нашем оригинальном подходе к этой проблеме.

Суть этого подхода состоит в выделении участков повышенной опасности (УПО) на маршруте движения и последующем их обходе. Под УПО мы понимаем компактную область с повышенной концентрацией ДТП. Такой областью может быть кластер, обнаруженный одним из многочисленных алгоритмов кластеризации [3] или целый дорожный сегмент с повышенным числом ДТП.

[1] Оценки показывают, что длина маршрута, вычисленного таким образом, будет в 2-3 раза больше минимальной, что вряд ли приемлемо.

Подходы к кластеризации

В данной работе для кластеризации используется известный алгоритм DBSCAN [4], поскольку он обнаруживает компактные (с заданным максимальным расстоянием между точками) кластеры любой формы. Полученные кластеры будут иметь разный размер (число ДТП в кластере) от заданного минимального до максимального, определяемого самими данными. Очевидно, не все полученные кластеры нужно учитывать, поскольку часть их будет просто случайным шумом, и будь у нас вторая выборка ДТП, кластеризация показала бы, что лишь часть кластеров устойчиво располагается на прежних местах, остальные же меняют свое положение.

Чтобы отсеять кластерный шум, в данной работе используются статистические испытания и проверка статистических гипотез. Иными словами, кластеры, полученные из реальных данных по ДТП, сравниваются с кластерами, полученными с помощью того же числа *равномерно распределенных* по дорожной сети точек. Получив несколько сотен наборов равномерно распределенных точек с числом точек в каждом наборе равном числу реальных ДТП и проведя кластеризацию по каждому такому набору, мы получим материал для проверки статистических гипотез и сможем выделить «статистически достоверные» кластеры [5]).

Использование статистически достоверных кластеров ДТП

Полученные кластеры можно легко интерпретировать за счет того, что при их вычислении задается максимальное расстояние между ДТП равное 10 м. За счет столь небольшого расстояния кластеры очень часто представляют собой компактные «сгустки» ДТП, расположенные на перекрестках, пересечениях дорог, там, где ТС часто перестраиваются из одной линии в другую и т.п. Эти места

требуют изучения и принятия административных мер (понижения скорости, увеличения времени перехода и т.д.), призванных уменьшить там число ДТП.

Повысить безопасность дорожного движения можно и простым обходом выделенных кластеров ДТП. Если, например, модифицировать дорожный граф таким образом, что ребра (дорожные сегменты), ведущие к кластеру, получат дополнительную длину (штраф)[2], алгоритм маршрутизации примет это во внимание и станет по возможности обходить выделенный кластер. Подробности нахождения маршрута, обходящего кластеры ДТП, см. в Главе 3.

Дорожные сегменты как УПО

Обход кластеров, как мы только что выяснили, сводится к обходу выделенных дорожных сегментов. Откуда возникает вопрос: а нельзя ли выделить «опасные» дорожные сегменты без всяких кластеров? Ответ почти очевиден: выделим те ребра дорожного графа (= сегменты дороги), где число ДТП превышает некий порог, определяемый статистическими испытаниями, пометим эти ребра как опасные, т.е. придадим им дополнительную длину (штраф), которая будет учитываться алгоритмом маршрутизации при прокладке маршрута из начальной точки в конечную. Заметим, что такой подход представляется более естественным, поскольку модифицируется непосредственно дорожный граф без привлечения дополнительных объектов (кластеров ДТП). Использование кластеров связано также с объемной ручной работой: необходимо рассмотреть каждый кластер и пометить ребра дорожного графа, ведущие к нему. Автоматизация этой задачи по меньшей мере весьма сложна. Подробности нахождения маршрута, обходящего «опасные» сегменты дороги см. в Главах 4, 5.

[2] Штраф может быть и бесконечным (очень большим числом). Тогда он станет запретительным и алгоритм нахождения маршрута будет вынужден избегать такие грани.

Оценка эффективности маршрутизации

После того как проложен маршрут по модифицированному графу возникает следующая проблема: как оценить эффективность новой маршрутизации, насколько она делает движение безопасней? Для этого, во-первых, нужно каким-то образом создать множество маршрутов, покрывающих весь дорожный граф, и, во-вторых, использовать показатель средней эффективности новой маршрутизации.

В данной работе предложено использовать для создания множества маршрутов квадратную решетку, покрывающую весь дорожный граф. Перебирая все узлы решетки, за исключением одинаковых, получим множество маршрутов типа «вершина, ближайшая к начальному узлу решетки» - «вершина, ближайшая к конечному узлу решетки».

Теперь для каждой пары вершин вычислим *относительный риск ДТП* (ОРДТП) как отношение числа ДТП вдоль измененного маршрута к числу ДТП вдоль оригинального маршрута (т.е. полученного на первоначальном, немодифицированном графе). Если среднее всех таких отношений значимо[3] меньше единицы, то можно сказать, что маршруты, вычисленные на модифицированном графе, в среднем безопасней, чем маршруты, полученные на оригинально графе. Имеет смысл усреднять ОРДТП лишь для определенных длин немодифицированного маршрута, что сделает картину более детальной.

Теоретическая и практическая значимость

Теоретическая значимость работы состоит в создании нового метода маршрутизации транспорта на основе обхода УПО на дороге, определенных с помощью статистических методов по зафиксированным на дорогах ДТП.

[3] То есть верна статистическая гипотеза, что средний ОРДТП < 1

Практическая значимость работы очевидна и состоит в том, что описанные способы маршрутизации должны привести к уменьшению числа ДТП и следовательно — числа человеческих травм и смертей. Обход УПО особенно может быть полезен тем, кто совершает много поездок (водителям такси) или водителям, находящимся в других группах риска (пожилым, людям с пониженной реакцией и пр.). Автором создан пакет программ на языке Python, позволяющий модифицировать дорожный граф любого города таким образом, что маршрут, проложенный по такому графу, будет в среднем безопасней оригинального маршрута[4].

Кроме того, к выявленным УПО могут быть применены административные меры (понижения скорости, увеличения времени перехода и т.д.), призванные уменьшить там число ДТП.

Методология и методы исследования

В работе широко применяются различные статистические методы: вычисление элементарных статистик, получение доверительных интервалов бутстреп-методом, кластеризация, метод статистических испытаний (Монте-Карло). Для оценки результатов работы и для визуализации дорожной сети широко используется геоинформационная система QGIS 3.

[4] Пакет программ рассматривает ребра дорожного как препятствия и подробно описан в Главе 6. Тексты программ приведены в Приложениях 1-10.

Научная специальность

Тематика и содержание данной работы полностью соответствуют паспорту научной специальности 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей», пункт 4 (Интеллектуальные системы машинного обучения, управления базами данных и знаний, инструментальные средства разработки цифровых продуктов).

Работы автора, опубликованные по теме диссертации

1. Герштейн А. М., Терехов А. Н. Выявление участков повышенной опасности на дорогах Массачусетса в 2013-2018 годах // Компьютерные инструменты в образовании. 2021. No 1. С. 45–57.
DOI: http://dx.doi.org/10.32603/2071-2340-2021-1-46-58

2. Герштейн А. М., Терехов А. Н. Маршрутизация транспорта при наличии опасных участков на дороге (на примере города Спрингфилд, Массачусетс) // Компьютерные инструменты в образовании. 2022. No 2. С. 5–18. http://cte.eltech.ru/ojs/index.php/kio/article/view/1729

3. Герштейн А. М., Терехов А. Н. Простой способ повышения безопасности дорожного движения за счет обхода опасных участков маршрута // Программная инженерия. 2023. Том 14, № 3. С. 103—109.
DOI: https://dx.doi.org/10.17587/prin.14.103-109

4. Герштейн, А. М., Терехов, А. Н. Обход опасных участков маршрута как способ повышения безопасности движения (на примере Санкт-Петербурга). // Компьютерные инструменты в образовании. 2023. No 1. С. 30-39
http://cte.eltech.ru/ojs/index.php/kio/article/view/1755

Основные научные результаты

1. В качестве показателя эффективности маршрутизации используется относительный риск ДТП, равный среднему отношению числа ДТП вдоль измененного (с учетом препятствий на дорогах) маршрута к числу ДТП вдоль не измененного (не учитывающего препятствия) маршрута ([57], стр. 8 абзац 6)
2. Статистически значимые кластеры ДТП используются как препятствия при построении альтернативного (более безопасного) маршрута ([48], стр. 47 абзац 6, [57], стр. 8 абзац 7)
3. В качестве препятствий используются отдельные сегменты дороги (ребра дорожного графа), содержащие число ДТП, статистически превышающее число ДТП, полученное из предположения о равномерности распределения ДТП по дорожной сети ([58] стр. 103 абзац 2, стр. 104 абзац 6, [61] стр. 30 абзац 1, стр. 32 абзац 2)
4. Для построения более безопасного маршрута каждое препятствие подвергается штрафу, т.е. к атрибуту длины соответствующего ребра прибавляется фиксированное число. Путем расчетов определяется оптимальное значение штрафа, которое обеспечивает минимальный риск ДТП при минимальном проигрыше в длине маршрута и числе проходимых вершин дорожного графа (([58], стр. 103 абзац 2, [61], стр. 32 абзац 3).

Все статьи по теме диссертации написаны в соавторстве с д.ф-м. наук, профессором А.Н. Тереховым. А.Н. Терехов осуществлял общее руководство работой, автор диссертации предложил использовать средний относительный риск ДТП и соответствующие доверительные интервалы в качестве показателя безопасности маршрута, а также штрафную длину, добавляемую к соответствующему атрибуту грани дорожного графа (= сегменту дороги) в

качестве средства достижения компромисса между длиной маршрута и его безопасностью. Автор также выполнил все расчеты, используя язык Python и библиотеки для работы с графами: OSmnx [6] и NetworkX [7].

Положения, выносимые на защиту

Основные выносимые на защиту результаты работы заключаются в следующем:

1. Показано, что статистически значимые кластеры серьезных ДТП существуют как для пригородов Бостона (Ньютон, Спрингфилд), так и для всего штата Массачусетс. Значимые кластеры определялись путем сравнения результатов работы алгоритма DBSCAN, примененного как к реальным ДТП, так и к симулированным выборкам точек той же численности, но распределенных равномерно вдоль дорожной сети.
2. Примерно 30% значимых кластеров серьезных ДТП, выделенных в Массачусетсе в течение одного года, повторяются в следующем году. Эти стабильные и компактные кластеры размером 10-20м представляют собой участки дорог, на которые следует обратить внимание соответствующим службам для выявления причин, приводящим к возникновению ДТП, и устранения их.
3. В г. Москве путем сравнения числа реальных ДТП с количеством, ожидаемым в соответствии с равномерным распределением по всей дорожной сети (с использованием имитационного моделирования Монте-Карло) были обнаружены дорожные сегменты со статистически значимым высоким числом ДТП.
4. Оптимальное значение «штрафа» для г. Москвы равно 2000м. Это значение приводит в существенному снижению относительного риска

ДТП на 9-31% за счет увеличения средней длины маршрута на 6-11%.

5. Алгоритм, разработанный для г. Москвы (пп. 3-4), был проверен на дорожной сети Санкт-Петербурга. Оптимальным для Санкт-Петербурга следует признать штраф 1000м, приводящий к меньшему увеличению длины пути по сравнению со штрафом 2000м, в то время как относительный риск при этом практически не меняется (средняя длина маршрута возрастает на 8,0-10,0% (1000m) сравнительно с 8.8-14.8% для 2000m). При этом средний относительный риск оказывается в интервале 14,5 – 36% (1000m) по сравнению с интервалом 13,9 – 36% для штрафа 2000м
6. Алгоритм маршрутизации, разработанный для Москвы [58], является устойчивым, то есть, его без каких-либо изменений (за исключением выбора оптимального штрафа) можно применять к дорожной сети других городов.

Подход, примененный к Москве и Санкт-Петербургу, планируется объединить в будущем с данными, получаемыми в реальном времени (данными о пробках) для нахождения альтернативных и безопасных маршрутов.

Глава 1. Обзор литературы

Изобретенные в конце 50-х годов алгоритмы Дейкстры [2], Беллмана-Форда: [8], [9], а также алгоритм A* [21], появившийся в 1968 году, позволяют построить оптимальный в некотором смысле маршрут на взвешенном ориентированном графе[5] G(V,E), где V - множество вершин графа, а E – множество его ребер. При этом первоначально в качестве весов ребер, минимальная сумма которых и достигается в процессе построения маршрута, рассматривались такие простые их свойства как длина или время прохождения, что, казалось, делало эти алгоритмы идеальными при построении маршрутов транспорта между двумя заданными вершинами графа (двумя точками, выбранными на карте). Однако, вскоре выяснилось, что в реальности эти простые свойства нуждаются в корректировке.

Так, например, в реальных условиях существуют пробки, которые делают построение маршрута минимальной длины бессмысленным. Значит, необходимо каким-то образом в режиме реального времени обнаруживать пробки и корректировать веса соответствующих граней дорожного графа (например, сделать пробку полностью «непроходимой», увеличив длину соответствующей грани дорожного графа до бесконечности) – и тогда алгоритм маршрутизации должен будет искать другой маршрут, в той или иной мере свободный от пробок.

С появлением смартфонов и массово доступного Интернета такая задача стала реальной. Достаточно установить на смартфон соответствующее приложение, передающее координаты транспортного средства (ТС) на сервер, чтобы соответствующие алгоритмы определили местоположение ТС (соответствующую грань дорожного графа) и скорость ТС, что и дает возможность выявить координаты и серьезность дорожной пробки. В качестве примера таких приложений можно назвать Yandex, Waze и Google Maps.

[5] То есть графе, ребра которого обладают такими свойствами как направление и вес, "Теория графов. Термины и определения в картинках" https://habr.com/ru/company/otus/blog/568026/

Естественно, приложение, установленное на смартфон, способно определять не только координаты транспортного средства, но и другие особенности его поведения (как часто водитель тормозит-ускоряется, перестраивается из одной линии в другую и пр.) Поэтому появились системы, которые стараются учесть не только пробки, но и вероятность ДТП внутри дорожного сегмента, другие особенности дорог, а также поведение и предпочтения водителя, указанные им самим в мобильном приложении. Так, например, в работе [22] используется взвешенная сумма длины дороги, среднего времени ее прохождения и числа ДТП, зарегистрированных на ней в течение года, а в работе [18] при определении окончательного маршрута некоторые сегменты дороги исключаются из дорожной сети на основе анализа поведения водителя и его предпочтений. По оставшимся сегментам алгоритм маршрутизации определяет окончательный маршрут, используя в качестве атрибута грани комбинацию показателя качества сегмента дороги (road health), среднего времени прохождения и предпочтений водителя. Такой подход в принципе способен повысить безопасность вождения наряду с отысканием маршрута, проходимого за минимальное время. Похожий на [18] подход можно найти в [23], но в последней работе поведение водителя определяется путем обмена данными между ТС.

Другой, быть может, несколько неожиданный пример маршрутизации связан с перевозкой различных опасных грузов. Здесь приоритетной целью маршрута из начальной точки в конечную становится безопасность, поскольку авария при перевозке ядовитого или взрывчатого вещества может привести к экологической катастрофе и человеческим жертвам. В отличие от предыдущего примера, в задаче маршрутизации опасных грузов приоритетным становится не время прохождения маршрута, а безопасность его прохождения. Часто задача может быть решена с помощью уже накопленных данных, а само решение будет зависеть от двух ключевых факторов – вероятности аварии на каждом из дорожных сегментов и плотности населения в районе каждого сегмента [10]. Учет плотности населения там, где находится каждый сегмент дороги, очевидно, будет зависеть от характера

груза, его способности затрагивать в случае аварии большие или меньшие площади. Могут также потребоваться в реальном времени данные о погоде (например, скорости ветра). Более подробный обзор проблемы можно найти в [11], [12], [15].

Третий пример маршрутизации связан с безопасностью пешеходного маршрута. Проблема безопасности может возникнуть при наличии, например, «зон повышенного риска», связанных с криминальной активностью [13]. Здесь задача маршрутизации состоит в обходе опасных участков с повышенной криминальной активностью. Как обычно, алгоритм маршрутизации должен знать значение «опасности» для каждого участка дороги. Чтобы получить это значение из отдельных точек – координат различных преступлений, совершенных в городе, – можно использовать стандартный прием «размытия» или KDE (Kernel Density Estimation, ядерная оценка плотности) [14], который позволяет найти значение риска для любой точки города. В некоторых работах, таких, например, как [28] для определения вероятности преступления для данного места и времени используется теорема Байеса (см. [30], [31]).

Поскольку такого рода маршрутизация производится главным образом для пешеходов, мы сталкиваемся здесь с новой и важной проблемой – слишком большой ценой, которую приходится платить за безопасность. Маршрут, минимизирующий риск, может быть слишком протяженным, поэтому приходится искать компромисс, либо усложняя выражение для атрибута ребра для алгоритма маршрутизации (включая в него сочетание самого риска и длины сегмента дороги), либо получая несколько самых подходящих маршрутов и выбирая из них компромиссный: и более безопасный и не очень протяженный.

Кроме «безопасного» маршрута с минимальной криминальной активностью пешеход может также стремиться выбрать маршрут с максимальной освещенностью, наличием достаточного числа ориентиров, минимальным числом поворотов и максимальной широкими тротуарами [19].

Любопытно, что для велосипедных прогулочных маршрутов в отличие от пешеходных, время прохождения от начального пункта к конечному может не

играть большой роли, поэтому в работе [29] для нахождения оптимального маршрута используются такие свойства дорожного сегмента как наклон, тип дороги, ее ширина, наличие дорожных знаков и освещения. В работе также применен оригинальный алгоритм нахождения оптимального маршрута на основе динамического программирования (о динамическом программировании см., например, [32]).

Перейдем, наконец, к пониманию безопасности, принятому в данной работе. Оно заключается в том, что нужно построить такой маршрут, вдоль которого риск попасть в ДТП будет меньше, чем по маршруту, минимизирующему его протяженность или время прохождения. Для этого известные нам работы используют чаще всего атрибуты ребер, сочетающие длину ребра или время прохождения и (в той или иной форме) – риск ДТП[6].

Так, например, в работе [17] в качестве атрибута грани используется взвешенная сумма длины грани и оценки риска ДТП в пределах этой грани. Заметим, что оценки риска ДТП в этой работе чисто субъективны и сообщаются студентами-информаторами, которые указывают опасные с точки зрения ДТП точки в дорожной сети кампуса вместе с оценкой риска ДТП. Такой подход к выявлению опасных мест дорожной сети можно распространить и на большой город, например, в рамках проекта OpenStreetMap [33].

Другая идея построения маршрута повышенной безопасности связана с получением нескольких альтернативных маршрутов и выбору среди них маршрута приемлемой длины. Например, в работе [16], используется модификация стандартного алгоритма маршрутизации Дейкстры, которая использует в качестве атрибута грани число ДТП, случившихся в каждом сегменте дороги, и дополнительные нормализованные показатели, учитывающие погоду, время суток, возраст водителя и т.п. Модификация алгоритма позволяет получить несколько вариантов маршрута, а затем отобрать маршруты с компромиссным сочетанием риска и длины.

---

[6] Связано это с тем, что построение маршрута, минимизирующего только риск ДТП приведет по нашим оценкам к маршруту слишком большой длины, превышающей оригинальную в 2-3 раза.

Дополнительные к числу ранее зафиксированных ДТП показатели (текущие свойства дорожного сегмента, погода, поведение водителей) используются и в работах [24], [25].

Несколько альтернативных маршрутов рассматриваются и в работе [20], любопытной тем, что в ней делается попытка предсказать вероятность ДТП для пожилых водителей и велосипедистов с помощью свойств трафика (скорости и плотности потока, времени реакции водителя), а также длины дорожного сегмента и коэффициента трения шин и дорожного покрытия.

Попытка предсказать вероятность ДТП при движении по заданному маршруту делается и в работе [26]. Для предсказания вероятности ДТП используется объем трафика, характеристики дороги и текущее состояние погоды. Объем трафика (для тех сегментов дороги, где не фиксируются данные о трафике) получается в результате интерполяции – определения трафика для данного времени по ближайшим сегментам дороги, для которых это значение известно. Вероятность ДТП для данного сегмента определяется с помощью обученного бинарного классификатора (см. [34]), а вероятность вдоль всего маршрута подсчитывается как результат серии испытаний Бернулли[7]. Если полученную вероятность использовать как атрибут грани, то алгоритм маршрутизации определит маршрут, для которого вероятность ДТП в среднем в два раза меньше (по сравнению со стандартным маршрутом, занимающим минимальное время) и в 1.7 раза длиннее. Далее в работе для маршрутизации используется комбинированный атрибут грани, содержащий взвешенную сумму времени прохождения и вероятности ДТП, чтобы найти приемлемый компромисс между длительностью маршрута и его безопасностью.

Вероятность ДТП для каждого дорожного сегмента предлагается оценивать и в работе [27], но уже с помощью совершенно другого математического аппарата – Байесовых сетей [35], использующих комбинацию статических (тип дороги, карта) и динамических (погода, освещение, плотность ТС, сведения о смене дорожных

[7] Испытание Бернулли - это испытание, которое может привести к одному из двух результатов: успеху или неудаче.

полос, скорость ТС и т.п.) данных, получать которые предлагается с помощью самих ТС, так и с датчиков, расположенных вдоль дороги.

Глава 2. Выявление участков повышенной опасности на дорогах Массачусетса в 2013-2018 годах

Введение

Цель данной главы состоит в том, чтобы найти компактные участки повышенной опасности (УПО) на дорогах, т.е. места со статистически достоверной повышенной плотностью серьезных дорожно-транспортных происшествий (ДТП), которые не меняют своего положения в течение нескольких последовательных лет.

Опасность таких участков обусловливается либо дорожной структурой (места пересечения дорог, въезд/выезд на шоссе), либо другими факторами (качеством дорожных покрытий, плохой видимостью и т.д.). Все эти факторы могут быть в известной степени скорректированы муниципальными или государственными органами, более того, эти опасные участки можно исключить из маршрутов коммерческого и частного транспорта.

Обнаружение УПО на земной поверхности является важным шагом в изучении явлений различной природы. Например, места с повышенной плотностью преступлений выявляют наиболее опасные районы в качестве основной цели для полиции и других организаций, чья деятельность направлена на снижение преступности [36, 37].

Существует несколько методов обнаружения УПО: ядерные оценки плотности KDE [2, 4, 14], I-статистика Морана [39], статистика Getis-Ord Gi* [41] а также различные алгоритмы кластеризации. KDE использует различные ядерные функции для преобразования точек на поверхности (например, мест совершения преступлений) в некоторую гладкую функцию в попытке восстановить плотность распределения этих точек. KDE идентифицирует опасные участки на плоскости, но не может оценить их статистическую значимость. Статистики Moran’s I и Getis-Ord Gi* позволяют выявлять статистически значимые области (кластеры). В соответствии с их природой нелегко использовать KDE, getis-ord GI* или

статистику Морана для обнаружения опасных участков заданного размера. Кроме того, статистики Getis-Ord Gi* и Moran's I работают только в 2d-случае [37], а дорожная сеть представляет собой, по существу, одномерный объект с иными представлениями о расстоянии. Заметим, что KDE в принципе можно применить для кластеризации ДТП, принадлежащих дорожной сети, с помощью специальных, довольно сложных ядер, разработанных в [40]. Однако, этот метод требует гораздо более трудоемких расчетов, а результаты не так легко поддаются интерпретации, а форма кластеров в случае применения KDE будет определяться применяемым ядром, в случае же применения методов кластеризации, форма кластеров может быть произвольной.

Что же касается алгоритмов кластеризации, таких как DBSCAN [4], то их легко приспособить к поиску кластеров ДТП, принадлежащих дорожной сети, задавая соответствующую матрицу расстояний, которую можно вычислить с помощью, например, пакета SANET [47]. Кроме того, алгоритм DBSCAN, как это показано в пункте 2.2.1, может легко быть настроен для обнаружения кластеров с заданным максимальным расстоянием между точками.

Существует еще одна сложность в обнаружении УПО, принадлежащих дорожной сети: отсутствие соответствующей статистики. Статистики Getis-Ord Gi* и Moran's I, как уже говорилось, работают только в 2d-случае [37], для методов кластеризации подобные статистики весьма редки. Поэтому для статистического обоснования полученных кластеров в данной работе используются статистические испытания (метод Монте-Карло) [5]. Пакет SANET позволяет относительно просто генерировать миллионы точек, равномерно распределенных по дорожной сети. Таким образом, план данной главы состоит в том, чтобы выполнить кластерный анализ по выбранным ДТП с использованием расстояний по дорожной сети, а затем использовать ту же сеть и то же, что и в реальных данных, количество равномерно распределенных точек для проведения статистических испытаний по меньшей мере несколько сот раз, чтобы получить статистику размеров кластеров. Затем нужно сравнить распределение размеров для реальных и моделируемых кластеров ДТП. В результате будут обнаружены статистически обоснованные кластеры.

2.1 Входные данные

В данном исследовании используются дорожные сети, предоставленные департаментом транспорта Массачусетса [10] в формате Esri shapefile, очень удобном для визуализации с помощью ГИС-приложений, таких как QGIS [44] и OpenJUMP [45]. Были также использованы данные портала MassDOT [46] о ДТП в штате Массачусетс за 2013-2018 годы в формате электронной таблицы .csv, который легко преобразуется в другие форматы, например, в тот же Esri shapefile. Как обычно, полученные данные должны быть предварительно обработаны. Изолированные фрагменты дорожной сети (если они есть) должны быть соединены с основной сетью. Для связи фрагментов с основной сетью можно использовать плагин **Disconnected Islands** и инструменты редактирования QGIS. В файлах, содержащих сведения о ДТП, нужно оставить только те записи, где присутствуют координаты ДТП (примерно 96% всех записей).

Поскольку сведения о ДТП и файлы, в которых хранится дорожная сеть, получены нами из разных источников, отдельные ДТП не принадлежат в точности элементам сети. Между тем для некоторых видов анализа важно, чтобы ДТП и дорожная сеть были единым целым. Поэтому точки ДТП должны быть спроецированы на элементы сети, для чего мы использовали плагин QGIS NNJoin, который создает дополнительный слой в QGIS, где сами координаты ДТП не меняются, но появляются дополнительные атрибуты, хранящие начальные и конечные координаты ближайшей к ДТП прямой линии, принадлежащей сети. Эти координаты позволяют спроектировать точку ДТП на соответствующую линию дорожной сети с помощью простого Python-скрипта:

```
def dropPoint2Line (point, line):
    #point (x,y)
    #line  (point1,point2)
    x0 = point[0]
    y0 = point[1]

    x2 = line[0][0]
    y2 = line[0][1]
    x3 = line[1][0]
    y3 = line[1][1]

    m = x3 - x2
    p = y3 - y2
    t = (m * x0 + p * y0 - m * x2 - p * y2)/(m**2 + p**2)
    x1 = m * t + x2
    y1 = p * t + y2
    return (x1,y1)
```

2.2 Методы исследования

2.2.1 DBSCAN

В качестве алгоритма кластеризации выберем DBSCAN (Density-Based Spatial Clustering of Applications with Noise) — эвристический алгоритм, находящий кластеры одинаковой минимальной плотности (буквы DB в названии алгоритма означают Dense-Based, т.е. ориентированный на плотность) и произвольной формы. При этом количество кластеров определяется автоматически. Каждая точка после работы алгоритма оказывается либо принадлежащей какому-то кластеру, либо выбросом — точкой, не попавшей ни в один кластер. Преимущество DBSCAN еще и в том, что этот алгоритм, реализованный в библиотеке sklearn, работает очень быстро, что важно для обработки данных большого объема и проведения статистических испытаний, когда алгоритм приходится использовать сотни раз.

DBSCAN использует матрицу расстояний между точками или сами координаты (в случае евклидовых расстояний), а также два параметра: eps (расстояние) и min_samples (натуральное число). Алгоритм сначала выделяет *основные* точки —

те, что в eps-окрестности содержат не менее min_samples других точек[8]. Первая основная точка, выделенная алгоритмом, становится «затравкой» кластера. Обозначим ее Q1-0. Кластеру приписываются все точки, *достижимые* из Q1. Это могут быть точки, достижимые непосредственно, то есть, находящиеся в eps-окрестности Q1-0, или точки, для которых существует путь Q1-0 -> Q1-1 -> Q1-2 …, причем Q1-1 находится в eps-окрестности Q1-0, а точка Q1-2 – в eps-окрестности Q1-1. Заметим, что все точки такого пути, кроме последней, должны быть основными. В кластер могут попасть как основные точки, так и обычные, для которых существует путь из Q1-0, но которые не содержат min_samples в своей eps-окрестности.

Когда кластер сформирован, алгоритм попытается найти основную точку, не принадлежащую уже выделенному кластеру и сформировать новый кластер вокруг нее. Алгоритм завершит работу, когда не сможет найти основную точку, не принадлежащую уже выделенным кластерам.

2.2.2 Матрица внутрисетевых расстояний (масштаб 20м)

Подготовив данные, попробуем обнаружить зоны повышенной опасности в городе Ньютон, штат Массачусетс, используя данные о серьезных ДТП за 2013 год (всего 369 случаев). В нашей первой попытке кластеризации будем использовать матрицу сетевых расстояний, вычисленную с помощью пакета SANET, и параметры eps=20м и min_samples = 3 алгоритма DBSCAN. Выбор параметра eps=20м представляется разумным, поскольку 20м — это расстояние порядка ширины дороги, состоящей из пяти полос шириной 3,6м и полученные кластеры, видимо, будут достаточно компактны, чтобы их можно было легко избежать, если использовать подходящий алгоритм маршрутизации. Но алгоритм DBSCAN устроен так, что eps=20м задает eps-окрестность точки, то есть, минимальную

[8] При этом сама точка, претендующая быть основной, тоже считается. Если, например, eps=3, то в eps-окрестности основной точки должно быть не менее 2-х других точек.

плотность, а не размер кластера. Сами кластеры могут иметь произвольную форму и быть вытянуты на расстояние, многократно превышающее 20м.

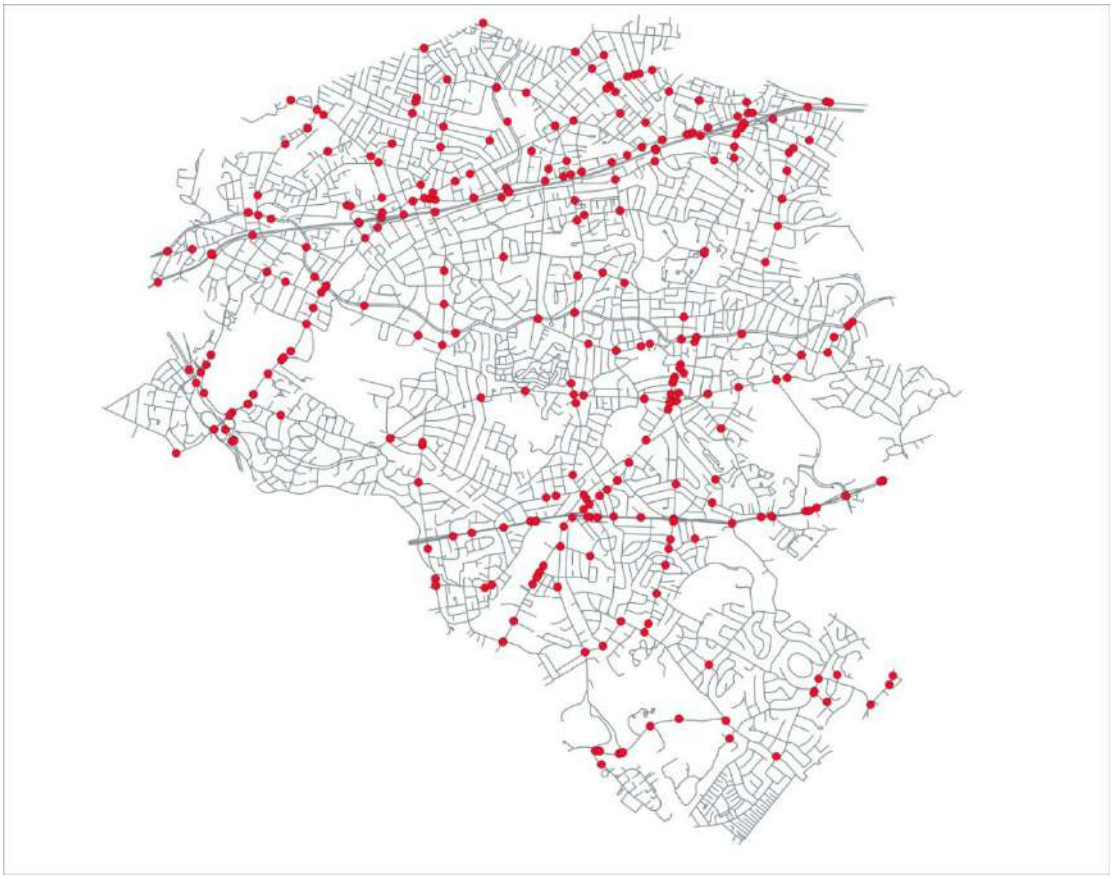

Рисунок 2-1. Серьезные ДТП (Newton MA, 2013)

В результате расчетов были получены 63 кластера размерами от 3 до 9 ДТП. Для выделения из этих кластеров статистически значимых сформулируем нулевую гипотезу: *ДТП распределены равномерно по дорожной сети*. Будем считать, что кластер размером **n** статистически значим на уровне **a**, если вероятность обнаружения хотя бы одного кластера размера, большего или равного **n** (в случае, если нулевая гипотеза верна), меньше **a** [5]. Учитывая, что статистики размеров кластеров в случае алгоритма DBSCAN не существует, мы провели ряд статистических испытаний методом Монте-Карло. С помощью пакета SANET были получены 1024 выборки равномерно распределенных по дорожной сети Ньютона точек (каждая выборка содержит 369 точек, как и в исходных данных), затем эти выборки были использованы для вычисления 1024 матриц расстояний по дорожной сети (1024 – число испытаний Монте-Карло) и далее к каждой матрице

был применен алгоритм DBSCAN с теми же параметрами, как и в случае реальных данных (eps=20 и min_samples=3). Результаты моделирования приведены в Таблице 2-1.

Таблица 2-1. Результаты моделирования (1024 испытания, дорожная сеть Ньютона, штат Массачусетс)

Размер	Количество кластеров одинакового или большего размера	P
3	59	0.058
4	0	0

Из Таблицы 2-1 видно, что на уровне 0,05 значимы кластеры с n >= 4. Удаление кластеров с n=3 из 63 первоначально обнаруженных дает 5 статистически достоверных кластеров.

2.2.3 Гибридный подход к кластеризации

Кластеризация, использующая матрицу сетевых расстояний, представленная в предыдущем разделе, требует очень больших вычислительных ресурсов и возможна (с учетом необходимости провести по крайней мере несколько сотен испытаний) лишь для небольших наборов данных, таких как данные о серьезных ДТП в Ньютоне, пригороде Бостона с населением около 90 тыс .

Поэтому очень заманчиво выглядит замена сетевых расстояний между отдельными ДТП на соответствующие евклидовы. В этом случае алгоритм DBSCAN, реализованный в пакете sklearn (Python), вычисляет матрицу расстояний чрезвычайно быстро, и для масштаба 20м кажется разумным, что расстояния (евклидовы и по сети) очень похожи. Повторив все вычисления — на этот раз с евклидовыми расстояниями и получив новый набор статистически значимых кластеров, можно сравнить их с кластерами, полученными в разделе 2.2.2.

Сравнивая два набора кластеров, получим следующее:

- 5 сетевых кластеров расположены идентично своим евклидовым собратьям;
- 2 евклидовых кластера расположены иначе;
- идентичные кластеры могут сильно отличаться при увеличении масштаба.

Итак, рассматривая сетевую кластеризацию как «истинную», мы можем перечислить некоторые недостатки евклидовой кластеризации:

1. Ложные кластеры — не имеют сетевого аналога
2. Испорченные кластеры — имеют истинные (те же, что в сетевом случае) и ложные фрагменты (на разных дорожных линиях или на разных дорогах).

Теперь можно суммировать два подхода к кластеризации с eps=20 в Таблице 2-2.

Таблица 2-2. Ньютон 2013. Два метода кластеризации (евклидов и сетевой, eps=20).

Метод	**N**	**Процент от общего числа серьезных ДТП**	**Ложные кластеры число/процент**	**Испорченные кластеры число/процент**
Сеть	5	7.3	0/(0%)	0/(0%)
Евклидов	7	10	2/(29%)	3/(43%)

2.2.4 Масштаб 10м

Из предыдущего раздела (Таблица 2-2) следует, что сетевой и эвклидов подход к кластеризации при eps=20м существенно отличаются.

Между тем очевидно, что при некотором достаточно малом масштабе сетевое и евклидово расстояние должны быть идентичны. Конечно, этот масштаб меньше 20 м. Чтобы оценить этот максимальный масштаб, исследуем зависимость разности

между евклидовым и сетевым расстояниями от, скажем, евклидовых расстояний и найдем точку расхождения, где евклидовы расстояния становятся неадекватны нашей задаче (Рис. 2-2).

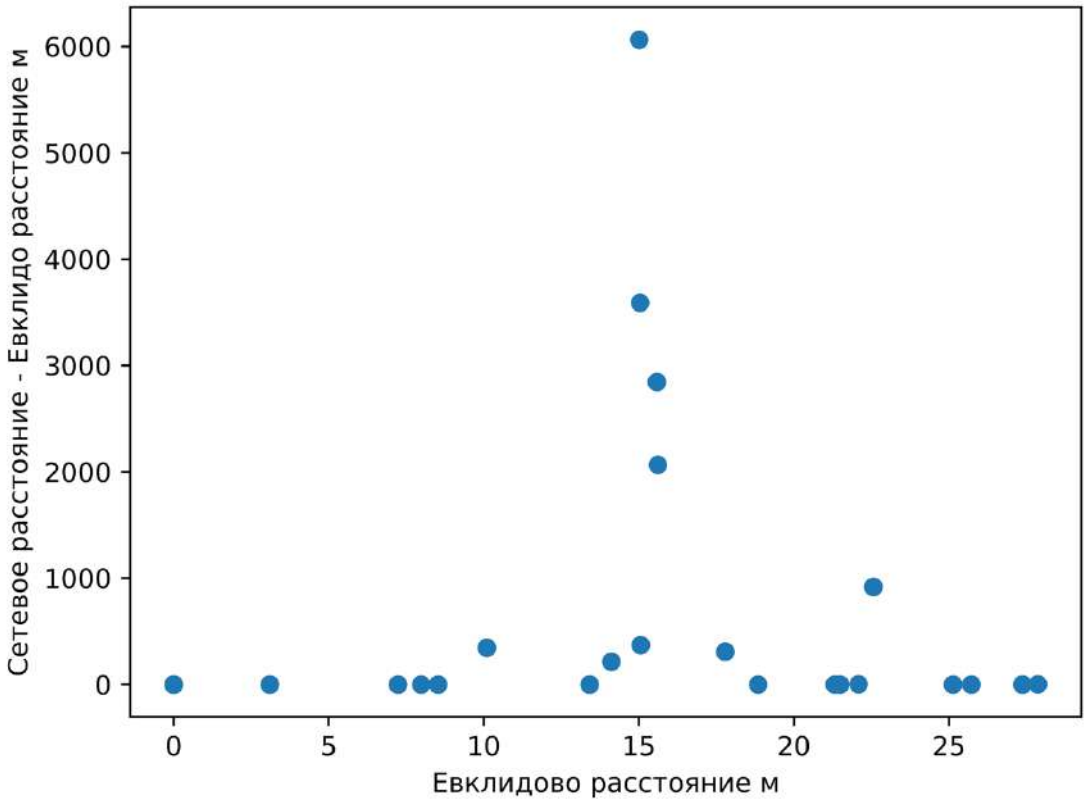

Рисунок 2-2. Ньютон, штат Массачусетс, 2013 год, серьезные ДТП. Зависимость разницы между евклидовыми и сетевым расстояниями от евклидовых расстояний

Как видно из Рис. 2-2, сетевое расстояние всегда не меньше евклидова, потому что прямая линия, соединяющая две точки, имеет минимальную длину в евклидовом мире. Также видно, что расхождение между евклидовым и сетевым расстояниями начинается с масштаба 10 м. На графике хорошо заметны большие всплески вблизи 15 м, и это, скорее всего, искажает евклидову кластеризацию с eps = 20. Таким образом, мы можем выбрать 10м в качестве масштаба, где евклидово расстояние, требующее гораздо меньших вычислений, может заменить расстояние по дорожной сети.

Значение 10м кажется очень маленьким, но на самом деле это не так. Оно лишь немного меньше ширины 3-полосного шоссе (в Соединенных Штатах каждая полоса имеет ширину примерно 12 футов или 3,6 м. Иными словами, 10 метров — это естественный масштаб для движения транспортных средств в одном направлении по дорожной сети.

2.2.5 Гибридная кластеризация

Глядя на Рис. 2-2 и принимая во внимание соображения о натуральном для дорожной сети масштабе, приведенные в конце предыдущего раздела, можно обосновать гибридный метод кластеризации: *использовать дорожную сеть только для генерации наборов равномерно распределенных точек и выполнять всю кластеризацию (для обнаружения реальных кластеров и для статических испытаний) с eps=10, используя евклидовы расстояния между ДТП.*

2.3 Содержательный пример

2.3.1 Массачусетс 2013, кластеризация серьезных ДТП

Согласно имеющимся данным, в штате Массачусетс в 2013 году зарегистрировано 30696 серьезных ДТП, из них 23964 (78%) на крупных дорогах. Применение кластеризации с eps=10 и min_samples=3 к 23964 серьезным ДТП дает 1340 кластеров, из которых нужно выделить статистически значимые. Для этого кластеризация была выполнена 1502 раза на наборах из 23964 равномерно распределенных по дорожной сети точек, сгенерированных программным пакетом SANET. Результаты испытаний методом Монте-Карло приведены в Таблице 2-3. Таким образом, на уровне 0,05 следует рассматривать кластеры размером >=5 как статистически значимые. Удаляя из первоначальных кластеров те, чей размер

меньше 5, получим 354 кластера, общее число серьезных ДТП в кластерах составляет 2301, то есть 9,6% от общего числа серьезных ДТП.

На Рис. 2-3 показаны статистически значимые кластеры серьезных ДТП, случившихся в Массачусетсе в 2013 году.

Таблица 2-3. Результаты моделирования методом Монте-Карло для Массачусетса, 2013 год (1502 испытания, 23964 точки в каждом)

Размер	**Количество кластеров одинакового или большего размера**	**P**
3	1502	1
4	160	0.1
5	6	0.004

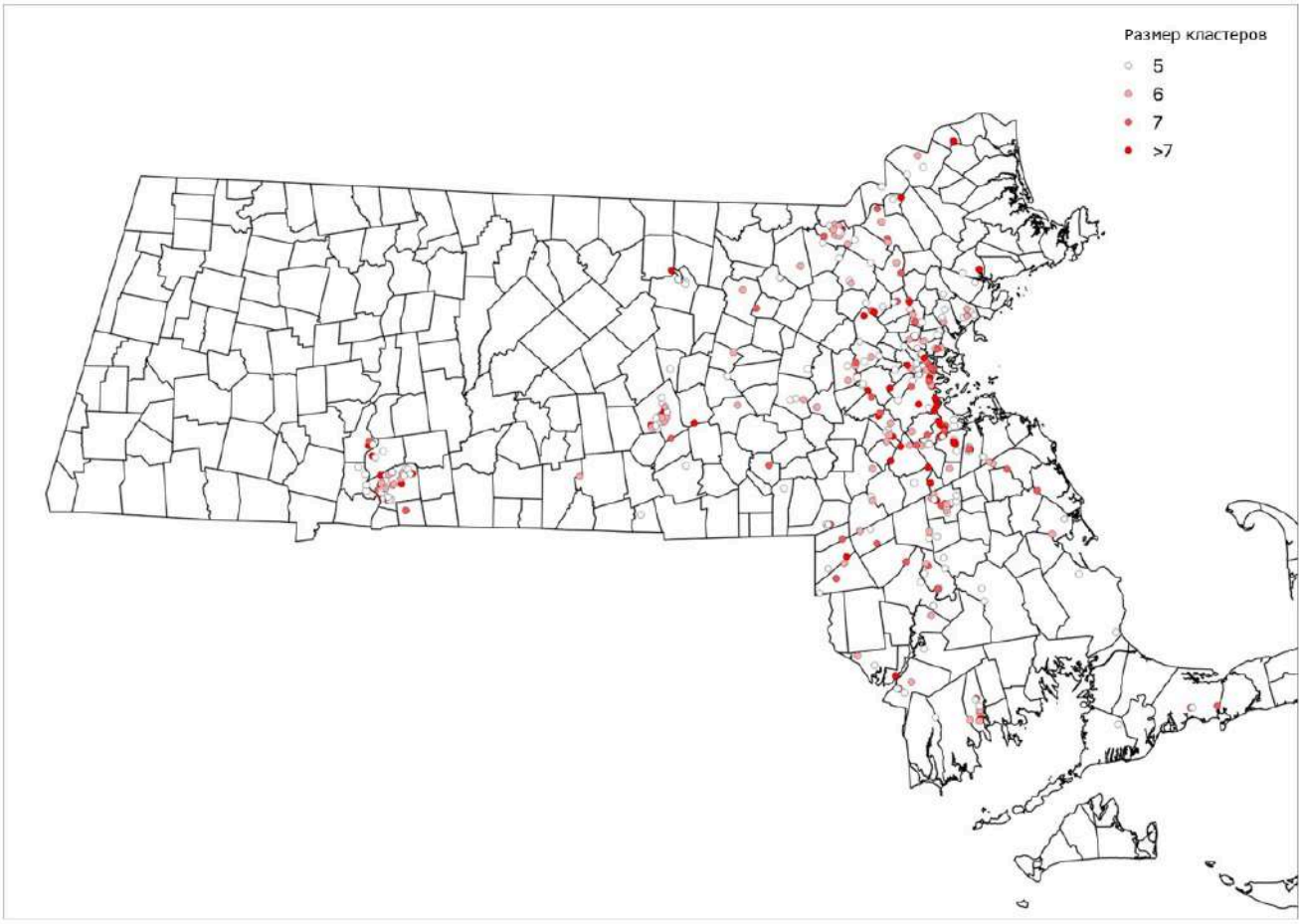

Рисунок 2-3. Массачусетс 2013, значимые кластеры серьезных ДТП (DBSCAN, eps=10)

Как видно из Рис. 2-3, пространственное распределение кластеров по всему штату далеко не однородно. В тринадцати городах найдены 166 (47%) кластеров (Таблица 2-4). Ньютон с 3 кластерами занимает 38-е место в общем списке и включен в таблицу только потому, что мы подробно изучали его в Разделе 2.

Таблица 2-4. Города Массачусетса с максимальным количеством кластеров

Город	Кластеры	Число ДТП
SPRINGFIELD	38	234
BOSTON	24	209
WORCESTER	23	148
LOWELL	15	93
BROCKTON	14	87
BRAINTREE	8	68
NEW BEDFORD	9	55
QUINCY	7	53
WOBURN	5	39
WEYMOUTH	5	36
HOLYOKE	5	35
RAYNHAM	7	35
WALTHAM	6	35
NEWTON	3	19

Из Рис. 2-3 также видно, что большинство кластеров имеют размеры от 5 до 7.

2.3.2 Мета-кластеры в Массачусетсе. Гибридный подход снизу вверх

В предыдущем разделе были проанализированы кластеры (УПО) серьезных дорожно-транспортных происшествий, случившихся в Массачусетсе в 2013 году. Поставим себе следующую цель — использовать данные за 2013-2018 годы для

выявления стабильных во времени кластеров, то есть мест, которые остаются опасными как минимум несколько лет. Будем использовать двухэтапную кластеризацию, то есть получение значимых кластеров за каждый год, а затем кластеризации этих кластеров еще раз с параметром min_samples = <минимальное число лет>. Например, min_samples = 4 означает, что результирующий кластер содержит данные по меньшей мере за 4 разных года.

Как и в разделе 2.3.1, получим сначала все (в том числе статистически незначимые) кластеры за 6 лет: 2013, 2014, 2015, 2016, 2017, 2018. Затем проведем испытания методом Монте-Карло, чтобы выделить значимые кластеры. Результаты испытаний методом Монте-Карло за все 6 лет собраны в Таблице 2-5.

Таблица 2-5. Результаты испытаний методом Монте-Карло за шесть лет, DBSCAN (eps=10, min_samples=3) равномерно распределенные по дорожной сети случайные точки

Год	**Всего ДТП**	**Количество испытаний**	**Размер кластера**	**Количество кластеров данного или большего размера**	**P**
2013	23964	1502	3	1502	1.0
			4	160	0.11
			5	6	0.004
2014	24921	1444	3	1442	0.99
			4	171	0.12
			5	6	0.004
2015	25959	1386	3	1384	0.99
			4	208	0.15
			5	11	0.008

Продолжение Таблицы 2.5

Год	**Всего ДТП**	**Количество испытаний**	**Размер кластера**	**Количество кластеров данного или большего размера**	**P**
2016	26772	1344	3	1344	1.0
			4	214	0.16
			5	8	0.006
			6	1	0.0007
2017	26672	1349	3	1349	1.0
			4	206	0.15
			5	9	0.007
2018	24763	1453	3	1453	1.0
			4	201	0.14
			5	7	0.005

После удаления статистически незначимых кластеров объединим кластеры за все 6 лет, добавив метку года к каждому кластеру, а затем выполним вторую кластеризацию с eps=10 и параметром min_samples=3. Некоторые статистические характеристики метакластеров показаны в Таблице 2-6. Количество ДТП во всех метакластерах – 7456 составляет 4,8% от 153051 – общего количества ДТП, используемых для двухступенчатого кластеризации. На Рис. 2-4 показано распределение кластеров на карте Массачусетса. Очевидно, пространственное распределение метакластеров аналогично 2013 году.

Таблица 2-6. Статистика метакластеров

Разные годы в кластере	Кластеров	ДТП	ДТП / кластер	Медиана ДТП	Стандартное отклонение ДТП
3	115	2269	19.7	19	3.29
4	74	1979	26.7	25	5.77
5	36	1412	39.2	36	10.68
6	29	1796	61.9	56	21.71
Всего	254	7456	29.3	24	16.23

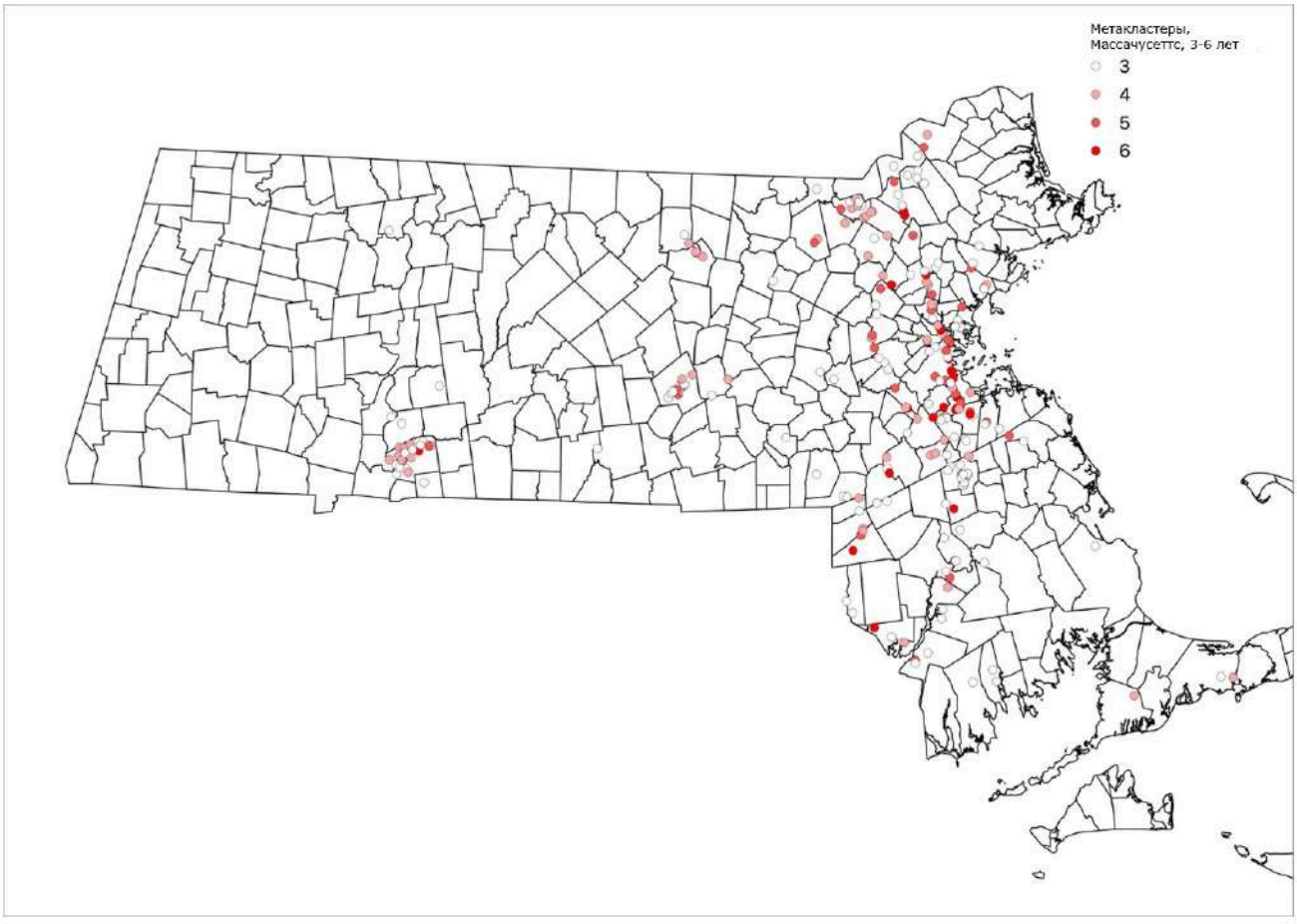

Рисунок 2-4. Мета-кластеры в Массачусетсе, 3-6 разных лет в каждом кластере

Кроме статистических характеристик метакластеров, содержащих различное число лет, когда произошли ДТП, весьма интересны характеристики повторяемости кластеров из года в год. Можно, например, для каждого кластера

ДТП, случившихся в одном году, найти (если он существует) пространственно эквивалентный кластер ДТП, случившихся в другом году. Результаты представлены в Таблице 2-7, которая напоминает матрицу расстояний и содержит относительное количество кластеров, наблюдавшихся в течение одного года и оставшихся в последующие годы. Например, 0,36 на пересечении строки 2018 и столбца 2017 означает, что отношение (количество кластеров в 2017 году, найденных в 2018 году/общее количество кластеров в 2017 году) равно 0,36. Общее количество значимых кластеров в соответствующем году показано в колонке Кластеры.

Таблица 2-7. Относительная повторяемость кластеров из года в год

Годы	2017	2016	2015	2014	2013	Кластеры
2018	0.36	0.35	0.35	0.33	0.36	378
2017		0.35	0.35	0.34	0.32	369
2016			0.39	0.33	0.3	374
2015				0.3	0.33	334
2014					0.36	349
2013						354

Таблица 2-8. Повторяемость кластеров для различных временных интервалов

1 год: 13-14, 14-15, 15-16, 16-17,17-18 **2 года**: 13-15, 15-17, 14-16, 16-18 **3 года**: 13-16, 14-17, 15-18	**1 год**	**2 года**	**3 года**
	0.36	0.33	0.3
	0.3	0.33	0.34
	0.39	0.35	0.35
	0.35	0.35	
	0.36		
Среднее	0.35	0.34	0.33
Стандартное отклонение	0.03	0.01	0.02

Интересно также выяснить, как повторяемость кластеров зависит от годового интервала. В Таблице 2-8 данные Таблицы 2-7 представлены несколько иначе. Столбец **1 год** содержит «расстояния» в один год между наборами кластеров (т. е. повторяемость кластеров за 2013-2014, 2014-2015 и так далее), столбец **2 года** представляет данные о двухлетней повторяемости (т. е. 2013-2015, 2014-2016 и так далее). Последняя колонка (3 года) представляет данные для трехлетней повторяемости. Последние две строки таблицы содержат элементарную статистику повторяемости для разных годичных интервалов. Видно, что «расстояние» между наборами кластеров остается относительно постоянным, незначительно уменьшаясь с увеличением годового интервала. По-видимому, дорожные условия для исследуемых ДТП существенно не меняются из года в год, т.е. относятся к одному распределению и могут быть объединены при необходимости для дальнейшего анализа.

Выводы

В данной работе показано, что гибридный метод кластеризации DBSCAN с использованием евклидова расстояния, применяемый к данным о серьезных[9] ДТП в штате Массачусетс, в сочетании с моделированием методом Монте-Карло на дорожной сети штата, способен идентифицировать статистически значимые УПО серьезных ДТП. Эти участки (кластеры) компактны, причем примерно треть УПО повторяется в следующем году. Выявленные участки повышенной опасности серьезных ДТП желательно избегать водителям, а городским и государственные органам необходимо использовать информацию о таких участках для планирования мероприятий, имеющих целью снижение травматизма и смертности на дорогах.

[9] То есть, приведших к травмам или смерти хотя бы одного участника ДТП

Дальнейшие исследования могут быть полезны для выявления пространственных и иных особенностей УПО (напр., перекресток, въезд/выезд на шоссе, соединение дорог, ширина дороги, ограничение скорости, плохая видимость, время дня, погодные условия и т.д.).

Глава 3. Маршрутизация транспорта при наличии опасных участков на дороге (на примере города Спрингфилд, Массачусетс)

Введение

Задача построения оптимального в том или ином смысле маршрута для индивидуального транспорта относится к числу хорошо изученных и давно решенных. Если дорожная сеть представлена двунаправленным мультиграфом (то есть графом, где через две любые вершины могут проходить несколько граней, а каждая грань может быть одно- или двунаправленной), то маршрутизация проводится с помощью алгоритма Дейкстры [2] или Беллмана-Форда [8, 9]. Каждой грани достаточно сопоставить число (для алгоритма Дейкстры – обязательно неотрицательное) и алгоритмы построят оптимальный (в смысле минимизации суммы таких чисел) маршрут.

Для обычных задач маршрутизации грани приписывается либо ее длина, либо время задержки — гораздо более сложный параметр, зависящий не только от геометрии дорожной сети, но и от текущего состояния трассы: наличия дорожных пробок, погодных условий и пр.

К сожалению, в обычной маршрутизации весьма редко учитываются другие важные факторы, в частности, участки дороги с компактными участками повышенной опасности УПО (см. [48]) или с большим количеством серьезных (т.е. приведших к травмам или летальным исходам) ДТП, находящихся в пределах одного дорожного сегмента (грани). Есть работа [49] (см. также обзор литературы в Главе 1), где сделана попытка создания интегрированного признака, приписываемого каждой грани дорожного графа и содержащего некую взвешенную сумму времени задержки и показателя опасности ДТП для каждой грани. Такой подход весьма трудоемок и требует отдельных и весьма непростых расчетов для каждой грани графа, представляющего дорожную сеть. Между тем,

существует простой прием, позволяющий обойти опасные участки, помечая их как «непроходимые».

Представим себе граф дорожной сети, в котором каждой грани приписана в качестве атрибута, используемого алгоритмом маршрутизации, ее длина. Тогда граням, ведущим к участку повышенной опасности (УПО), можно приписать бесконечную или, что более практично, очень большую (на несколько порядков больше длины самой протяженной грани) длину и тогда алгоритм маршрутизации автоматически обойдет опасный участок дороги.

3.1 Инструменты и данные

Применим этот простой прием для дорожной сети города Спрингфилд (Springfield, MA), извлеченной из глобальной карты земной поверхности OpenStreetMap (OSM)[10]. Карты OSM, полученные в результате многолетней работы некоммерческого вики-проекта, могут быть разного качества в зависимости от местности[11], но для целей исследования алгоритмов маршрутизации (а как показано в работе [50] на примере дорожной сети Вены, также и для реальной маршрутизации) они вполне подходят.

Единственная трудность состоит в извлечении дорожной сети для заданного участка поверхности. Когда для этого использовался OSM API, задача была далека от тривиальной. К счастью, недавно появившаяся библиотека OSMnx для языка Питон (см. подробности в [51]) упростила эту задачу до пары-другой строчек кода. Так, для получения карты любого города, достаточно указать сам город, штат (для США) и страну.

[10] См. https://www.openstreetmap.org/

[11] Существуют многочисленные средства проверки актуальности и точности OSM-карт (см. https://wiki.openstreetmap.org/wiki/Quality_assurance). Так, например, сайт https://is-osm-uptodate.frafra.eu/ позволяет посмотреть последние изменения карт и их прошлые редакции. Карта Спрингфилда согласно этому сайту создана в 2014 году, последние изменения датируются 2021 годом, что говорит об ее актуальности.

Например, дорожная сеть Спрингфилда загружается с помощью функции graph_from_place() пакета OSMnx:

```
import osmnx as ox
ox.config(use_cache=True, log_console=True, all_oneway=False)
G = ox.graph_from_place('SPRINGFIELD, MA, USA',
network_type='drive')
```

Остается только сохранить объект G, содержащий дорожную сеть, на диске для дальнейшего использования.

Объект G хранит дорожную сеть в виде МультиДиГрафа (MultiDiGraph), то есть графа с возможностью существования многих одно- и двунаправленных граней, проходящих через одну и ту же пару вершин. Такой формат графа не всегда удобен для геоинформационных систем (ГИС), подобных QGIS или ArcGIS, поэтому в пакете OSMnx существует возможность сохранения в векторных .shp файлах, стандартных для ГИС, отдельно граней и вершин МультиДиГрафа, которой мы будем широко пользоваться для определения граней, ведущих к УПО.

В Таблице 3-1 представлены некоторые статистические характеристики граней дорожного графа Спригфилда.

Таблица 3-1 Элементарная статистика длин граней дорожного графа Спрингфилда

Среднее м	130
Медиана м	92
Ст. отклонение м	116
Минимум м	1.47
Максимум м	2084,8

Нам также понадобятся данные о ДТП в штате Массачусетс за 2013-2018 годы, которые можно получить в текстовом формате .csv на сайте портала MassDOT[12].

[12] https://massdot-impact-crashes-vhb.opendata.arcgis.com/search

Рисунок 3-1. Сетка из равноотстоящих точек, наложенная на карту Спрингфилда

3.2 Метод

Для исследования эффективности маршрутизации при наличии опасных участков дорожной сети, зададим сетку равноотстоящих точек и наложим ее на карту Спрингфилда (Рис. 3-1). На карте поместилось 95 узлов сетки с расстоянием около 830 м. между соседними узлами, что дает 95*94=8930 различных направленных маршрутов[13] . Наши дальнейшие действия можно уложить в следующую схему:

а/ Объединим все случившиеся в Спрингфилде в 2013-2018 годах серьезные ДТП и дорожный граф, извлеченный из OSM. Заметим, что координаты ДТП и дорожный граф получены из независимых источников, поэтому точки ДТП могут не совпадать с гранями графа. Следовательно, нужна процедура, которая проецирует каждое ДТП на ближайшую

[13] Здесь учитывается, что некоторые грани дорожного графа могут быть однонаправленными, так что маршрут n->m не идентичен маршруту m->n.

грань. В результате этой операции каждой грани графа будет сопоставлено неотрицательное число ДТП, случившихся на ее протяжении.

б/ Подготовим две версии дорожного графа: G0 и G1. Версия G0 будет содержать исходный граф, извлеченный из OSM. В версии G1 атрибут, хранящий длину грани, будет для некоторых граней заменен очень большим числом , на много порядков превышающем максимальную длину грани, что делает такие грани практически непроходимыми (при наличии альтернатив) для алгоритма маршрутизации.

в/ Пользуясь сеткой, наложенной на карту Спрингфилда, построим все возможные маршруты для графов G0 и G1. Далее выберем из всех маршрутов, соединяющих одни и те же точки в том же направлении, только отличающиеся для G0 и G1, то есть те, которые затронуты изменениями в графе G1.

г/ Для всех выбранных направленных маршрутов сравним их длину и число вершин (по данным, хранящимся в графе G0), то есть, посмотрим, как меняется длина и «дробность» маршрута при обходе препятствий.

д/ Для всех выбранных направленных маршрутов сравним числа ДТП вдоль маршрута, то есть посмотрим, как меняется относительный риск ДТП при обходе препятствий. Под относительным риском здесь понимается отношение ДТП1/ДТП0, где ДТП0 – число ДТП, подсчитанное вдоль оригинального маршрута на графе G0, а ДТП1 – число ДТП вдоль маршрута, соединяющего те же начальную и конечную точку, но на графе G1.

3.3 Обход участков повышенной опасности (УПО)

Попробуем строить маршруты транспортных средств так, чтобы обойти участки повышенной опасности (УПО) или статистически значимые метакластеры ДТП, то есть кластеры, полученные в результате вторичной кластеризации кластеров ДТП, случившихся в Спрингфилде в 2013-2018 годах. Из всех таких вторичных кластеров были отобраны только те, что содержат кластеры ДТП не менее чем за три различных года (всего 31). См. подробности в [48].

Наша задача достаточно сложна, поэтому разумно будет разбить ее на несколько этапов, сохраняя (там, где это необходимо) промежуточные данные в файлах формата .csv:

1. Загрузить с диска два оригинальных дорожных графа и назвать их G0 и G1.
2. Найти все грани, ведущие к УПО (метакластерам). Эту задачу приходится выполнять вручную, поскольку кластер может быть по-разному расположен. Если кластер ДТП расположен на однонаправленной грани, достаточно просто присвоить атрибуту, хранящему длину грани, очень большое число и грань станет практически непроходимой для алгоритмов маршрутизации, то есть кластер (УПО) будет исключен из прокладываемого по графу G1 маршрута.

   Если же кластер находится на двунаправленной грани, задача модификации графа усложняется, поскольку нужно будет приписать кластер одному из направлений грани[14]. Наконец, в случае кластера, расположенного на перекрестке, приходится «закрывать» все пути, ведущие к кластеру, потому что нет какой-то выделенной грани и ДТП происходят по всем направлениям. Таким образом, граф G1 легко можно изменить, если заранее подготовить

[14] Для небольшого числа ДТП, составляющих кластер, это возможно, если использовать дополнительную информацию: адрес, направление движения. Если же кластер не удается приписать определенному направлению, приходится «запрещать» оба.

список «запретных» граней, рассматривая отдельно каждый метакластер и все грани, ведущие к нему.

3. Для всех граней графа G1, определенных в предыдущем пункте, заменим значение атрибута длины грани очень большим числом 100000000000, на много порядков большим, чем длина самой длинной грани в G1[15].

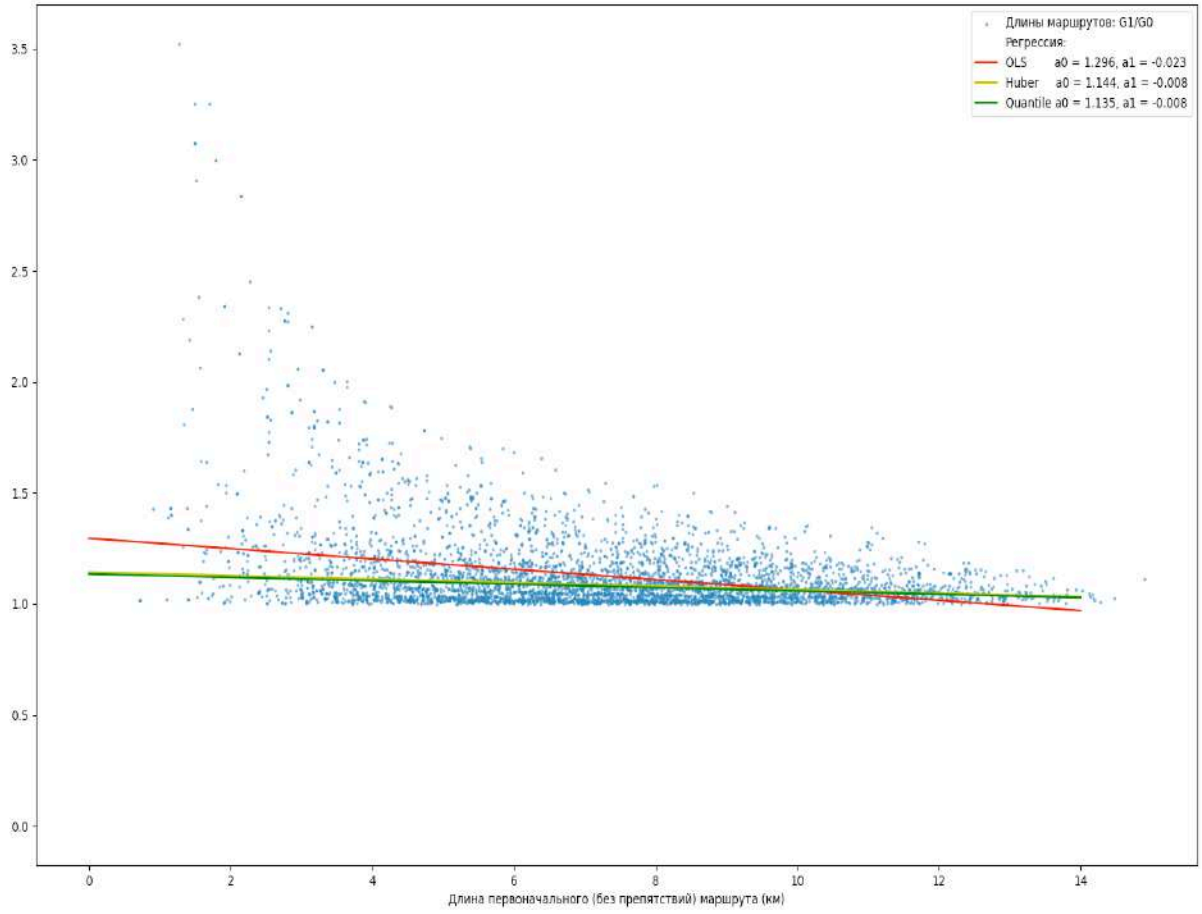

Рисунок 3-2. Отношения длин маршрутов, вычисленных по графам G1 (с учетом препятствий) и G0 (без учета препятствий).

4. Для всех начальных и конечных пунктов, задаваемых сеткой из равноотстоящих точек (см. Рис. 3-1) вычислим оптимальные маршруты и выделим из них те, которые различаются для графов G0 и G1 при

[15] Максимальная длина грани согласно таблице 3-1 равна 2085 метров, так что число 100000000000, превышающее максимальную длину в 48 миллионов раз, можно с практической точки зрения считать бесконечным.

одинаковых начальной и конечной точках маршрутов. В результате получаются 4822 таких маршрута, причем длина некоторых маршрутов (всего их 91, и это составляет менее 2% от всех полученных) превышает 100000000000, что говорит о том, что в этих случаях алгоритм маршрутизации не нашел путей обхода препятствий. Исключив такие маршруты из наших данных, получим окончательно 4731 пары маршрутов на графах G0 и G1, имеющих одинаковые начало и конец маршрута.

Эти оставшиеся маршруты могут быть отображены в виде точек на графике, где по оси X отложены первоначальные длины маршрутов без учета препятствий, а по оси Y – отношения длин маршрутов: вычисленных по графам G1 и G0 (Рис. 3-2) и отношения числа вершин заданного маршрута, вычисленных по графу G1 к числу вершин, вычисленных по графу G0 (Рис. 3-3).

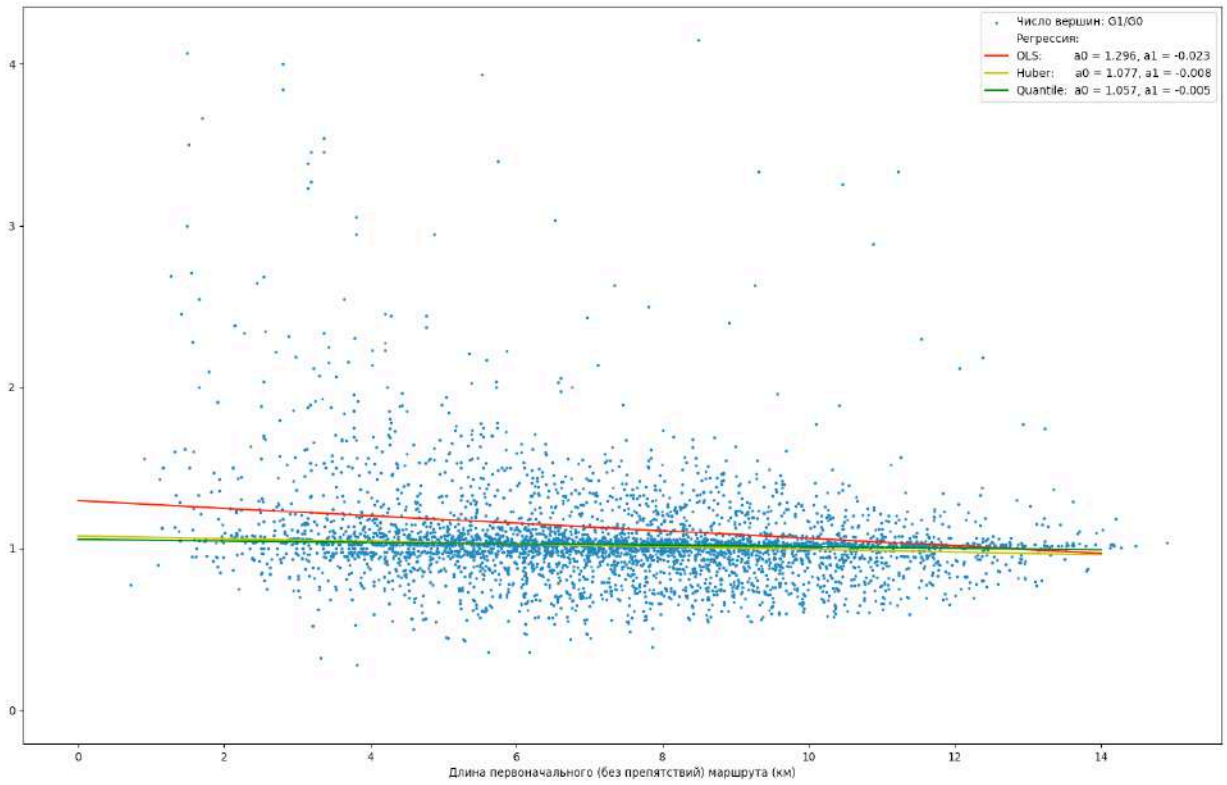

Рисунок 3-3. Отношения числа вершин маршрутов, вычисленных по графам G1 (с учетом препятствий) и G0 (без учета препятствий).

Чтобы лучше представлять, где сосредоточены основные данные, на рисунках 3-2 и 3-3 показаны различные регрессионные зависимости вида y = a0 + a1*x отношений длин (Рис. 3-2) или отношений числа вершин (Рис. 3-3) от длины первоначального (без учета УПО) маршрута. Поскольку данные отличаются большим разбросом, на рисунках показаны также регрессионные прямые, полученные с помощью двух робастных (устойчивых к выбросам) алгоритмов: регрессии Хубера (см. [52, 53]) (желтая прямая) и квантильной регрессии (см. [54]) (зеленая прямая), позволяющей предсказывать произвольные квантили зависимой переменной. В нашем случае мы использовали предсказание для медианы (50% квантиль). Коэффициенты регрессии показаны в пояснениях к рисункам.

Из диаграмм, показанных на Рис. 3-2 и 3-3, хорошо видно, что длина пути в случае обхода препятствий всегда больше «оригинального» пути, построенного без учета препятствий. Также хорошо видно, что отношение длины маршрутов G1/G0 уменьшается по мере увеличения длины оригинального пути.

Что же касается отношения числа вершин G1/G0, то для отдельных маршрутов оно может быть как больше, так и меньше единицы — в зависимости от того, на какую трассу «лег» обходной маршрут. Как показывают данные Таблицы 3-3 (последний столбец), среднее отношение числа вершин стремится к единице для протяженных (более 6 км. маршрутов)[16], и увеличивается (видимо, за счет того, что обходные маршруты оказываются тут существенно длиннее) до 1.52 на самых коротких маршрутах.

Поскольку отношение числа вершин оказывается малоинформативным и стремится к единице для достаточно протяженных маршрутов, основной интерес представляет проигрыш в расстоянии, возникающий при выборе альтернативного (с учетом обхода УПО) маршрута.

Приведем прежде всего общую описательную статистику по всем различающимся (при одних и тех же начальных и конечных координатах) маршрутам (Таблица 3-2). Поскольку распределение отношений различных величин в нашем случае

[16] Отличие от единицы среднего отношения вершин G1/G0 для маршрутов с оригинальной (без учета УПО) протяженностью больше 13 км. связано, видимо, с небольшим размером выборки и вытекающей из этого потерей точности вычисления среднего.

весьма далеки от нормального[17], для вычисления доверительных интервалов всюду используется бутстрэп-метод (см., например [55]). Конкретные расчеты выполнялись с помощью Python-пакета bootstrapped (https://github.com/facebookarchive/bootstrapped).

Таблица 3-2. Общая описательная статистика отношений длин маршрутов (Длина при обходе УПО/ Первоначальная длина)

Показатель	Значение	95% Доверительный интервал
Среднее	1.128	(1.122 - 1.133)
Медиана	1.07	(1.069 - 1.074)
Стандартное отклонение	0.18	(0.161 - 0.203)
Минимум	1.00	
Максимум	3.52	

Таблица 3-3. Зависимость средних отношений длин маршрутов (Длина при обходе УПО/ Первоначальная длина) и числа вершин вдоль маршрутов (Число вершин при обходе УПО/Число вершин первоначального маршрута) для различной длины оригинального (проложенного без учета УПО) маршрута. Для отношений длин и числа вершин в скобках указаны 95% доверительные интервалы

Расстояние (км)	Число маршрутов	Среднее отношение длин	Среднее отношение числа вершин
1-2	63	1,58 (1,404 - 1,747)	1,52 (1,33 - 1,69)
2-3	156	1,36 (1,299 - 1,423)	1,25 (1,17 - 1,32)
3-4	354	1,22 (1,192 - 1,245)	1,18 (1,13 - 1,22)
4-5	471	1,16 (1,142 - 1,174)	1,09 (1,06 - 1,11)
5-6	562	1,13 (1,122 - 1,146)	1,07 (1,04 - 1,09)

[17] Например, для отношений длин, чья описательная статистика показана в Табл. 3-2, тест D'Agostino (см. [56]) scipy.stats.normaltest() дает значение pvalue = 0.0

Продолжение таблицы 3-3

Расстояние (км)	**Число маршрутов**	**Среднее отношение длин**	**Среднее отношение числа вершин**
6-7	659	1,12 (1,107 - 1,125)	1,02 (1,00 - 1,04)
7-8	626	1,10 (1,088 - 1,103)	1,02 (1,01 - 1,04)
8-9	593	1,09 (1,085 - 1,100)	1,02 (1,00 - 1,03)
9-10	502	1,08 (1,077 - 1,091)	1,00 (0,98 - 1,02)
10-11	349	1,07 (1,069 - 1,083)	1,00 (0,97 - 1,02)
11-12	220	1,07 (1,060 - 1,076)	1,00 (0,96 - 1,03)
12-13	119	1,06 (1,054 - 1,072)	1,01 (0,97 - 1,04)
13-14	45	1,05 (1,040 - 1,061)	1,05 (1,00 - 1,09)
14-15	9	1,04 (1,021 - 1,058)	1,05 (1,01 - 1,08)

Таблица 3-4. Описательная статистика отношений (Длина при обходе УПО/ Первоначальная длина) для выбранных интервалов первоначальной длины маршрута. Для среднего, медианы и стандартного отклонения в скобках указаны 95% доверительные интервалы

Длина км,	**N**	**Среднее**	**Медиана**	**Стандартное отклонение**	**Мин.**	**Макс.**	**Диапазон Макс.– Мин.**
1-5	1044	1,23 (1,216 - 1,255)	1,12 (1,106 -1,137)	0,317 (0,286 - 0,358)	1,001	3,521	2,52
5-9	2440	1,11 (1,105 - 1,114)	1,07 (1,061 - 1,072)	0,115 (0,102 - 0,121)	1,000	1,707	0,707
9-13	1190	1,08 (1,073 - 1,081)	1,06 (1,060 - 1,067)	0,069 (0,067 - 0,077)	1,000	1,439	0,439
13-15	54	1,05 (1,04 - 1,059)	1,04 (1,028 - 1,045)	0,034 (0,027 - 0,043)	1,005	1,151	0,146

Результаты расчетов для различной длины исходного (проложенного без учета УПО) маршрута показаны в Таблице 3-3 и на Рис. 3-4.

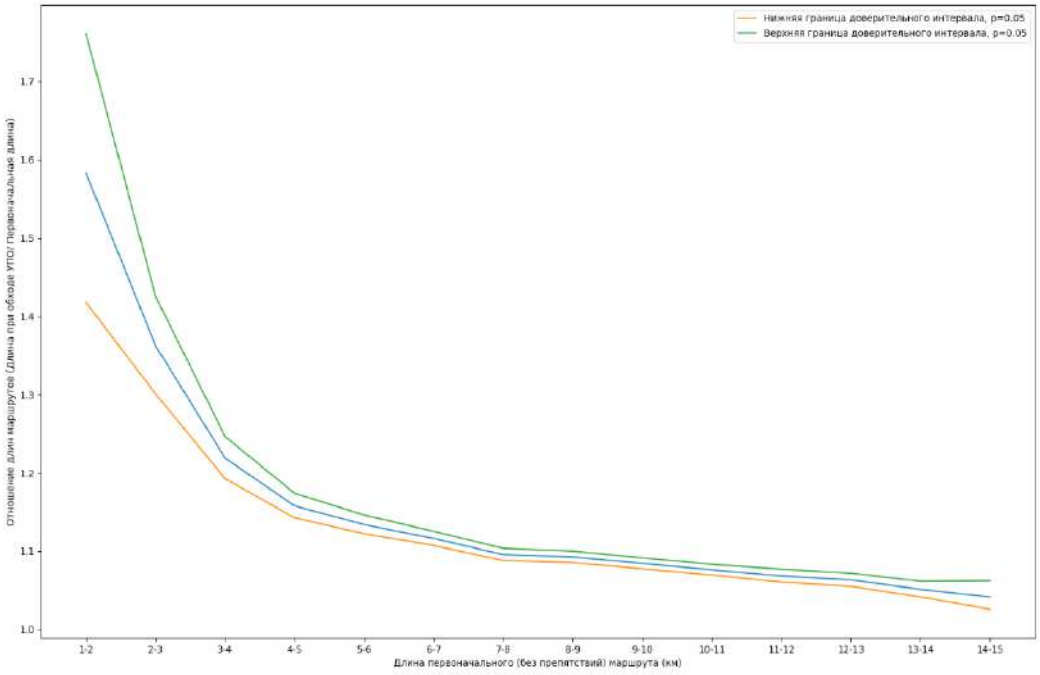

Рисунок. 3-4. Зависимость средних отношений длин маршрутов (Длина при обходе УПО/ Первоначальная длина) для различной длины оригинального, проложенного без учета УПО, маршрута

Наконец, в Таблице 3-4 дана более подробная описательная статистика отношений (Длина при обходе УПО/ Первоначальная длина) для четырехкилометровых интервалов первоначальной длины маршрута, из которой видно, что все показатели (среднее, медиана, стандартное отклонение, разница между максимальным и минимальным значением) плавно уменьшаются по мере увеличения длины оригинального маршрута.

3.3 Относительный риск ДТП при выборе маршрута, обходящего УПО

Как видно из предыдущего раздела, обход УПО гарантирует увеличение длины маршрута. При этом совсем не очевидно, что альтернативный маршрут будет более безопасным. Весьма важно поэтому оценить относительный риск ДТП при выборе маршрута, обходящего УПО.

Пусть необходимо проложить маршрут из A в B. Обозначим A-----> B маршрут с обходом УПО, а A-> B - маршрут без обхода УПО. Пусть ДТП1 – число серьезных ДТП, зарегистрированных вдоль маршрута A-----> B, а ДТП2 – число серьезных ДТП вдоль маршрута A->B. Тогда отношение ДТП1/ДТП2 будет мерой относительного риска на двух маршрутах[18].

Для вычисления относительного риска необходимо перебрать координаты всех серьезных ДТП, случившихся в Спрингфилде в 2013-2018 годах и определить для каждого ДТП ближайшую к нему грань в графе G0. Для этого используется метод get_nearest_edge() пакета OSMnx.

К сожалению, get_nearest_edge() дает однозначные результаты лишь для однонаправленных граней и не в состоянии определить, какому направлению двунаправленной грани принадлежит конкретное ДТП. Чтобы узнать это, можно, казалось бы, сопоставить данные о направлении ДТП с данными о направлении трассы (например S, W, N, E), которой принадлежит данное ребро графа. Но данные о направлении движения известны далеко не для всех ДТП, к тому же, они не особо надежны и для трассы, идущей с севера на юг, может быть указано направление W или E.

Таким образом, нам остается только присвоить каждому направлению двунаправленной грани половину всех найденных ДТП и надеяться, что усреднение по многим граням для достаточно протяженного маршрута

[18] Здесь необходимо предположить, что выбор альтернативного маршрута происходит достаточно редко, иначе исказилась бы статическая картина ДТП, построенная без учета обхода препятствий

компенсирует неточность определению числа ДТП для конкретного направления двунаправленной грани.

Итак, с учетом замечаний о числе ДТП для двунаправленной грани подсчитаем для каждой пары маршрутов А-----> В и А-> В число ДТП вдоль маршрута, после чего можно получить общую описательную статистику отношений ДТП1/ДТП2 по всем маршрутам (Таблица 3-5).

Таблица 3-5. Общая статистика отношений
(ДТП вдоль А-----> В/ДТП вдоль А-> В)

Показатель	**Значение**	**95% доверительный интервал**
Среднее	0,91	(0,89 - 0,93)
Медиана	0,84	(0,83 - 0,842)
Стандартное отклонение	0,73	(0,49 - 0,96)
Минимум	0,13	
Максимум	27,75	

Таблица 3-6. Зависимость средних отношений (ДТП вдоль А-----> В/ДТП вдоль А-> В) для различной длины маршрута А-> В

Расстояние (км)	**N**	**Среднее отношение числа ДТП**	**95% доверительный интервал**
1-2	63	1,32	(0,834 - 1,689)
2-3	156	1,16	(0,978 - 1,335)
3-4	354	1,02	(0,948 - 1,087)
4-5	471	0,94	(0,894 - 0,979)
5-6	562	0,88	(0,850 - 0,909)
6-7	659	0,85	(0,828 - 0,874)
7-8	626	0,85	(0,830 - 0,871)
8-9	593	0,84	(0,823 - 0,859)
9-10	502	0,82	(0,808 - 0,842)
10-11	349	0,82	(0,803 - 0,846)
11-12	220	0,81	(0,789 - 0,839)
12-13	119	0,84	(0,809 - 0,870)
13-14	45	0,84	(0,796 - 0,886)
14-15	9	0,83	(0,738 - 0,926)

Таблица 3-7. Описательная статистика отношений (ДТП вдоль А-----> В/ДТП вдоль А-> В) для выбранных интервалов первоначальной длины маршрута. Для среднего, медианы и стандартного отклонения в скобках указаны 95% доверительные интервалы.

Длина км	N	Среднее	Медиана	Стандартное отклонение	Мин,	Макс,	Диапазон Макс,- Мин,
1-5	1044	1,02 (0,97 - 1,07)	0,85 (0,84 - 0,87)	0,80 (0,64 - 0,95)	0,17	10,0	9,83
5-9	2440	0,86 (0,84 - 0,87)	0,83 (0,82 - 0,84)	0,29 (0,27 - 0,31)	0,25	3,57	3,32
9-13	1190	0,82 (0,81 - 0,840)	0,84 (0,82 - 0,85)	0,20 (0,19 - 0,21)	0,27	1,56	1,29
13-15	54	0,83 (0,80 - 0,88)	0,85 (0,83 - 0,9)	0,15 (0,12 - 0,18)	0,43	1,22	0,79

В Таблице 3-6. показан относительный риск ДТП в зависимости от длины оригинального маршрута А-> В. Как и ранее, для вычисления доверительного интервала использовался бутстреп-метод.

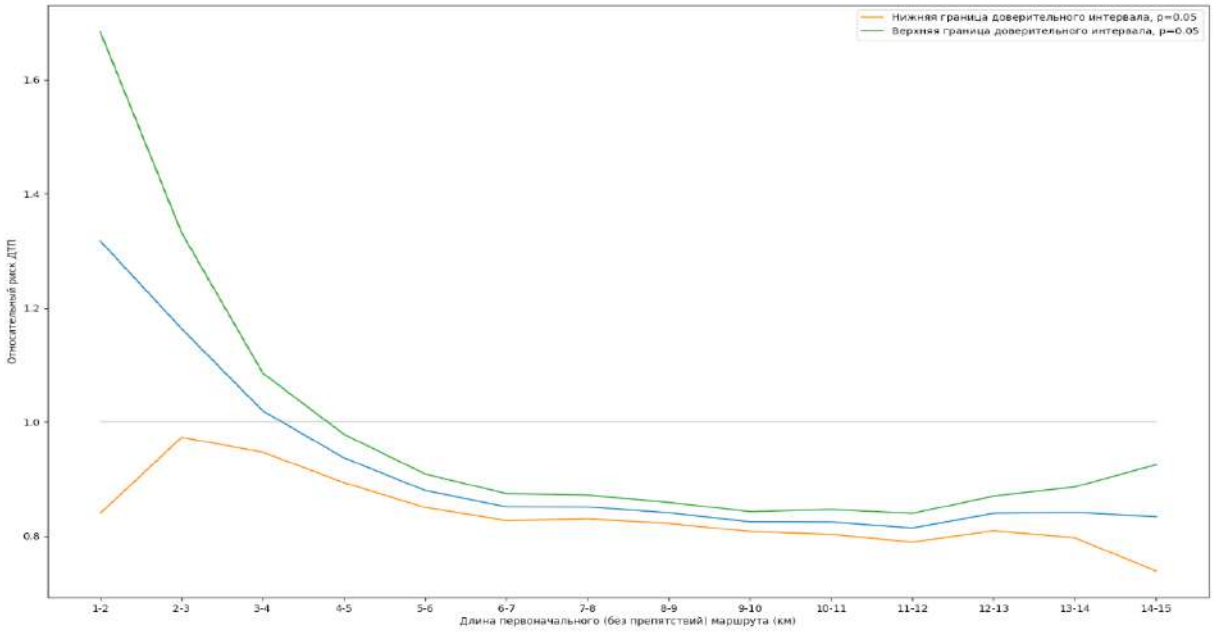

Рисунок 3-5. Зависимость средних отношений (ДТП вдоль А-----> В/ДТП вдоль А-> В) для различной длины маршрута А-> В.

Те же, что и в Таблице 3-6, результаты более наглядно показаны на Рис. 3-5, где серая горизонтальная прямая соответствует уровню относительного риска 1.0. Сами значения показаны синим цветом, а доверительные интервалы — оранжевым и зеленым.

Как видим, снижение относительного риска происходит только для достаточно протяженных маршрутов (в нашем случае — более 4 км.). Что же касается меньших расстояний, то там, согласно Таблице 3-3, происходит значительное увеличение протяженности альтернативного (обходящего УПО) маршрута и как следствие — увеличение числа ДТП вдоль него.

Наконец, в Таблице 3-7 дана более подробная описательная статистика отношений (ДТП вдоль А-----> В/ДТП вдоль А-> В) для четырехкилометровых интервалов первоначальной длины маршрута. Как видим, все статистические показатели либо уменьшаются по мере увеличения первоначальной длины маршрута (стандартное отклонение, разница между максимальным и минимальным значением), либо выходят на «плато» (среднее, медиана.

Комбинируя Таблицы 3-4 и 3-7, можно сопоставить увеличение среднего отношения длин маршрутов с изменением среднего относительного риска ДТП для различных интервалов протяженности оригинального (без учета УПО) маршрута (Таблица 3-8). Так, например, для интервала 5-9 км. имеем в среднем увеличение длины маршрута на 11% и уменьшение относительного риска ДТП на 14%.

Table 3-8. Зависимость изменения среднего относительного риска ДТП от среднего увеличения длины маршрута, обходящего УПО, для различных интервалов оригинального, не учитывающего УПО, маршрута. В скобках указаны соответствующие 95% доверительные интервалы

Длина оригинального маршрута, км	**Увеличение среднего отношения длин маршрутов, %**	**Изменение среднего относительного риска, ДТП %**
1-5	23% (21,6% - 25,5%)	+2% (-3%, + 7%)
5-9	11% (10,5% - 11,4%)	-14% (-13%, - 16%)

Продолжение таблицы 3-8.

Длина оригинального маршрута, км	**Увеличение среднего отношения длин маршрутов, %**	**Изменение среднего относительного риска, ДТП %**
9-13	8% (7,3% - 8,1%)	-18% (-16%, -19%)
13-15	5% (4,1% - 5,9%)	-17% (-12%, -20%)

Для оригинальных (не учитывающих УПО) длин маршрута в интервале 1-5 км. маршрут, обходящий УПО, в среднем на 23% (95% доверительный интервал 21.6% - 25.5%) длиннее, чем маршрут, не учитывающий препятствия, и риск ДТП возрастает в среднем на 2% (95% доверительный интервал: -3%, +7%).

Для оригинальных (не учитывающих УПО) длин маршрута в интервале 5-9 км. маршрут, обходящий УПО, в среднем на 11% (95% доверительный интервал 10.5% - 11.4%) длиннее, чем маршрут, не учитывающий препятствия и риск ДТП уменьшается в среднем на 14% (95% доверительный интервал: 13%-16%).

Для оригинальных (не учитывающих УПО) длин маршрута в интервале 9-13 км. маршрут, обходящий УПО, в среднем на 8% (95% доверительный интервал 7.3% - 8.1%) длиннее, чем маршрут, не учитывающий препятствия и риск ДТП уменьшается в среднем на 18% (95% доверительный интервал: 16%-19%).

Для оригинальных (не учитывающих УПО) длин маршрута в интервале 13-15 км. маршрут, обходящий УПО, в среднем на 5% (95% доверительный интервал 4.1% - 5.9%) длиннее, чем маршрут, не учитывающий препятствия и риск ДТП уменьшается в среднем на 17% (95% доверительный интервал: 12%-20%).

Выводы

Метакластеры серьезных ДТП, выявленные в Главе 2 (см. также [48]), могут быть использованы соответствующими службами для выявления причин, приводящим к возникновению ДТП, и устранения их.

Водителям автомобилей, такси и автобусов в городе Спрингфилд при использовании маршрута протяженностью более 4 км можно рекомендовать алгоритм, позволяющий обойти метакластеры серьезных дорожно-транспортных происшествий, описанный в данной главе, и тем самым снизить риск серьезного ДТП в среднем на 16% при увеличении длины маршрута в среднем на 8%.

Глава 4. Простой способ повышения безопасности дорожного движения за счет обхода опасных участков маршрута (на примере г. Москвы)

Введение

В работе [57] показано на примере города Springfield MA, что путем обхода опасных участков дорожной сети (кластеров ДТП) удается понизить относительный риск ДТП за счет некоторого увеличения длины маршрута и числа проходимых вершин графа, представляющего дорожную сеть. Построение более безопасного маршрута требовало в [57] некоторой ручной работы: необходимо было рассмотреть все дороги (ребра дорожного графа), ведущие к опасному участку (кластеру), и пометить их таким образом, чтобы они стали непроходимыми для алгоритма маршрутизации. Для небольшого города это не требовало больших усилий, но для мегаполиса, подобного Москве, эта задача становится слишком трудоемкой. Кроме того, всегда есть сомнение, стоит ли запрещать все пути, ведущие к кластеру, или, быть может, только некоторые, самые «опасные»?

В данной главе предложен несколько иной метод, в основе которого лежит способ определения опасности отдельных участков дороги (ребер дорожного графа), состоящий в том, что из всех ребер выбираются лишь те, вдоль которых совершается наибольшее количество ДТП и для ребер с наибольшим числом ДТП проводятся статистические испытания, которые позволяют выделить лишь те ребра, где число ДТП статистически достоверно. Эти статистически достоверные ребра помечаются в модифицированном дорожном графе тем, что к атрибуту их длины прибавляется фиксированное число, что заставляет обычный алгоритм маршрутизации, такой как алгоритм Дейкстры[2] или Беллмана-Форда [8, 9], изменить маршрут и включить в него менее опасные (не помеченные) ребра дорожного графа. Заметим, что в работе [57] к атрибуту длины ребра прибавлялось, по сути, бесконечное число, что приводило к тому, что такое ребро практически

выключалась из маршрута. Как показано в данной работе, такой подход был не самым оптимальным и прибавка к атрибуту длины конечного числа, сравнимого со средней длиной маршрута, приводит к лучшим результатам.

4.1 Данные

Для получения дорожной сети города Москвы воспользуемся глобальной картой мира OpenStreetMap (OSM). Карты OSM, как правило, очень хорошего качества и широко используются в работах, посвященных маршрутизации (см. например, [13], в которой затрагивается похожая проблема обхода криминально опасных районов пешеходами). Получить дорожную сеть Москвы проще всего с помощью библиотеки OSMnx для языка Python (см. [6]). Полученная дорожная сеть представляет собой мультидиграф, то есть, направленный граф, ребра которого могут быть одно- и двунаправленными, а через любые две вершины графа может проходить сколько угодно ребер. Пакет OSMnx позволяет сохранить исходный мультидиграф в виде двух отдельных объектов – ребер и вершин в векторном формате .shp. Это позволяет отобразить дорожную сеть в геоинформационной системе (например, в QGIS [59]).

Дорожная сеть Москвы, полученная из OSM, содержит 25397 ребер (15696 однонаправленных и 9701 двунаправленных) и 16797 вершин. Описательная статистика некоторых атрибутов ребер приведена в таблице 4-1.

Что же касается данных о ДТП, то они получены из официального источника ГИБДД [60]. Для данного исследования мы выбрали данные о 27798 ДТП, зафиксированных в Москве в 2019-2021 годах. Поскольку дорожная сеть Москвы и данные о ДТП получены из независимых источников, необходимо удалить ДТП, явно не принадлежащие (например, из-за ошибки регистрации) дорожной сети. С этой целью мы оставили для дальнейшего анализа лишь те ДТП, чье расстояние до

ближайшего ребра дорожной сети меньше 35м[19]. Всего таких ДТП оказалось 21956 (79% от общего числа).

Таблица 4-1. Описательная статистика некоторых атрибутов ребер дорожного графа г. Москвы

Показатель	Длина (м)	Скорость(км/ч)	Время пути(мин)
Среднее	217,4	51,7	15,9
Ст. отклонение	275,5	12,13	20,9
Минимум	1,06	5	0,1
25% процентиль	47,2	41,9	3,3
Медиана	127,9	50,5	9,2
75% процентиль	291,6	53,1	21,2
Максимум	8043	100	673

4.2 Инструменты и метод

Список отфильтрованных ДТП, случившихся в Москве в 2019-2021 годах, о котором мы говорили в предыдущем пункте, содержит координаты ДТП, расстояние от ДТП до ближайшего ребра и само ребро в виде <вершина1> <вершина2> <ключ>, где <вершина1> и <вершина2> - идентификаторы вершин дорожного графа, через которые проходит ребро, а <ключ> - номер ребра, проходящего через конкретные вершины. В случае, когда ребро только одно <ключ> = 0. В случае, когда ребер два, существуют ключи под номерами 0,1 и т.д. Таким образом, ключи позволяют идентифицировать ребра, проходящие через одни и те же вершины. Теперь можно перебрать все ребра дорожного графа и искать каждое в этом списке, тем самым определяя, какие ДТП принадлежат данному ребру. Естественно, для двунаправленных ребер нужно искать как ребро <вершина1> <вершина2> <ключ>, так и ребро <вершина2> <вершина1> <ключ>. Полученный список ребер можно отсортировать по количеству ДТП,

[19] Если считать ширину полосы равной 3.5 м., а максимальное количество полос равным 10, то 35м. будет служить оценкой максимальной ширины полосы в Москве.

принадлежащих им, и затем выделить значимые ребра, проведя статистические испытания.

Для статистических испытаний нами использовался программный пакет SANET [47], позволяющий получить миллионы точек, равномерно распределенных вдоль дорожной сети. Испытания состоят в том, что формируются группы из 21956 точек (столько же, сколько произошло ДТП за 2019-2021 годы) и для каждой группы вновь перебираются все ребра и находится число точек, принадлежащих каждому рбру. Произведя, скажем, 1000 испытаний, получим статистику ДТП в случае, когда ДТП распределены равномерно по дорожной сети. Если число реальных ДТП ребра превысит 95-й или 99-й процентиль, полученный в результате статистических испытаний, будем считать, что число ДТП данного ребра достоверно на уровне 5% и 1% соответственно[20]. Полученные таким образом «опасные» ребра можно использовать так же, как и в [57].

Для этого создадим два идентичных дорожных графа G0 и G1. Граф G0 оставим без изменений, а в графе G1 изменим атрибут <Length> ребер, выбранных в результате статистических испытаний, добавив к нему достаточно большое число, одинаковое для всех таких ребер, чтобы показать алгоритму маршрутизации, что это ребро нежелательно в прокладываемом маршруте и его по возможности следует избегать.

Нам осталось только сравнить все маршруты, проложенные по графам G0 и G1 и убедиться в том, что маршруты на графе G1 в некотором смысле безопасней. Для этого зададим квадратную сетку с фиксированным шагом, равным в нашем случае 3 км. (Рис. 4-1), и построим все маршруты, начинающиеся и заканчивающиеся в узлах этой сетки[21].

Обозначим два любых узла сетки как A и B и найдем маршрут A->B[22] на графе G0 и маршрут A—>B на графе G1. Для сравнения этих маршрутов подсчитаем

[20] Заметим, что число ДТП для выбранного ребра может быть как выше полученного в результате статистических испытаний (превышать 95% или 99% процентиль), так и ниже (попадать в 1% или 5% процентиль). В этом случае можно применить поощрение вместо штрафа (например, что-то вычитать из атрибута длины). В данной работе эта возможность не используется.

[21] Точнее говоря, начинающиеся и заканчивающиеся в вершинах графа, ближайших к узлам сетки

[22] Маршрут A->B не эквивалентен B->A, поскольку может содержать однонаправленные ребра.

относительный риск ДТП (ОРДТП[23]), равный отношению ДТП, случившихся вдоль маршрута A—>B к числу ДТП, произошедших вдоль маршрута A->B. Если верхняя граница доверительного интервала для среднего ОРДТП < 1, то маршруты, проложенные по графу G1 более безопасны, чем маршруты, определенные на исходном графе G0.

Рис. 4-1. Узлы квадратной сетки, наложенные на дорожную карту Москвы

Заметим, что в [57] для собирания статистики использовались только отличающиеся для графов G0 и G1 маршруты. В данной главе используются *все* маршруты, поскольку в зависимости от значения, прибавляемого к атрибуту <Length>, количество маршрутов, отличающихся для G0 и G1, будет различно.

[23] Определение ОРДТП см. во Введении данной работы на странице 5.

4.3 Результаты

Для статистических испытаний были выбраны первые 2000 ребер дорожного графа, содержащие наибольшее число ДТП, поскольку расчеты для всех 25397 ребер крайне трудоемки[24] и по большому счеты бессмысленны. Всего для каждого ребра из 2000 было проведено 1000 испытаний, значимо «опасными» оказались на уровне 0,05 – 815 ребер и на уровне 0,01 - 469 ребер дорожного графа. Элементарная статистика для «опасных» ребер, приведенная в таблице 4-1, показывает, что на уровне значимости 0.01 препятствия характеризуются в среднем большим числом и большим разбросом числа ДТП.

Таблица 4-2. Элементарная статистика для «опасных» ребер

Вероятность	p = 0,05	p = 0,01
Всего ребер	815	469
Среднее	6,6	7,8
Стандартное отклонение	5,4	6,3
Минимум	3	3
25% процентиль	4	4
Медиана	5	6
75% процентиль	7,5	9
Максимум	58	58

Рис. 4-2 показывает 815 ребер-препятствий на карте Москвы. Ребра с различным числом ДТП показаны разным цветом (3-7 ДТП – зеленый, 7-20 ДТП – красный, более 20 ДТП - желтый). Отметим, что ребра с наибольшим числом ДТП располагаются на Центральной кольцевой дороге (ЦКАД), а также на отрезке Кутузовского проспекта от улицы Минской до улицы Алексея Свиридова и на участке Третьего кольца от шоссе Энтузиастов до Новой Переведеновской.

После получения препятствий порядок действий становится таким же, как и в [57]. Необходимо, пользуясь квадратной сеткой, наложенной на карту Москвы (см. Рис. 4-1), перебрать все начальные и конечные точки и для каждого маршрута A->B,

[24] Все расчеты производились на рабочей станции с четырехъядерным процессором и 32гб оперативной памяти

проложенного без учета препятствий, вычислить маршрут A—>B, имеющий те же начальную и конечную точки, но проложенный с учетом препятствий. Далее следует сравнить все такие маршруты и понять, как в среднем меняется относительный риск ДТП, длина маршрута и количество проходимых вершин дорожного графа. В отличие от [57] мы провели этот анализ для различных значений штрафа, налагаемого на препятствие, чтобы понять, как он влияет на эффективность альтернативной маршрутизации.

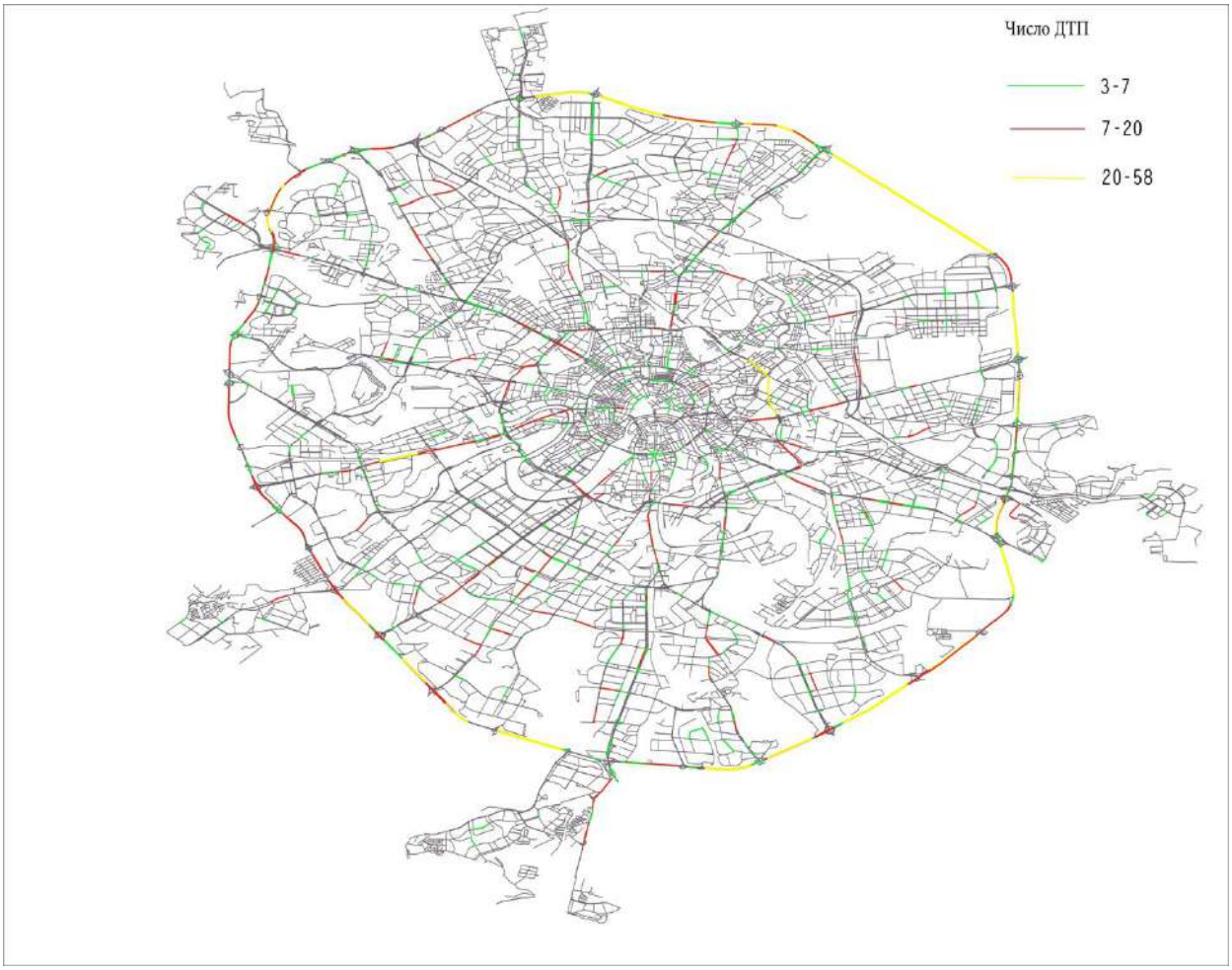

Рис. 4-2. Препятствия, отображенные на дорожной карте Москвы

Результаты расчетов показаны на рисунках 4-3… 4-5 и в таблицах 4-3… 4-5. Кроме самих значений в таблицах 4-3… 4-5 приведены более мелким шрифтом, а на рисунках показаны цветом меньшей насыщенности 95% доверительные интервалы значений, полученные бутстрэп-методом [25]. В качестве препятствий

[25] бутстрэп — это статистический метод для оценки количественных показателей популяции путем усреднения оценок по нескольким небольшим выборкам данных. Выборки создаются путем извлечения наблюдений из

использовались первые 800 значимых на уровне 0,05 ребер, а в качестве штрафа значения 200, 500, 1000, 2000, 5000 и 10000000000000 метров. Последнее значение, намного превышающее все мыслимые расстояния в Москве, можно считать практически бесконечным, что и отображено на рисунках 4-3…4-5 и в таблицах 4-3…4-5.

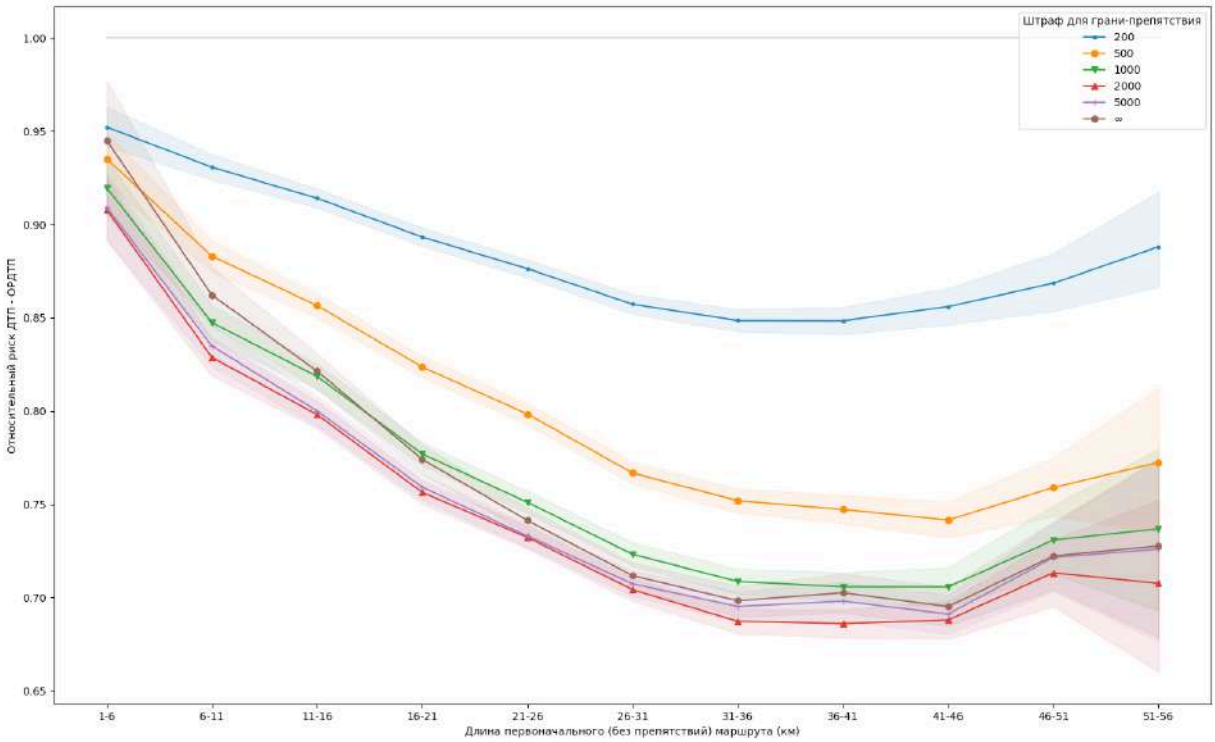

Рисунок 4-3. Относительный риск ДТП в зависимости от длины исходного маршрута и штрафа для ребра-препятствия с 95% доверительными интервалами

Как видно из рисунков и таблиц, оптимальное значение штрафа равно 2000м. При этом значении кривая, показывающая зависимость среднего относительного риска ДТП от длины исходного маршрута (Рис. 4-3) лежит чуть ниже всех остальных и минимальное значение относительного риска достигает, согласно таблице 4-3, величины 0.686.

большой выборки данных по одному за раз и возврата их в выборку данных после того, как они были выбраны. Конкретные расчеты проводились с помощью пакета bootstrapped для языка Python.

Таблица 4-3. Относительный риск ДТП в зависимости от длины исходного маршрута и штрафа для ребра-препятствия с 95% доверительными интервалами

Интервал км,	Число маршрутов	200м	500 м	1000 м	2000 м	5000 м	∞
1-6	535	0,952 0,942-0,963	0,935 0,922-0,949	0,920 0,905-0,935	0,908 0,892-0,925	0,909 0,892-0,927	0,945 0,907-0,977
6-11	1524	0,931 0,924-0,938	0,883 0,874-0,891	0,847 0,838-0,857	0,829 0,819-0,839	0,835 0,824-0,846	0,862 0,846-0,877
11-16	2679	0,914 0,909-0,919	0,857 0,850-0,863	0,819 0,812-0,825	0,798 0,791-0,805	0,800 0,792-0,808	0,821 0,812-0,831
16-21	3065	0,893 0,888-0,898	0,824 0,818-0,829	0,777 0,771-0,783	0,756 0,750-0,763	0,759 0,752-0,766	0,774 0,766-0,782
21-26	3255	0,876 0,872-0,881	0,798 0,793-0,804	0,751 0,745-0,757	0,732 0,726-0,738	0,733 0,727-0,739	0,741 0,734-0,748
26-31	2854	0,857 0,852-0,863	0,767 0,761-0,773	0,723 0,717-0,729	0,704 0,698-0,710	0,707 0,701-0,714	0,712 0,705-0,719
31-36	2196	0,849 0,843-0,855	0,752 0,745-0,758	0,709 0,702-0,715	0,687 0,681-0,694	0,695 0,687-0,703	0,698 0,690-0,706
36-41	1315	0,848 0,841-0,856	0,747 0,740-0,755	0,706 0,698-0,713	0,686 0,678-0,693	0,698 0,687-0,708	0,703 0,692-0,713
41-46	591	0,856 0,846-0,866	0,742 0,732-0,751	0,706 0,695-0,716	0,688 0,678-0,698	0,691 0,680-0,701	0,695 0,685-0,706
46-51	179	0,869 0,854-0,885	0,759 0,743-0,775	0,731 0,713-0,749	0,713 0,695-0,731	0,721 0,703-0,740	0,722 0,704-0,741
51-56	32	0,888 0,867-0,917	0,772 0,735-0,813	0,737 0,693-0,780	0,708 0,660-0,752	0,726 0,676-0,774	0,728 0,679-0,774
Доля измененных маршрутов		0,7	0,84	0,9	0,92	0,94	0,94

Более того, при этом штрафе зависимость отношений длин маршрутов (длина с учетом препятствий/длина без учета препятствий) от длины первоначального маршрута (рис. 4-4) ведет себя гораздо более плавно, чем при штрафе, равном бесконечности и 5000м. Согласно таблице 4-4, среднее отношений длин маршрута при штрафе 2000м колеблется в пределах 1.064 - 1.105, то есть (см. таблицы 4-3, 4-4) *среднее уменьшение относительного риска на 9-31% достигается при штрафе 2000м за счет увеличения средней длины маршрута на 6-11%.*

Как видно из Рис. 4-4, при бесконечном значении штрафа и небольших длинах оригинального пути отношение длин маршрутов сильно возрастает, очевидно, из-за того, что алгоритм маршрутизации испытывает трудности при нахождении альтернативного маршрута. Между тем, штраф 2000м дает весьма плавную зависимость и небольшое изменение отношения Средней длины пути после/до (6%-10.5%)

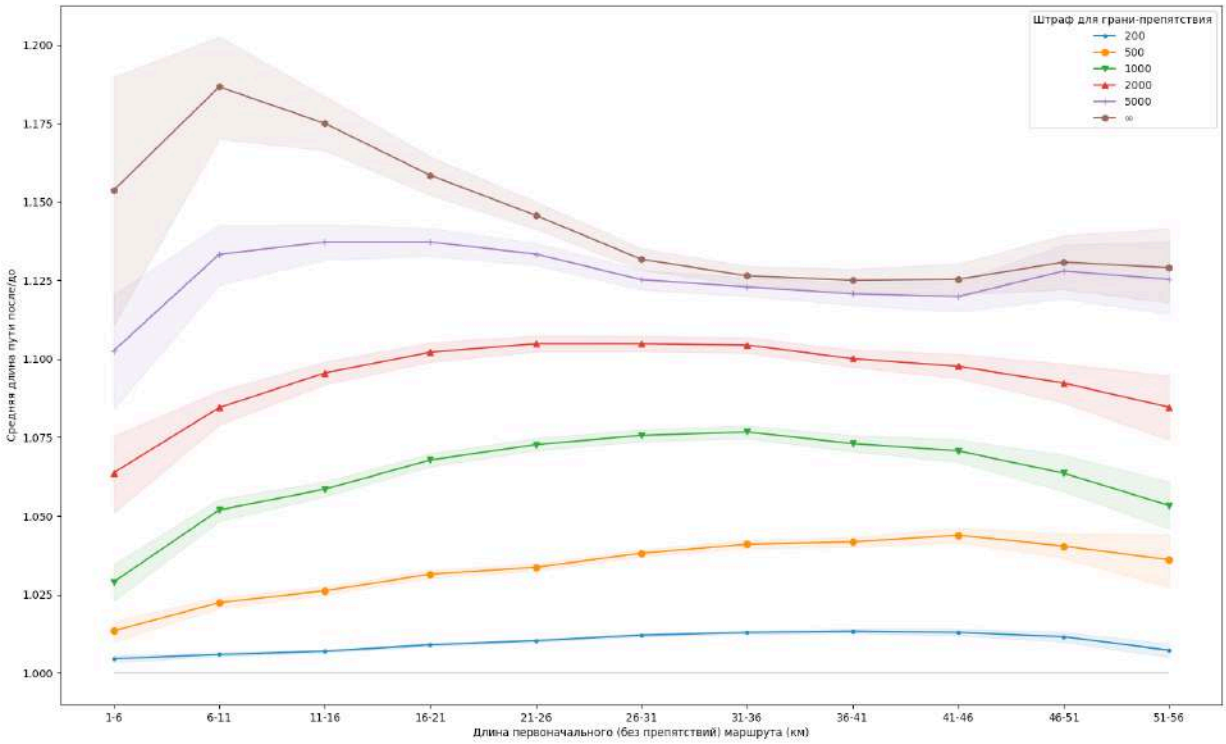

Рисунок 4-4. Зависимость средних отношений длин маршрутов (Длина при обходе УПО/ Первоначальная длина) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия

Таблица 4-4. Средние отношения длин маршрутов (после/до) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия (с 95% доверительными интервалами)

Интервал км	**Число маршрутов**	**200 м**	**500 м**	**1000 м**	**2000 м**	**5000 м**	∞
1-6	535	1.005 1.003-1.006	1.013 1.010-1.016	1.029 1.023-1.035	1.064 1.051-1.075	1.103 1.084-1.120	1.154 1.111-1.190
6-11	1524	1.006 1.005-1.007	1.022 1.021-1.024	1.052 1.048-1.055	1.084 1.079-1.090	1.133 1.124-1.143	1.187 1.170-1.203
11-16	2679	1.007 1.007-1.007	1.026 1.025-1.027	1.059 1.056-1.061	1.096 1.092-1.099	1.137 1.132-1.143	1.175 1.166-1.184
16-21	3065	1.009 1.009-1.009	1.031 1.030-1.033	1.068 1.066-1.070	1.102 1.099-1.105	1.137 1.133-1.142	1.158 1.152-1.164
21-26	3255	1.010 1.010-1.011	1.034 1.033-1.035	1.073 1.071-1.075	1.105 1.102-1.107	1.133 1.130-1.137	1.146 1.141-1.150
26-31	2854	1.012 1.012-1.013	1.038 1.037-1.039	1.076 1.074-1.078	1.105 1.102-1.107	1.125 1.122-1.128	1.132 1.128-1.135
31-36	2196	1.013 1.012-1.014	1.041 1.040-1.042	1.077 1.075-1.079	1.104 1.102-1.107	1.123 1.120-1.126	1.126 1.123-1.130
36-41	1315	1.013 1.013-1.014	1.042 1.040-1.043	1.073 1.071-1.076	1.100 1.097-1.103	1.121 1.117-1.124	1.125 1.121-1.129

Продолжение таблицы 4-4.

Интервал км	**Число маршрутов**	**200 м**	**500 м**	**1000 м**	**2000 м**	**5000 м**	∞
41-46	591	1.013 1.012-1.014	1.044 1.042-1.046	1.071 1.067-1.074	1.098 1.094-1.102	1.120 1.115-1.125	1.125 1.120-1.130
46-51	179	1.012 1.010-1.013	1.040 1.037-1.044	1.064 1.058-1.069	1.092 1.086-1.098	1.128 1.119-1.136	1.131 1.122-1.139
51-56	32	1.007 1.005-1.009	1.036 1.027-1.044	1.053 1.046-1.061	1.085 1.074-1.095	1.125 1.114-1.137	1.129 1.118-1.141

Уменьшение относительного риска сопровождается, как это следует из Рис. 4-5 и таблицы 4-5, еще одной потерей: увеличением числа проходимых вершин дорожного графа примерно на 6-27,6% в зависимости от протяженности исходного маршрута.

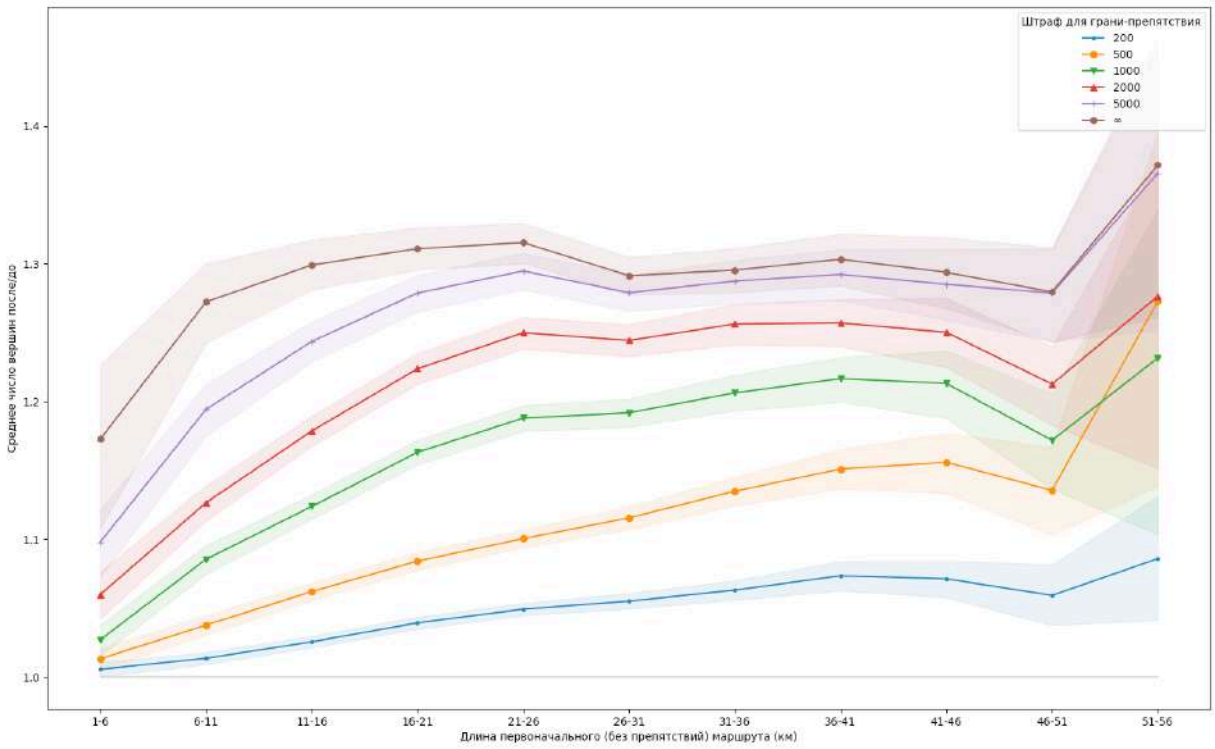

Рисунок 4-5. Средние отношения числа вершин маршрутов (после/до) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия (с 95% доверительными интервалами)

Таблица 4-5. Средние отношения числа вершин маршрутов (после/до) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия (с 95% доверительными интервалами)

Интервал км	**Число маршрутов**	**200 м**	**500 м**	**1000 м**	**2000 м**	**5000 м**	∞
1-6	535	1,006 1,000-1,011	1,013 1,005-1,021	1,027 1,015-1,038	1,060 1,042-1,076	1,098 1,073-1,121	1,173 1,108-1,227
6-11	1524	1,014 1,009-1,018	1,038 1,031-1,045	1,086 1,075-1,096	1,127 1,114-1,139	1,194 1,175-1,213	1,272 1,243-1,300
11-16	2679	1,026 1,021-1,030	1,062 1,056-1,068	1,124 1,115-1,132	1,179 1,168-1,190	1,244 1,229-1,258	1,299 1,281-1,317
16-21	3065	1,039 1,035-1,044	1,084 1,078-1,091	1,163 1,154-1,172	1,224 1,213-1,235	1,279 1,265-1,292	1,311 1,296-1,326
21-26	3255	1,049 1,045-1,054	1,101 1,094-1,107	1,188 1,179-1,197	1,250 1,238-1,261	1,295 1,281-1,308	1,315 1,300-1,329
26-31	2854	1,055 1,050-1,061	1,116 1,107-1,124	1,192 1,182-1,202	1,244 1,233-1,256	1,279 1,266-1,292	1,291 1,277-1,305
31-36	2196	1,063 1,056-1,070	1,135 1,124-1,145	1,206 1,193-1,219	1,256 1,241-1,271	1,287 1,271-1,303	1,295 1,279-1,311
36-41	1315	1,074 1,063-1,084	1,151 1,137-1,165	1,217 1,200-1,232	1,257 1,240-1,274	1,292 1,274-1,310	1,303 1,284-1,322
41-46	591	1,071 1,058-1,084	1,156 1,133-1,177	1,213 1,188-1,237	1,250 1,225-1,275	1,285 1,259-1,311	1,294 1,268-1,319
46-51	179	1,059 1,038-1,082	1,136 1,103-1,167	1,172 1,137-1,206	1,213 1,183-1,240	1,279 1,243-1,311	1,280 1,243-1,312
51-56	32	1,086 1,041-1,131	1,273 1,139-1,395	1,232 1,103-1,339	1,276 1,151-1,375	1,366 1,252-1,458	1,372 1,261-1,463

В заключение кратко остановимся на оптимальном выборе препятствий (их числа и уровня значимости), который обеспечил бы наилучшее значение относительного риска ДТП. К сожалению, эту проблему приходится пока решать эмпирически – при помощи довольно трудоемких расчетов. Прежде всего сравним первые 400 препятствий с уровнем значимости 0,05 и 0,01, обозначенных как 95_400 и 99_400 на Рис. 4-6. Видно, что достоверность 0,01 оказывается выгодней, что не удивительно, ведь согласно таблице 2 (да и здравому смыслу) ребра, достоверные на уровне 0,01, в среднем содержат больше ДТП и больше годятся на роль препятствий. Теперь сравним разное количество препятствий, достоверных на уровне 0,05, и обозначенных на Рис. 4-6 как 95_400, 95_500 и 95_800. Хорошо видно, что в нашем конкретном случае наибольшее число препятствий 800 оказывается оптимальным, хотя вывести отсюда правило «чем больше, тем лучше»

нельзя, поскольку слишком большое число препятствий может оставить мало альтернатив для изменения маршрута, что приведет к увеличению относительного риска ДТП вместо его уменьшения.

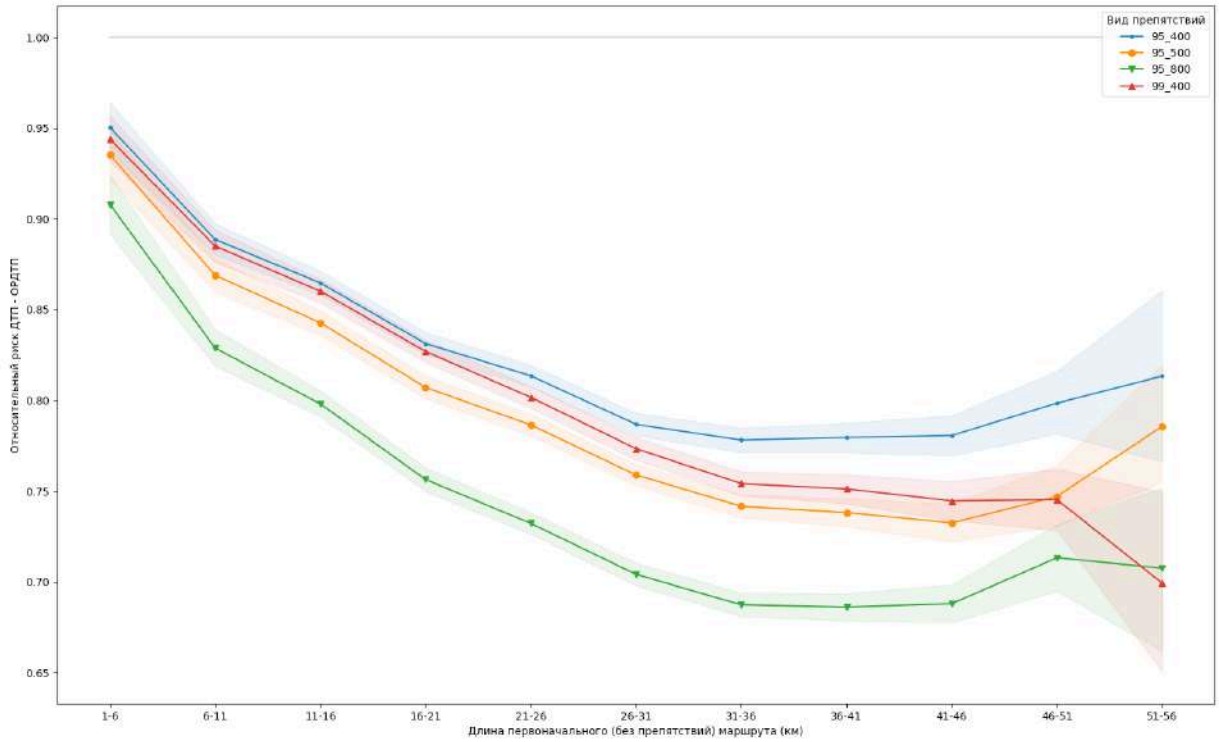

Рисунок 4-6. Зависимость среднего относительного риска ДТП от выбора различных препятствий для разных длин исходного маршрута и штрафе 2000 м

Выводы

На примере г. Москвы рассмотрен способ повышения безопасности дорожного движения, заключающийся в построении маршрута, обходящего препятствия, выявленные на дорожной карте (графе). Препятствиями служат ребра дорожного графа, содержащие статистически достоверно большое число дорожно-транспортных происшествий (ДТП).

Для проверки эффективности маршрутизации используется показатель, предложенный в [57] – относительный риск ДТП, равный отношению числа ДТП вдоль маршрута, учитывающего препятствия, к числу ДТП, подсчитанных вдоль исходного, построенного без учета препятствий, маршрута.

Показано, что обход препятствий позволяет снизить относительный риск ДТП на 9-31% (в зависимости от длины исходного маршрута) за счет увеличения средней длины маршрута на 6-11% и увеличения среднего числа проходимых вершин дорожного графа на 6-27,6%.

Глава 5. Обход опасных участков маршрута как способ повышения безопасности движения (на примере Санкт-Петербурга)

Введение

В Главе 4 на примере г. Москвы был опробован способ повышения безопасности дорожного движения, заключающийся в построении маршрута, обходящего препятствия, выявленные на дорожной карте (графе). Препятствиями служат ребра дорожного графа (сегменты дороги), содержащие статистически достоверно большое число дорожно-транспортных происшествий (ДТП). Каждое выявленное препятствие подвергается штрафу – добавлению некоторой величины к атрибуту длины соответствующего ребра дорожного графа, после чего запускается обычный алгоритм маршрутизации, определяющий маршрут минимальной (с учетом штрафов) длины.

В данной главе тот же подход применен к Санкт-Петербургу (см. [61]). Результаты расчетов показывают, что и в случае Санкт-Петербурга выделение и последующий обход препятствий позволяет существенно снизить средний относительный риск ДТП[26] за счет некоторого увеличения длины маршрута и числа проходимых вдоль маршрута вершин дорожного графа.

5.1 Данные

Как и в предыдущей главе, воспользуемся глобальной картой мира OpenStreetMap (OSM) для получения дорожной сети, на этот раз — города Санкт-Петербурга. Загрузить дорожную сеть проще всего с помощью библиотеки OSMnx для языка Python (см. [67] Полученная дорожная сеть представляет собой

[26] Под относительным риском ДТП (ОРДТП) понимается отношение числа ДТП вдоль маршрута, учитывающего препятствия, к числу ДТП, подсчитанных вдоль исходного, построенного без учета препятствий, маршрута.

мультидиграф, то есть, направленный граф, ребра которого могут быть одно- и двунаправленными, а через любые две вершины графа могут проходить сколько угодно ребер. Пакет OSMnx позволяет сохранить исходный граф в виде двух отдельных объектов – ребер и вершин в векторном формате .shp, что позволяет отобразить дорожную сеть в геоинформационной системе, например QGIS [120]

Таблица 5-1. Параметры дорожных сетей Москвы и Санкт-Петербурга

Показатели	Москва	Санкт-Петербург
Всего вершин	16797	14624
Всего ребер	25415	22787
Однонаправленных	15696	15222
Двунаправленных	9719	7565

Таблица 5-2. Описательная статистика длин (в метрах) ребер дорожного графа для Москвы и Санкт-Петербурга

Показатели	Москва	Санкт-Петербург
Средняя длина	217,4	155,1
Ст, отклонение	275,5	219,2
Минимум	1,06	0,66
25% процентиль	47,2	24,1
Медиана	127,9	81,9
75% процентиль	291,6	197,9
Максимум	8043	4542

Несмотря на значительно меньшую по сравнению с Москвой численность населения Санкт-Петербурга, его дорожная сеть, полученная из OSMnx, мало уступает, как это видно из Табл. 5-1, в сложности московской и насчитывает 30352 ребер[27] (35134 ребер для Москвы). Статистика длин ребер дорожного графа обоих городов, приведенная в Табл. 5-2, показывает, что дорожные сегменты (ребра дорожного графа) для Санкт-Петербурга в среднем короче московских (на 36%, если судить по медианам).

Что же касается данных о ДТП, то они получены из официального источника ГИБДД [121]. Для данного исследования были выбраны записи о 16865 серьезных (то есть, приведших к травме хотя бы одного из участников) ДТП, зафиксированных в Санкт-Петербурге в 2019-2021 годах. Поскольку дорожная сеть

[27] При этом каждое двунаправленное ребро рассматривается как два ребра

Санкт-Петербурга и данные о ДТП получены из независимых источников, необходимо удалить ДТП, явно не принадлежащие (например, из-за ошибки регистрации) дорожной сети. С этой целью были оставлены для дальнейшего анализа лишь те ДТП, чье расстояние до ближайшего ребра дорожной сети меньше 35м[28]. Всего таких ДТП оказалось 13503 (80% от общего числа).

5.2 Инструменты и метод

Как и в работе [119], для каждого ребра дорожного графа можно подсчитать число принадлежащих ему ДТП и затем найти ребра, в которых число ДТП превышает, скажем, 95-й процентиль значений, полученных в результате статистических испытаний, достаточно подробно описанных в [58]. Считая число ДТП таких ребер статистически значимым (на уровне 0,95), будем в дальнейшем использовать такие «опасные» ребра в качестве препятствий.

Для этого создадим граф G0, представляющий дорожную сеть Санкт-Петербурга, и его копию – граф G1. Граф G0 оставим без изменений, а в графе G1 добавим к атрибуту длины <Length> ребер, выбранных в результате статистических испытаний, достаточно большое число (штраф), одинаковое для всех ребер, чтобы показать алгоритму маршрутизации, что эти ребра нежелательны в прокладываемом маршруте и их по возможности следует избегать.

Нам осталось только сравнить все маршруты, проложенные по графам G0 и G1 и убедиться в том, что маршруты на графе G1 в некотором смысле безопасней. Для этого зададим квадратную сетку с фиксированным шагом, равным в нашем случае 3км. (Рис. 5-1), и построим все маршруты, начинающиеся и заканчивающиеся в вершинах графа, ближайших к узлам сетки.

[28] Если считать ширину полосы равной 3.5 м., а максимальное количество полос равным 10, то 35м. будет служить оценкой максимальной ширины полосы.

Рисунок 5-1. Узлы квадратной сетки, наложенные на дорожную карту Санкт-Петербурга

Обозначим два любых узла сетки как A и B и найдем маршрут A->B[29] на графе G0 и маршрут A—>B на графе G1. Для сравнения этих маршрутов подсчитаем относительный риск ДТП (ОРДТП), равный отношению ДТП, случившихся вдоль маршрута A—>B к числу ДТП, произошедших вдоль маршрута A->B. Если верхняя граница доверительного интервала для среднего ОРДТП < 1, то маршруты, проложенные по графу G1, более безопасны, чем маршруты, определенные на исходном графе G0.

[29] Маршрут A->B не эквивалентен B->A, поскольку может содержать однонаправленные ребра.

5.3 Результаты

Для статистических испытаний были выбраны первые 2000 ребер дорожного графа, содержащие наибольшее количество ДТП, поскольку расчеты для всех 30352 ребер крайне трудоемки[30] и по большому счету бессмысленны.

Рисунок 5-2. Препятствия, отображенные на дорожной карте Санкт-Петербурга

Всего для каждого ребра из 2000 было проведено 1000 испытаний, значимо «опасными» оказались на уровне 0,95 – 983 ребра, из которых, как и в работе [58], использовались первые 800 ребер, содержащих наибольшее число ДТП. Рис. 5-2 показывает 983 ребра-препятствия на карте Санкт-Петербурга. Ребра с различным числом ДТП показаны разным цветом (0-4 ДТП – зеленый, 4-7 ДТП – синий, 7-13

[30] Все расчеты производились на рабочей станции с четырехъядерным процессором и 32гб оперативной памяти

ДТП – красный, 13-18 ДТП - желтый). В таблице 5-3 приведен список 10 сегментов дороги (с координатами начала и конца сегмента) с наибольшим числом ДТП.

После получения препятствий порядок действий становится таким же, как и в [58]. Необходимо, пользуясь квадратной сеткой, наложенной на карту Санкт-Петербурга, перебрать все начальные и конечные точки сетки и для каждого маршрута A->B, проложенного без учета препятствий, вычислить маршрут A—>B, имеющий те же начальную и конечную точки, но проложенный с учетом препятствий. Далее следует сравнить все такие маршруты и понять, как в среднем меняется относительный риск ДТП, длина маршрута и число проходимых вершин. Этот анализ был проведен для различных значений штрафа, налагаемого на препятствие, чтобы понять, как он влияет на эффективность альтернативной маршрутизации.

Таблица 5-3. Дорожные сегменты в Санкт-Петербурге с наибольшим числом ДТП

Описание	Начало (широта, долгота)	Конец (широта, долгота)	ДТП
КАД 58-60 км. внутр.	59,8523, 30,4745	59,8352, 30,4484	18
КАД 81-83км. внешн.	59,8344, 30,4477	59,8524, 30,4753	18
КАД 74км	59,8114, 30,3423	59,8151, 30,3615	17
КАД 35-37км	60,0155, 30,4664	59,9931, 30,4798	16
Литейный проспект от ул. Пестеля до ул. Некрасова	59,9429, 30,3484	59,9390, 30,3482	15
Пр-т Энгельса от пр. Луначарского, до Выборгского шоссе	60,0438, 30,3279	60,0361, 30,322	14
Ленинский проспект от ул. Зины Портновой до бульвара Новаторов	59,8518, 30,2564	59,8518, 30,2678	14
Свердловская набережная от ул. Ватутина до ул. Арсенальной	59,9595, 30,3839	59,9549, 30,3746	14
Проспект Науки от Гражданского проспекта до ул. Бутлерова	60,0130, 30,3978	60,011, 30,4058	14
КАД 102км	59,9824, 30,5070	59,9866, 30,4904	14

Результаты расчетов показаны на рисунках 5-3…5-5 и в таблицах 5-4…5-6. Кроме самих значений в таблицах 5-4…5-6 приведены более мелким шрифтом, а на рисунках показаны цветом меньшей насыщенности 95% доверительные интервалы значений, полученные бутстрэп-методом. В качестве препятствий

использовались первые 800 значимых на уровне 0,95 ребер, а в качестве штрафа значения 200, 500, 1000, 2000, и 5000 метров.

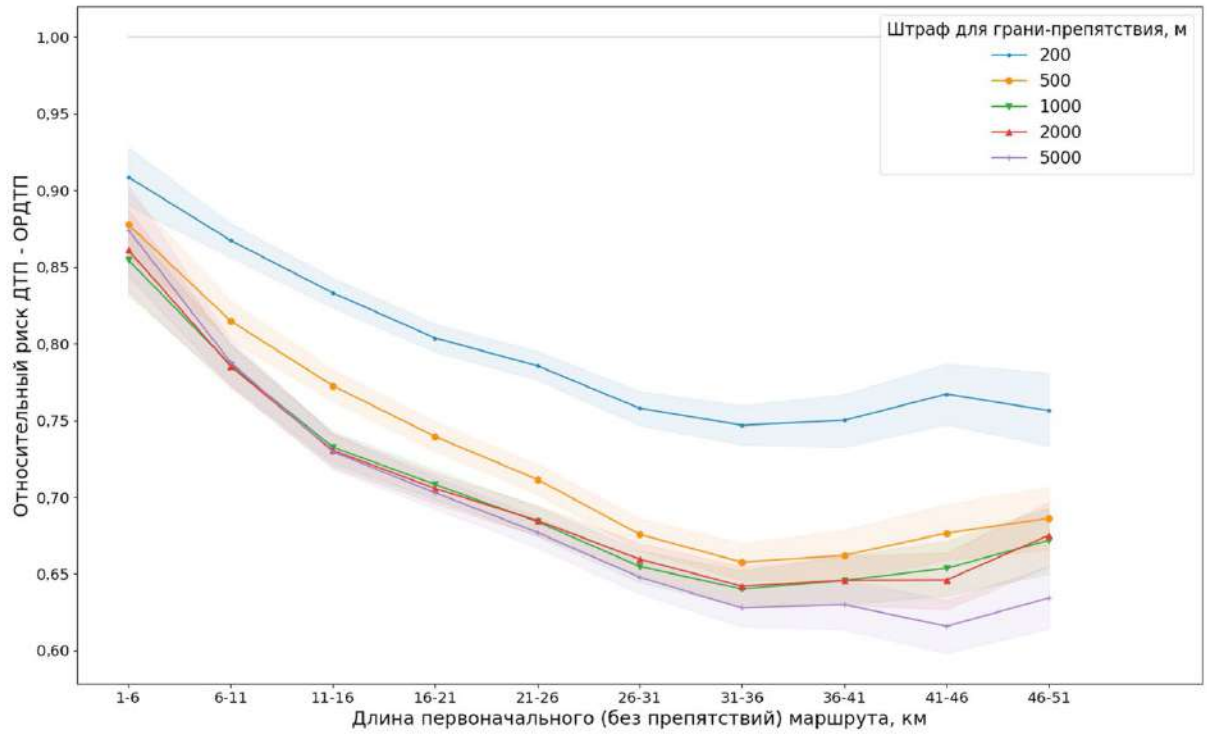

Рисунок 5-3. Относительный риск ДТП в зависимости от длины исходного маршрута и штрафа для ребра-препятствия с 95% доверительными интервалами

Если учитывать только относительный риск ДТП, то оптимальным будет штраф, равный 5000м (см. Рис. 5-3). Таблица 5-4 показывает, что минимальное значение риска при этом штрафе равно 0,616.

Но одновременно при штрафе 5000м (см. рисунки 5-4 и 5-5) сильно возрастет проигрыш в длине маршрута и числе проходимых вершин дорожного графа.

Как показывают Рис. 5-3 и таблица 5-4, штрафы равные 1000 м и 2000 м дают практически одинаковую зависимость среднего относительного риска от длины первоначального маршрута. Но при этом, как это следует из рисунков 5-4 и 5-5, для штрафа 1000 м значительно уменьшается проигрыш в длине маршрута и числе проходимых вершин дорожного графа. Поэтому штраф 1000 м следует считать оптимальным.

Таблица 5-4. Относительный риск ДТП в зависимости от длины исходного маршрута и штрафа для ребра-препятствия с 95% доверительными интервалами

Интервал км	Число маршрутов	200 м	500 м	1000 м	2000 м	5000 м
1-6	297	0,909 0,890-0,928	0,878 0,857-0,899	0,855 0,831-0,879	0,861 0,834-0,888	0,874 0,844-0,904
6-11	853	0,867 0,857-0,878	0,815 0,803-0,828	0,786 0,773-0,800	0,785 0,771-0,799	0,788 0,773-0,803
11-16	1231	0,833 0,824-0,843	0,773 0,763-0,783	0,733 0,722-0,743	0,730 0,720-0,742	0,730 0,718-0,741
16-21	1272	0,804 0,795-0,813	0,739 0,730-0,749	0,708 0,699-0,718	0,706 0,696-0,715	0,703 0,692-0,713
21-26	1088	0,786 0,776-0,795	0,712 0,702-0,721	0,684 0,674-0,694	0,685 0,675-0,694	0,677 0,667-0,687
26-31	768	0,758 0,747-0,769	0,676 0,665-0,686	0,655 0,645-0,665	0,660 0,649-0,670	0,648 0,637-0,658
31-36	493	0,747 0,734-0,760	0,658 0,646-0,670	0,640 0,628-0,652	0,642 0,630-0,654	0,628 0,615-0,640
36-41	233	0,750 0,733-0,768	0,662 0,646-0,678	0,646 0,630-0,661	0,646 0,630-0,662	0,630 0,613-0,646
41-46	97	0,767 0,748-0,787	0,677 0,659-0,695	0,654 0,636-0,672	0,646 0,628-0,664	0,616 0,598-0,633
46-51	53	0,757 0,733-0,780	0,686 0,666-0,707	0,672 0,651-0,694	0,675 0,654-0,696	0,634 0,614-0,655
Доля измененных маршрутов		0,82	0,88	0,92	0,94	0,95

Для этого значения штрафа, согласно таблице 5-5, среднее отношений длин маршрута колеблется в пределах 1,08 – 1,1, то есть (см. таблицу 5-5,) *среднее уменьшение относительного риска на 14,5-36% достигается при штрафе 1000м за счет увеличения средней длины маршрута на 8-10%.* Уменьшение относительного риска сопровождается, как это следует из Рис. 5-6 и таблицы 5-6, еще одной потерей: увеличением числа проходимых вершин дорожного графа примерно на 3-12% (для штрафа 1000м) в зависимости от протяженности исходного маршрута.

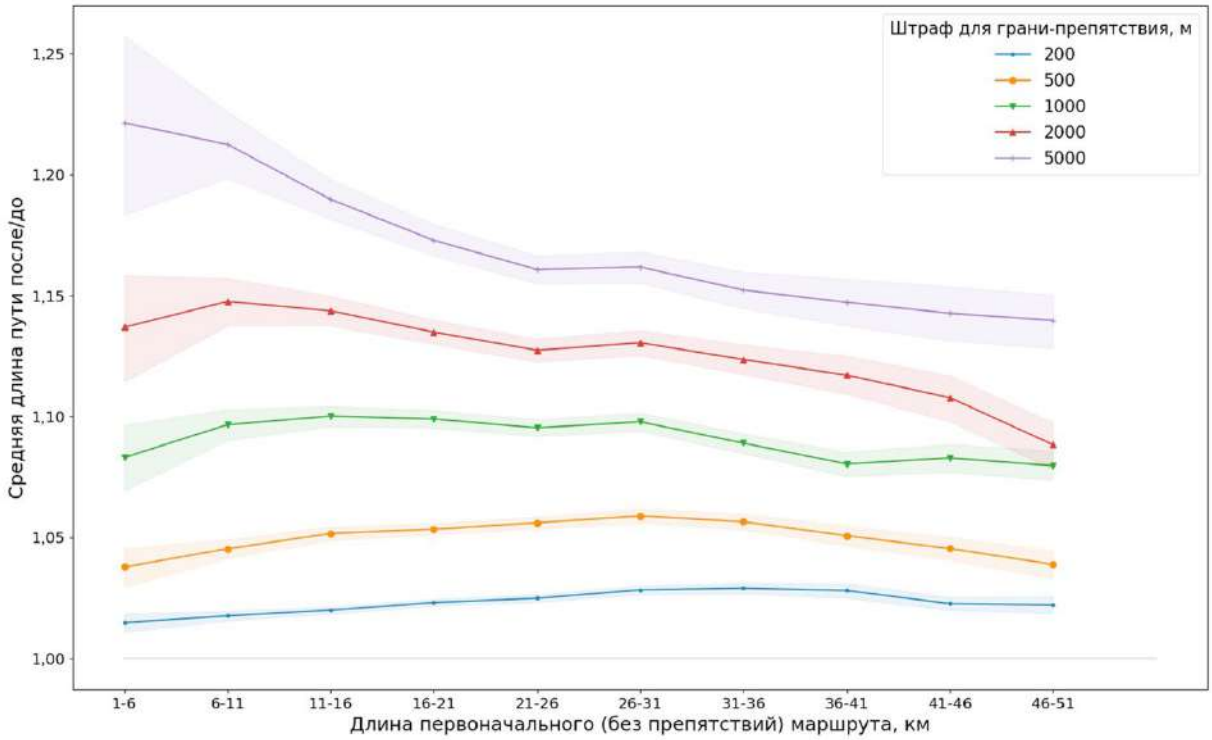

Рисунок 5-4. Зависимость средних отношений длин маршрутов (Длина при обходе УПО/ Первоначальная длина) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия

Таблица 5-5. Средние отношения длин маршрутов (после/до) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия (с 95% доверительными интервалами)

Интервал км	Число маршрутов	200м	500 м	1000 м	2000 м	5000 м
1-6	297	1,015 1,011-1,018	1,038 1,030-1,045	1,083 1,070-1,096	1,137 1,115-1,158	1,221 1,183-1,257
6-11	853	1,018 1,016-1,019	1,045 1,041-1,049	1,097 1,090-1,103	1,148 1,138-1,157	1,213 1,199-1,226
11-16	1231	1,020 1,018-1,021	1,052 1,049-1,054	1,100 1,096-1,104	1,144 1,138-1,150	1,190 1,182-1,198
16-21	1272	1,023 1,022-1,025	1,053 1,051-1,056	1,099 1,095-1,103	1,135 1,130-1,140	1,173 1,167-1,179
21-26	1088	1,025 1,023-1,026	1,056 1,054-1,059	1,095 1,092-1,099	1,127 1,123-1,132	1,161 1,155-1,166
26-31	768	1,028 1,026-1,030	1,059 1,056-1,062	1,098 1,094-1,101	1,130 1,125-1,136	1,162 1,155-1,168
31-36	493	1,029 1,027-1,031	1,056 1,053-1,060	1,089 1,085-1,093	1,124 1,117-1,130	1,152 1,145-1,160
36-41	233	1,028 1,025-1,031	1,051 1,047-1,055	1,080 1,075-1,085	1,117 1,109-1,125	1,147 1,138-1,157
41-46	97	1,023 1,020-1,025	1,045 1,041-1,050	1,083 1,077-1,089	1,108 1,098-1,117	1,143 1,131-1,154
46-51	53	1,022 1,019-1,026	1,039 1,033-1,045	1,080 1,074-1,086	1,088 1,078-1,097	1,140 1,128-1,150
Доля измененных маршрутов		0,82	0,88	0,92	0,94	0,95

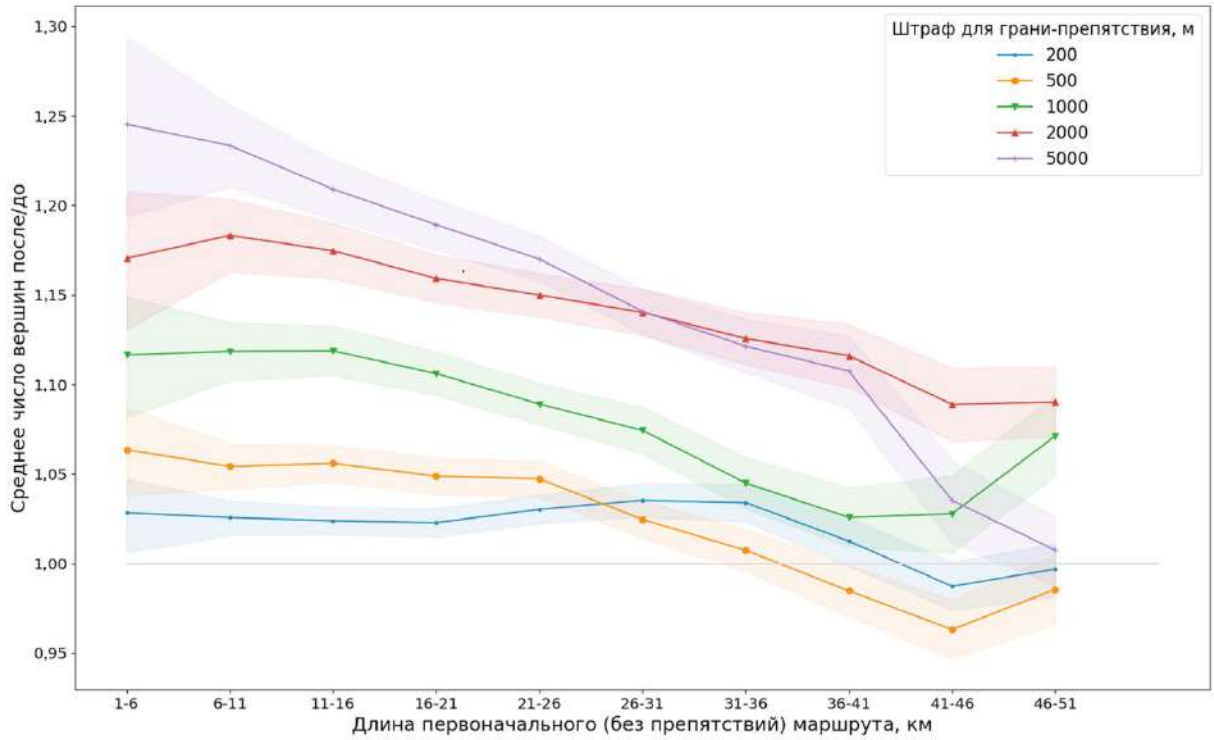

Рисунок 5-5. Средние отношения числа вершин маршрутов (после/до) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия (с 95% доверительными интервалами)

Таблица 5-6. Средние отношения числа вершин маршрутов (после/до) для различной длины оригинального маршрута и различных значений штрафа для ребра-препятствия (с 95% доверительными интервалами)

Интервал км	Число маршрутов	200м	500 м	1000 м	2000 м	5000 м
1-6	297	1,028 1,007-1,047	1,064 1,038-1,088	1,116 1,081-1,149	1,171 1,129-1,208	1,245 1,193-1,296
6-11	853	1,026 1,016-1,035	1,054 1,041-1,067	1,118 1,102-1,134	1,183 1,163-1,203	1,233 1,210-1,256
11-16	1231	1,024 1,016-1,032	1,056 1,046-1,066	1,119 1,105-1,132	1,175 1,159-1,190	1,209 1,192-1,226
16-21	1272	1,023 1,015-1,031	1,049 1,039-1,059	1,106 1,094-1,118	1,159 1,146-1,173	1,189 1,175-1,203
21-26	1088	1,030 1,022-1,038	1,047 1,037-1,057	1,089 1,077-1,101	1,150 1,138-1,162	1,170 1,158-1,183
26-31	768	1,035 1,026-1,045	1,025 1,014-1,036	1,074 1,061-1,087	1,140 1,127-1,153	1,141 1,128-1,154
31-36	493	1,034 1,023-1,045	1,007 0,996-1,019	1,045 1,031-1,059	1,126 1,111-1,140	1,121 1,106-1,136
36-41	233	1,012 0,999-1,026	0,985 0,971-0,999	1,026 1,009-1,042	1,116 1,098-1,133	1,107 1,086-1,128
41-46	97	0,987 0,973-1,001	0,963 0,947-0,980	1,028 1,006-1,050	1,089 1,069-1,109	1,035 1,012-1,058
46-51	53	0,997 0,982-1,011	0,985 0,966-1,004	1,071 1,049-1,094	1,090 1,071-1,110	1,007 0,988-1,026
Доля измененных маршрутов		0,82	0,88	0,92	0,94	0,95

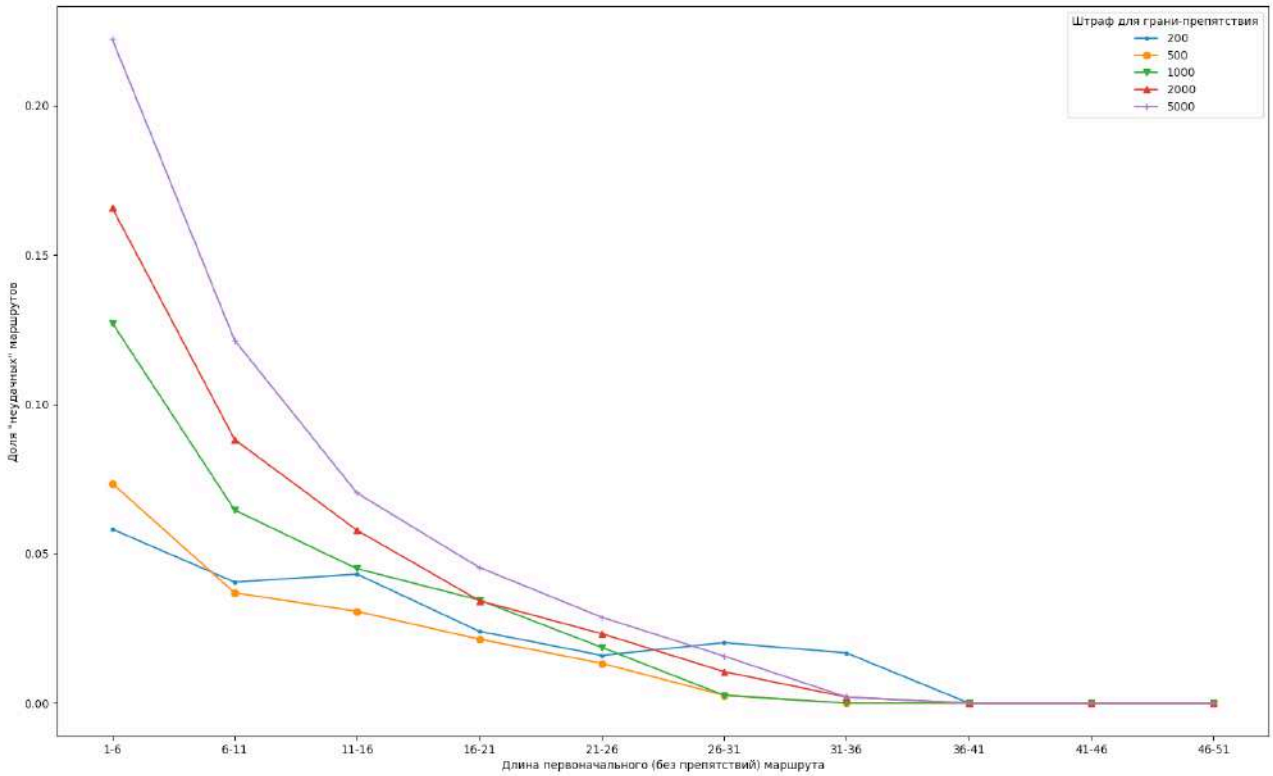

Рисунок 5-6. Доля маршрутов, для которых не достигается выигрыш в безопасности, в зависимости от длины оригинального маршрута и штрафа

Рисунки 5-3…5-5 и таблицы 5-4…5-6 дают значения средних величин (среднего относительного риска, среднего удлинения маршрута и т.д.). Но конкретный маршрут, рассчитанный с учетом препятствий, может и не отвечать эти средним значениям. В частности, маршрут может не давать выигрыша в безопасности по сравнению с маршрутом, проложенным по исходной (без введенных препятствий) дорожной сети. Рисунок 5-6 показывает долю таких маршрутов[31], как обычно, в зависимости от длины оригинального маршрута.

Из Рис. 5-6 хорошо видно, что доля неудачных маршрутов максимальна при малых длинах оригинального маршрута и достигает 0.23 для самого большого значения штрафа (5000м). Для оптимального значения штрафа доля неудачных маршрутов колеблется от 12.7% (для самых коротких, длиной меньше 6 км маршрутов) до 0% (для маршрутов длиннее 31 км) Это значит, что для

[31] Учитываются только *различные* маршруты, имеющие общие начало и конец.

практических расчетов необходимо проверять каждый вычисленный маршрут и если он не получается более безопасным, использовать другое значение штрафа или вовсе отказываться от альтернативной маршрутизации.

Выводы

Обход препятствий [32], которыми служат ребра дорожного графа г. Санкт-Петербурга, содержащие статистически достоверно большое число дорожно-транспортных происшествий (ДТП), позволяет снизить относительный риск ДТП на 14,5-36% (в зависимости от длины исходного маршрута) за счет увеличения средней длины маршрута на 8-10% и увеличения среднего числа проходимых вершин дорожного графа на 3-12%.

Получение сходных результатов для Санкт-Петербурга путем повторения алгоритма, ранее примененного для Москвы [58], позволяет говорить об устойчивости предложенного метода и возможности его использования и для других городов.

[32] Имеется в виду, что на каждое препятствие налагается штраф 1000м.

Глава 6. Пакет программ, модифицирующих дорожный граф в соответствии с принадлежащим ему числом ДТП

6.1 Введение

В данной главе будет подробно описан пакет программ на языке Python, позволяющий модифицировать дорожный граф таким образом, что применение к нему обычных алгоритмов маршрутизации приведет к построению в среднем более безопасных маршрутов, как это показано в главах 4, 5.
Построение модифицированного дорожного графа представляет собой довольно сложную задачу (как алгоритмически, так и в смысле объема вычислений). Нам представляется, что эту задачу удобнее решать с помощью набора последовательно применяемых небольших программ, чем с помощью одной большой программы. К тому же, мы для решения задачи прибегаем к сторонним программам (SANET), которые позволяют генерировать точки, равномерно распределенные по дорожной сети, необходимые для проведения статистических испытаний (см. 4.2).

6.2 Получение дорожного графа в формате OSMNX и (отдельно) ребер и вершин дорожной сети в формате ShapeFile

В Приложении 1 показана программа, позволяющая загрузить из депозитория OpenStreetMap векторную карту (дорожный граф) выбранного города. В Приложении 1 это Санкт-Петербург. Процессом загрузки управляет функция ox.config( ), где самая важная опция - all_oneway = False означает, что двунаправленное ребро дорожного графа сохраняется как одно ребро с атрибутом oneway = 1 (а однонаправленное ребро — с атрибутом oneway = 0).

Сама загрузка графа выполняется методом ox.graph_from_place(), два следующих метода ox.add_edge_speeds(G) и ox.add_edge_travel_times(G) загружают оценку максимальной скорости и минимального времени проезда через ребро графа (длина ребра просто делится на максимальную скорость) соответственно. Остаток программы сохраняет исходный граф и его представления в виде наборов ребер и вершин (в виде файлов в формате ShapeFile) на диске.

6.3. Объединение ДТП и дорожного графа

По определению каждое ДТП должно принадлежать дорожному графу, то есть, располагаться на одном и только одном его ребре. Но поскольку дорожный граф и ДТП получаются из разных источников, они не совпадают друг с другом. Значит, нам придется для каждого ДТП определить ближайшее к нему ребро дорожного графа и далее считать, что именно на этом ребре дорожного графа ДТП и произошло. При этом в случае двунаправленного ребра возникает трудность, связанная с невозможностью определить, какому направлению ребра принадлежит ДТП. В принципе это можно выяснить, рассмотрев метаданные ДТП, в которых указывается улица и дом, вблизи которого произошло ДТП, но алгоритмизировать этот процесс весьма трудно, а перебрать вручную несколько тысяч ДТП и вовсе невозможно. Поэтому там, где это необходимо, нам приходится «расщеплять» ДТП, присваивая его половину одному направлению, а половину — другому. Программа на языке Python, определяющая ближайшеее к ДТП ребро дорожного графа, показана в Приложении 2. «Сердце» программы – это строка

u, v, key, dist = ox.get_nearest_edge(Gp, (y, x), return_dist=True)

в которой для ДТП с координатами (y,x) вычисляется четверка значений – (u,v,key) – три значения, опредяляющие ближайшее к ДТП ребро, где u –

идентификатор начала ребра (= идентификатор вершины графа, располагающейся в гначале ребра), v - идентификатор конца ребра, а key – целое число, по которому определяется одно из ребер, проходящих через две вершины (u,v). Если через заданные вершины проходит только одно ребро, key имеет единственное значение 0. Если же через данные вершины проходит два ребра, key имеет два значения – 0,1 и т.д.

Обратим внимание на то, что метод ox.get_nearest_edge() имеет в качестве параметров спроектированный (projected) граф Gp и координаты (x,y), а не ожидаемые широту и долготу (lat, lon). Это делается для того, чтобы вычислять расстояния между ДТП и ребром дорожного графа, поскольку в случае, когда ДТП определяется широтой и долготой, этого сделать нельзя. Для определения расстояний в данной работе используется координатная система UTM (Universal Transverse Mercator) (см, например, [62]). Преимущество системы UTM состоит в том, что расстояние между двумя точками можно вычислять с высокой точностью с помощью обычной евклидовой формулы d = sqrt((x1-x2)**2 + (y1-y2)**2)

В Приложении 3 дается программа, которая оценивает точность вычисления расстояний между двумя точками в системе UTM. В качестве двух точек взяты координаты (широта и долгота) крайней северной и крайней южной точки Санкт-Петербурга. Вычисленное с учетом кривизны поверхности земли в строке

dist = geopy.distance.geodesic(coords_1, coords_2).m ,

расстояние равно 31879 м. Вычисления показывают, что разница между «истинным» расстоянием и расстоянием, вычисленным по формуле Евклида с помощью UTM-координат, равна всего 4m. Поскольку нас интересуют расстояния < 35м, точность вычисления расстояний можно считать вполне удовлетворительной.

Переход к координатам UTM для дорожного графа (см. Приложение 2) производит метод project_graph():

```
Gp = ox.project_graph(G)
```

А проекция точки ДТП — методом .projection.project_geometry

point_geom_proj, crs = ox.projection.project_geometry(Point(reversed(point)),\
to_crs=Gp.graph['crs']) ,

где point — это широта и долгота, а crs – это зона UTM, равная для Петербурга 36.

Нам осталось описать вход и выход программы. На входе имеем список ДТП в формате .csv:

```
YEAR,DATE,TIME,LAT,LON,X,Y
19,31.01.2019,14:50,59.896364,29.842107,658989.2947785741,6643282.0244597355
19,31.01.2019,11:20,59.752223,30.288019,347630.30501833995,6626933.342702196
19,31.01.2019,22:35,59.884464,30.387443,353794.9774347632,6641428.438857373
19,31.01.2019,12:10,60.004061,30.436099,357032.93212231743,6654634.490192406
```

Каждая строка, как это видно из заголовка файла, содержит описание одного ДТП: год (последние две цифры), дату, время, широту и долготу, а также координаты x,y, происхождение которых не было понятно из данных, предоставленных ГИБДД, поэтому было принято решение перейти в систему UTM.

На выходе программы имеем файл в формате .csv:

```
YEAR,DATE,TIME,ID,X,Y,LAT,LON,NODE1,NODE2,KEY, DIST
9, 31.01.2019, 11:20, 126439, 347630.30501808145, 6626933.341987754, 59.752223,
30.288019, 800275641, 1038407432, 0, 2.23623
19, 31.01.2019, 11:20, 126439, 347630.30501808145, 6626933.341987754, 59.752223,
30.288019, 1038407432, 800275641, 0, 2.23623
19, 31.01.2019, 22:35, 126440, 353794.97743462917, 6641428.438138614, 59.884464,
30.387443, 1710211567, 300429771, 0, 0.72741
19, 31.01.2019, 12:10, 126441, 357032.93212221784, 6654634.489469756, 60.004061,
30.436099, 25896964, 276609, 0, 1.30547
```

Это тоже список ДТП, но к каждому ДТП теперь добавлены 4 параметра: (NODE1, NODE2, KEY) – идентификатор ближайшего ребра и DIST – расстояние ДТП до ближайшего ребра.

6.4. Подсчет числа ДТП для каждого ребра дорожного графа

В предыдущем пункте мы перебирали все зарегистрированные ДТП и для каждого определяли ребро графа, которому ДТП принадлежит. В результате

получился список, в котором для каждого ДТП указаны идентификаторы ребра, которому ДТП принадлежит.

Ничто не мешает проделать обратную операцию: перебирать все ребра графа и искать каждое ребро в файле, где перечислены все ДТП. Сколько раз будет найдено в этом файле ребро графа, столько ДТП и принадлежит этому ребру. Программа, подсчитывающее число ДТП для каждой грани, приведена в Приложении 4.

На входе программы используются полученные ранее дорожный граф G и список ДТП с идентификаторами ребер, которым ДТП принадлежат: (tas_to_edges_2019_2021_twoway_proj_spb_gibdd_flt_35.csv). Ясно, что искать какие-то строки непосредственно в .csv файле бессмысленно. Поэтому в программе .csv файл преобразуется в объект Pandas с помощью метода pd.read_csv()

```
ta_edges =
pd.read_csv('data/tas_to_edges_2019_2021_twoway_proj_spb_gibdd_flt_35.csv')
```

После того как сведения о ДТП попали в объект Pandas подсчет ДТП для каждого ребра сводится к поиску, реализованному в основной функции программы tas_along_edge():

```
def tas_along_edge(u, v,  key):
    edges_outg = ta_edges.loc[ta_edges['NODE1'] == u]
    edge_tas = edges_outg.loc[(edges_outg['NODE2'] == v) &\
                              (edges_outg['KEY'] == key)]
    tas = edge_tas.shape[0]
    oneway = G.edges[u, v, key]['oneway']
    if not oneway:
       tas = tas/2.0
    return tas
```

Поиск реализован в первых трех строках функции. Остальные строки нужны, чтобы проверить ребро на двунаправленность. Если ребро двунаправленное, нужно распределить найденные ДТП по двум напрвлениям. Мы уже говорили в пункте 6.3, что приписать ДТП какому-то направлению крайне сложно, если вообще возможно, используя реальные данные о ДТП. Поэтому в программе мы просто приписываем каждому ребру половину найденных ДТП. Это не должно

сильно повлиять на результаты маршрутизации, поскольку двунаправленные ребра, как это следует из таблицы 5-1, составляют около трети всех ребер, а для длинных маршрутов (наиболее важных с точки зрения безопасности), включающих в себя десятки ребер, должно произойти усреднение ошибок, связанных с отнесением ДТП к неверному направлению двунаправленного ребра.

Нам осталось упомянуть только некоторые особенности программы. Функция collect_edges() готовит список всех ребер дорожного графа. Казалось бы, этот список можно получить из атрибута edges объекта G: G.edges(keys = True, data = True), но оказывается, что некоторые ребра в этом первоначальном списке повторяются, и функция collect_edges() просто фильтрует его.

Основная функция программы tas_to_edges(G) перебирает все ребра из списка edge_list[] и для каждого ребра вычисляет число ДТП, плотность ДТП, равную числу ДТП поделенному на длину ребра, а также координаты начала и конца ребра. Далее список ребер сортируется, так что ребра с самым большим числом ДТП оказываются в списке первыми. Результат расчетов записывается в файл edges_most_populated_spb_sorted_by_ta_flt_35_debug.csv, первые строки которого выглядят так, как в листинге 6.4.1.

Листинг 6.4.1. Файл, содержащий отсортированный список ребер дорожного графа, число и плотность ДТП для каждого ребра

```
cn,node_start,node_end,key,lat0,lon0,lat1,lon1,tas,ta_density
0,10593917,10593888,0,59.8523445,30.4745321,59.835247,30.4484255,18,0.007
1,339921466,248196221,0,59.834425,30.4477147,59.8523655,30.4752937,18,0.007
2,251397970,251398032,0,59.9159712,30.4137958,59.8928928,30.4469675,17,0.005
3,2490216682,187587141,0,59.8113625,30.3422923,59.8151242,30.361476,17,0.014
4,587052882,245828,0,60.0154892,30.4663785,59.9930789,30.4798397,16,0.006
5,9708925354,9708927324,0,59.9428586,30.3484161,59.9390119,30.3481549,15,0.035
```

Заголовок файла показывает, что в строке находятся 10 параметров: номер ребра, его идентификаторы (node_start,node_end,key), координаты начала и конца, число ДТП (tas) и плотность ДТП (ta_density).

6.5 Статистические испытания для определения ребер со статистически значимым числом ДТП

Будем считать статистически значимым такое значение числа ДТП для данной грани, которое превышает 95-ю или 98-процентиль распределения ДТП для этой грани — при условии, что ДТП распределены равномерно по дорожной сети.
Для получения точек, равномерно распределенных по дорожной сети, в данной работе используется программный пакет SANET [47], позволяющий получить (в формате ShapeFile) миллионы точек, равномерно распределенных по дорожной сети. Чтобы получить эти точки, достаточно иметь ребра дорожного графа в формате ShapeFile (их получение кратко описано в разделе 6.2).
Сами статистические испытания выполняет программа, показанная в Приложении 5. В цикле while True: сначала читается построчно список ребер, отсортированный по числу принадлежащих им ДТП, откуда извлекается идентификатор ребра (edge1, edge2, key) и число ДТП (tas):

```
if edge_cnt > tot_edges: break;
      line = ta_edges.readline()
      vals = line.split(',')
      if len(vals) < 10: break
      vals[-1] = vals[-1].strip() #delete last symbol '\n'
      edge1 = int(vals[hp.index('node_start')])
      edge2 = int(vals[hp.index('node_end')])
      key   = int(vals[hp.index('key')])
      tas = vals[hp.index('tas')]
```

Далее из общего списка списков равномерно распределенных точек sanet_trials извлекаются точки для конкретного испытания trials и среди этих точек ищется ребро (edge1, edge2, key). Сколько раз найдется ребро — столько точек ему и принадлежит. Всего для данного ребра проводится 1000 испытаний, их результат записывается в файле sanet_trials_file следующим образом:

```
node_start,node_end,key,tas,trial_values
10593917,10593888,0,18,15#14#9#7#12#8#10#6#8#6#7#13#9#15#10#8#8#14#8#8#12#7#9#6…
```

Сначала идет идентификатор ребра, для которого проводятся испытания (10593917,10593888,0), затем число ДТП для этого ребра, и далее — сами результаты испытаний (1000 значений, разделенных знаком '#').
Нам осталось упомянуть вспомогательную функцию prepare_sanet_points(trials, size), которая делит общий список сгенерированных равномерно распределенных точек на порции размером равные реальному числу ДТП. Всего для 1000 испытаний требуется 1000 таких порций.

6.6 Выделение ребер со статистически значимым числом ДТП

Будем считать число ДТП, принадлежащих ребру дорожного графа, значимым, если оно превышает 95-ю процентиль значений, полученную в результате статистических испытаний. Это значит, что имея на входе файл, полученный в пункте 6.5, необходимо читать его построчно, извлекая из каждой строки число ДТП и вычисляя 95-ю процентиль статистических испытаний. Далее нужно сравнить число ДТП и процентиль, и если число ДТП больше, записать соответствующу строку в выходной файл, который и будет содержать ребра с достоверно высокими числами ДТП. Программа, выполняющая действия, описанные выше, приведена в Приложении 7.
В основном цикле программы while Tue: совершаются описанные выше действия. Процентиль значений статистических испытаний находится с помощью функции percentile(vals, p), имеющей в качестве параметра значения статистических испытаний vals и процентиль, например, 95.

6.7 Построение маршрутов на модифицированном графе и сбор статистики относительного риска ДТП

Наконец мы готовы модифицировать дорожный граф таким образом, что ребра со статистически значимым высоким числом ДТП получат «штраф» - одинаковую прибавку в длине каждого ребра. Но эта прибавка может и не приводить к построению более безопасного маршрута. Чтобы убедиться, что маршрут из пункта А в пункт Б получился более безопасным, вычислим отношение ДТП по модифицированному маршруту/ДТП по оригинальному (на немодифицированном графе) маршруту (назовем его относительным риском ДТП). Если это отношение меньше единицы, маршрут, построенный по модифицированному графу, более безопасен.

Естественно, для определения оптимального значения штрафа необходимо собрать статистику по некоему множеству маршрутов и убедиться в том, что среднее значение относительного риска ДТП значимо меньше единицы.

Для задания множества маршрутов будем использовать квадратную сетку, наложенную на дорожный граф, а для получения статистики относительного риска проложим маршруты между всеми различными узлами решетки.

Чтобы создать решетку, будем использовать программу, показанную в Приложении 8. Эта программа, несмотря на приличный объем, весьма проста. В начале задаются координаты левого верхнего угла решетки ilat, ilon, ширина и высота решетки width, height. Далее широта и долгота начала решетки переводятся в систему UTM, которой мы кратко коснулись в разделе 6.3:

ini_coords = utm.from_latlon(ilat, ilon)

Затем координаты начала корректируются (переносятся на 3км влево и 4,5 км вверх). И наконец, вполне понятным образом вычисляются узлы решетки, которые сохраняются в файле. Далее этот файл открывается, чтобы профильтровать узлы решетки – оставить только те узлы, которые расположены не далее половины шага решетки от ближайшей вершины дорожного графа. Это делается для того, чтобы узлы решетки были насположены на самом дорожном

графе. Узлы, находящиеся далеко от ближайшей дороги, для нас бесполезны. Решетка, созданная для Санкт-Петербурга, показана на Рис. 5.1.

Теперь, когда решетка создана и записана в файл, можно приступить к основной задаче – собиранию статистики относительного риска ДТП. Делает это программа, показанная в Приложении 9.

В отличие от других программ, эта запускается из командной строки. Дело в том, что программа выполняет большой объем вычислений и работает довольно долго. Чтобы ускорить расчеты, можно запустить несколько копий программы с разными параметрами из командной строки. Если, скажем, в процессоре 4 ядра, то можно без проблем запустить сразу 3 копии программы с разными параметрами.

Сама программа сначала загружает необходимые файлы (две копии дорожного графа — G и GO, квадратную решетку и список ребер со статистически значимым высоким числом ДТП), а затем модифицирует граф GO таким образом, что грани с «большим» ДТП получают штраф – увеличение длины на величину, вводимую в командной строке. Делается это в функции set_obstacles(fname, penalty, maxp), где fname — имя файла со списком ребер, содержащих статистически значимое число ДТП (см. раздел 6.6), penalty – значение штрафа в метрах, а maxp — число ребер, которое участвует в модификации графа GO, то есть, программа позволяет использовать либо все ребра со значимым числом ДТП или же их часть.

Далее следует основной цикл программы

```
for rin,rend in product(range(grd_pnt), range(grd_pnt)):
```

в котором перебираются все неидентичные пары узлов решетки.

Для каждой пары узлов вычисляются ближайшие к ним вершины графа:

```
start_node = ox.get_nearest_node(G, start)
end_node = ox.get_nearest_node(G, end) ,
```

где start, end – координаты соответствующих узлов решетки.

Для GO и G строятся маршруты, имеющие общее начало (start_node) и общее завершение (end_node), и для каждой пары маршрутов вычисляется относительный риск ДТП, равный отношению числа ДТП вдоль

модифицированного маршрута, проложенного по графу GO, к числу ДТП вдоль оригинального маршрута, проложенного по графу G. Подсчет ДТП вдоль маршрута выполняет функция tas_along_route(G, route), где route – список вершин графа, проходимых вдоль маршрута.

Кроме того, для каждого маршрута вычисляется его длина и число проходимых вдоль маршрута вершин дорожного графа.

Результаты расчетов записываются в .csv файл со следующей структурой:

Листинг 6.7.1 Данные о маршрутах

```
BEGIN,END,ROUTE_LENGTH,ROUTE_O_LENGTH,ROUTE_NODES,ROUTE_O_NODES,TA_DIRECT,TA_AVOID
,AFFECTED
60.035373133218066#30.08738874095845, 60.0084702205044#30.089756684912672,
4734.367, 4734.367, 10, 10, 0.0, 0.0, no
60.035373133218066#30.08738874095845, 59.873953156252696#30.10152938432629,
40672.12399999999, 40781.512999999984, 186, 186, 163.0, 155.5, yes
60.035373133218066#30.08738874095845, 59.847049243899896#30.10387060306932,
40015.45499999998, 40124.843999999975, 180, 180, 162.5, 155.0, yes
60.035373133218066#30.08738874095845, 59.82014516525566#30.10620740887939,
44565.799000000006, 44629.78200000001, 228, 223, 165.5, 160.0, yes
60.035373133218066#30.08738874095845, 60.03654801588501#30.141163834961077,
5827.031, 5827.031, 13, 13, 2.0, 2.0, no
```

Здесь BEGIN, END — координаты начала и конца маршрута, ROUTE_LENGTH– длина немодифициованного маршрута, ROUTE_O_LENGTH – длина модифицированного маршрута, ROUTE_NODES и ROUTE_O_NODES – число вершин графа вдоль оригинального и модифицированного маршрута соответственно, TA_DIRECT – число ДТП вдоль оригинального маршрута, TA_AVOID – число ДТП вдоль модифицированного маршрута, AFFECTED равно yes, если маршрут изменился, no — если нет.

6.8. Окончательная обработка и визуализация данных

Имея несколько файлов для различных значений штрафа (их структура показана в листинге 6.7.1), можно построить графики основных показателей маршрутизации (относительного риска ДТП, отношения длин модифицированного и исходного маршрута, а также отношения числа вершин

дорожного графа для модифицированного и исходного маршрута) в зависимости от длины первоначального, построенного без модификации графа G, маршрута. На таком графике необходимо показать не только средние значения, но и доверительные интервалы для средних. Все эти задачи выполняет программа, показанная в Приложении 10.

Основная функция plot_dependency(ylabel, column, label, legend): получает на входе ylabel – надпись на оси ординат, column – отображаемые значения, например, массив отношений Длина модифицированного маршрута/ Длина исходного маршрута, label – список штрафов (в виде строк), и legend – заголовок пояснительной надписи.

Для отображения данных, необходимо подсчитать их средние и доверительные интервалы средних. Это делает функция interval_stat(), которая в свою очередь использует bootstrap пакет bootstrapped. Подробнее об использовании бутстреп-оценок см. раздел 3.3.

Остальная часть программы не слишком сложна и не представляет большого интереса, поскольку подготавливает данные для функции plot_dependency(ylabel). Примеры графиков, построенных данной программой, показаны на Рис. 5.3, 5.4, 5.5.

Глядя на построенные графики, изучая соответствующие таблицы, как это описано в главах 4-5, можно найти оптимальное значение штрафа, что в паре со списком ребер со статистически значимым числом ДТП (см. раздел 6.6) дает полную информацию, необходимую для модификации дорожного графа. Для модификации подходит функция set_obstacles(fname, penalty, maxp), кратко описанная в разделе 6.7, которую нужно использовать с оптимальным для данного графа значением penalty.

Заключение (основные научные результаты)

1. В качестве показателя эффективности маршрутизации используется относительный риск ДТП, равный среднему отношению числа ДТП вдоль измененного (с учетом препятствий на дорогах) маршрута к числу ДТП вдоль не измененного (не учитывающего препятствия) маршрута ([57], стр. 8 абзац 6)
2. Статистически значимые кластеры ДТП используются как препятствия при построении альтернативного (более безопасного) маршрута ([48], стр. 47 абзац 6, [57], стр. 8 абзац 7)
3. В качестве препятствий используются отдельные сегменты дороги (ребра дорожного графа), содержащие число ДТП, статистически превышающее число ДТП, полученное из предположения о равномерности распределения ДТП по дорожной сети ([58] стр. 103 абзац 2, стр. 104 абзац 6, [61] стр. 30 абзац 1, стр. 32 абзац 2)
4. Для построения более безопасного маршрута каждое препятствие подвергается штрафу, т.е. к атрибуту длины соответствующего ребра прибавляется фиксированное число. Путем расчетов определяется оптимальное значение штрафа, которое обеспечивает минимальный риск ДТП при минимальном проигрыше в длине маршрута и числе проходимых вершин дорожного графа (([58], стр. 103 абзац 2, [61], стр. 32 абзац 3).

Благодарности

Автор признателен Терехову А.Н. за научное руководство, поддержку и помощь при работе над диссертацией. Автор благодарен своим родителям Михаилу и Марии, жене Галине и детям: Ариель, Оскару и Алику за постоянную помощь и поддержку.

Словарь терминов

Бутстрэп — метод оценки статистических характеристик, основанный на многократной генерации выборок методом Монте-Карло на базе имеющейся выборки

DBSCAN — алгоритм кластеризации, выделяющий скопления точек произвольной формы и заданной локальной плотности.

ГИС – геоинформационная система — система сбора, хранения, анализа и графической визуализации пространственных (географических) данных

Доверительный интервал — интервал значений случайной переменной, в который среднее значение этого переменная попадает с заданной вероятностью, например, 95% доверительный интервал.

ДТП – дорожно-транспортное происшествие

KDE – ядерная оценка плотности

Монте-Карло метод — широкий класс вычислительных алгоритмов, которые опираются на повторяющуюся случайную выборку для получения численных результатов. В данной работе для оценки статистической значимости величин генерируются наборы точек, равномерно распределённых по дорожной сети.

ОРДТП — относительный риск ДТП

Относительный риск ДТП — отношение числа ДТП вдоль измененного (предположительно более безопасного) маршрута к числу ДТП оригинального маршрута, имеющего те же начальную и конечную точки.

УПО – участок повышенной опасности. В данной работе — либо кластер ДТП, либо дорожный сегмент с повышенным числом ДТП

Литература

1. Ф. А. Новиков, дискретная математика для программистов // Питер, 2002.
2. E. W. Dijkstra, A note on two problems in connexion with graphs // Numerische Mathematik volume 1, pages 269–271 (1959), doi:10.1007/BF01386390
3. Rodriguez MZ, Comin CH, Casanova D, Bruno OM, Amancio DR, Costa LdF, et al. (2019) Clustering algorithms: A comparative approach. PLoS ONE 14(1): doi: 10.1371/journal.pone.0210236
4. Ester M., Kriegel H-P., J. S, Sander J., Xu X., A density-based algorithm for discovering clusters in large spatial databases with noise, AAAIPress, 1996, pp. 226–231.
5. Yiqun Xie, Shashi Shekhar, 2019. Significant DBSCAN towards Statistically Robust Clustering. In '19: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, August 2019, Pages 31–40. DOI: 10.1145/3340964.3340968
6. Boeing, G. 2017. "OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks." Computers, Environment and Urban Systems 65, 126-139. doi:10.1016/j.compenvurbsys.2017.05.004
7. Edward L. Platt, Network Science with Python and NetworkX Quick Start Guide: Explore and visualize network data effectively // Packt Publishing, 2019, 190pp.
8. R. Bellman: On a Routing Problem // Quarterly of Applied Mathematics. 1958. Vol 16, No. 1. C. 87-90, 1958.
9. L. R. Ford, Jr., D. R. Fulkerson. Flows in Networks, Princeton University Press, 1962.
10. Highway routing of hazardous materials, NHI Course 38064, National Highway Institute.
11. Gustavo Alfredo Bula. Vehicle Routing for Hazardous Material Transportation. Operations Research [cs.RO]. Université de Technologie de Troyes; Universidad nacional de Colombia, 2018. English. NNT : 2018TROY0014 . tel-03146101

12. Safe route-finding: A review of literature and future directions, Soheil Sohrabi, Yanmo Weng Subasish Das, Stephanie German Paal, Accident Analysis & Prevention, ISSN: 0001-4575, Vol: 177, Page: 106816
13. Urban navigation beyond shortest route: The case of safe paths, Esther Galbrun, Konstantinos Pelechrinis, Evimaria Terzia, Information Systems, Volume 57, April 2016, Pages 160-171
14. https://en.wikipedia.org/wiki/Kernel_density_estimation
15. Safe and secure vehicle routing: a survey on minimization of risk exposure, Georg E. A. Fröhlicha, Margaretha Gansterera, and Karl F. Doernera, Intl. Trans. in Op. Res. 0 (2022) 1–35, DOI: 10.1111/itor.13130
16. Driver Route Planning Method Based on Accident Risk Cost Prediction, Xiaoleng Liao, Tong Zhou, Xu Wang, Rongjian Dai, Xuehui Chen and Xiangmin Zhu, Journal of Advanced Transportation Volume 2022, Article ID 5023052, DOI:https://doi.org/10.1155/2022/5023052
17. A Route Navigation System for Reducing Risk of Traffic Accidents, Ryo Takeno, Yosuke Seki, Masahiko Sano, Kenji Matsuura, Kenji Ohira, Tetsushi Ueta, 2016 IEEE 5th Global Conference on Consumer Electronics.
18. Abdelhamid, S., Elsayed, S. A., AbuAli, N. & Hassanein, H. S. Driver-centric route guidance. 2016 2016. IEEE, 1-6.
19. Bao, S., Nitta, T., Yanagisawa, M., Togawa, N., 2017. A Safe and Comprehensive Route Finding Algorithm for Pedestrians Based on Lighting and Landmark Conditions. In: IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, pp. 2439–2450.
20. Chandra, S., 2014. Safety-based path finding in urban areas for older drivers and bicyclists. Transp. Res. Part C: Emerg. Technol. 48, 143–157.
21. Hart, P., Nilsson, N. and Raphael, B. (1968) A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions Systems Science and Cybernetics, 4, 100-107. http://dx.doi.org/10.1109/TSSC.1968.300136
22. Hayes, S., Wang, S. & Djahel, S. Personalized Road Networks Routing with Road Safety Consideration: A Case Study in Manchester. 2020 2020. IEEE, 1-6.

23. Hoseinzadeh, N., Arvin, R., Khattak, A.J., Han, L.D., 2020. Integrating safety and mobility for pathfinding using big data generated by connected vehicles. J. Intell. Transp. Syst. 24, 404–420.
24. Jiang, S., Jafari, M., Kharbeche, M., Jalayer, M., Al-Khalifa, K.N., 2020. Safe route mapping of roadways using multiple sourced data. IEEE Trans. Intell. Transp. Syst.
25. Jiang, S., Zhang, Y., Liu, R., Jafari, M., Kharbeche, M., 2022. Data-Driven Optimization for Dynamic Shortest Path Problem Considering Traffic Safety. IEEE Trans. Intell. Transp. Syst.
26. John Krumm, Eric Horvitz, Risk-aware planning: methods and case study on safe driving routes, Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence February 2017 Pages 4708–4714.
27. Liu, Q., Kumar, S. & Mago, V. SafeRNet: Safe transportation routing in the era of Internet of vehicles and mobile crowd sensing. 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), 8-11 Jan. 2017 2017. 299-304.
28. Mata, F., Torres-Ruiz, M., Guzm´an, G., Quintero, R., Zagal-Flores, R., Moreno-Ibarra, M.,Loza, E., 2016. A Mobile Information System Based on Crowd-Sensed and Official Crime Data for Finding Safe Routes: A Case Study of Mexico City. Mobile Inf. Syst. 2016, 1–11.
29. Ouyang, W., Yu, C.-W., Yu, K.-M., Lin, K.-J., Chang, H.-T., 2014. Safe path planning strategy for bike net. Wireless Pers. Commun. 78, 1995–2007
30. Scott Hartshorn, Tell Me The Odds: A 15 Page Introduction To Bayes Theorem, 2017 (Kindle Edition)
31. https://ru.wikipedia.org/wiki/Теорема_Байеса
32. https://ru.wikipedia.org/wiki/Динамическое_программирование
33. Jamal Jokar Arsanjani, Alexander Zipf, Peter Mooney, Marco Helbich (eds.),OpenStreetMap in GIScience: Experiences, Research, and Applications // Springer International Publishing, 2015, 324pp.
34. https://en.wikipedia.org/wiki/Binary_classification

35. Heckerman, D. Geiger, and D. M. Chickering, Learning bayesian networks: The combination of knowledge and statistical data, Machine learning, vol. 20, no. 3, pp. 197–243, 1995.

36. Chainey S, Tompson L, Uhlig S., 2008. "The utility of hotspot mapping for predicting spatial patterns of crime". Secur J 21(1):4–28, DOI: 10.1057/palgrave.sj.8350066

37. Chainey, S., Ratcliffe, J., 2005. GIS and Crime Mapping. John Wiley and Sons, UK. DOI: 10.1002/9781118685181

38. Gramacki Artur, Nonparametric Kernel Density Estimation and Its Computational Aspects, Springer International Publishing, 2018. DOI: 10.1007/978-3-319-71688-6

39. Moran, P. A. P., 1950. Notes on Continuous Stochastic Phenomena. Biometrika, Vol. 37, No. 1/2 (Jun., 1950), pp. 17-23. DOI: 10.2307/2332142.

40. Okabe, A., Sugihara, K., 2012. Spatial Analysis along Networks: Statistical and Computational Methods; John Wiley & Sons: Hoboken, NJ, USA, DOI:10.1002/9781119967101

41. Songchitruksa, P., Zeng X., 2010. Getis–Ord Spatial Statistics to Identify Hot Spots by Using Incident Management Data. Transportation Research Record: Journal of the Transportation Research Board. 2165. pp 42-51. DOI: 10.3141/2165-05.

42. Yingjie, L., Liwei,Zh., Junping,Y., Pengtao,W., Ningke,H., Wei,Ch., and Bojie,F.2017. Mapping the hotspots and coldspots of ecosystem services in conservation priority setting. Journal of Geographical Sciences,27(6):681-696. DOI: 10.1007/s11442-017-1400-x

43. Massgis data-massachusetts department transportation massdot roads // URL: docs.digital.mass.gov.

44. QGIS // www.qgis.org, URL:https://www.qgis.org/en/site/

45. Open jump // www.openjump.org. URL:http://www.openjump.org/

46. MassDOT Crash Open Data Portal // Mass.gov. URL:https://massdot-impact-crashes-vhb.opendata.arcgis.com/search

47.SANET //sanet.csis.u-tokyo.ac.jp. URL:http://sanet.csis.u-tokyo.ac.jp/

48.Герштейн А. М., Терехов А. Н. Выявление участков повышенной опасности на дорогах Массачусетса в 2013-2018 годах // Компьютерные инструменты в образовании. 2021. No 1. С. 27–40. DOI: http://dx.doi.org/10.32603/2071-2340-2021-1-46-58

49. Sahnoon, Iyad & Ahmed, Mohamed & Alghafli, Abdulla. (2018). Integrating Traffic Safety in Vehicle Routing Solution. 251-263. DOI:10.1007/978-3-319-60441-1_25.

50. Is OSM Good Enough for Vehicle Routing? A Study Comparing Street Networks in Vienna Anita Graser, Markus Straub and Melitta Dragaschnig, 11th International Symposium on Location-Based ServicesAt: Vienna, AustriaVolume: Progress in Location-Based Services 2014

51. Boeing, G. 2017. "OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks." Computers, Environment and Urban Systems 65, 126-139. doi:10.1016/j.compenvurbsys.2017.05.004

52. Peter J. Huber, Elvezio M. Ronchetti, Robust Statistics Concomitant scale estimates, pg 172

53. Art B. Owen (2006), A robust hybrid of lasso and ridge regression. https://statweb.stanford.edu/~owen/reports/hhu.pdf

54. Koenker, Roger and Kevin F. Hallock. “Quantile Regression”. Journal of Economic Perspectives, Volume 15, Number 4, Fall 2001, Pages 143–156

55. Alan Agresti, Christine Franklin, Bernhard Klingenberg
Statistics: The Art and Science of Learning from Data,
Fourth Edition, Pearson, 2018

56. D’Agostino, R. and Pearson, E. S. (1973), “Tests for departure from normality”, Biometrika, 60, 613-622

57. Герштейн А. М., Терехов А. Н. Маршрутизация транспорта при наличии опасных участков на дороге (на примере города Спрингфилд, Массачусетс) // Компьютерные инструменты в образовании. 2022. No 2. С. 5–18, URL: http://cte.eltech.ru/ojs/index.php/kio/article/view/1729

58. Герштейн А. М., Терехов А. Н. Простой способ повышения безопасности дорожного движения за счет обхода опасных участков маршрута // Программная инженерия, 2023, Том 14, №3. DOI: https://dx.doi.org/10.17587/prin.14.103-109 (in Russian).

59. https://www.qgis.org/ru/site/

60. http://stat.gibdd.ru/

61. Герштейн, А. М., Терехов, А. Н. Обход опасных участков маршрута как способ повышения безопасности движения (на примере Санкт-Петербурга)./ / Компьютерные инструменты в образовании. 2023. N 1. С. 30-39 URL: http://cte.eltech.ru/ojs/index.php/kio/article/view/1755

62. https://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system

Приложение 1. Загрузка дорожного графа в собственном формате OSMNX и в формате ShapeFile (для GIS-приложения)

```
# _map_retrieve.py
import osmnx as ox
import pickle

city = 'Petersburg'
loc = city + ',RU'
ox.config(use_cache=True, log_console=True, all_oneway=False)

G = ox.graph_from_place(loc, network_type='drive')
G = ox.add_edge_speeds(G)
G = ox.add_edge_travel_times(G)

ox.io.save_graph_shapefile(G, 'data/' + 'spb_one_way_false', encoding='utf-8')

with open(city + '.p', 'wb') as f:
    pickle.dump(G, f)
```

Приложение 2. Нахождение ближайших к ДТП ребер дорожного графа

```
#edges_for_all_tas_spb_projected_gibdd.py
import pandas as pd
import networkx as nx
import osmnx as ox
from shapely.geometry import LineString,Point
import geopandas as gpd
from ipyleaflet import *
import pickle
import matplotlib.pyplot as plt
import datetime


max_dist = 35
save_oneway = False

with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
    G = pickle.load(f)
Gp = ox.project_graph(G)


if save_oneway:
   f = open('zzz_tas_to_edges_2019_2021_oneway_proj_spb_gibdd.csv','w')
else:
   f = open('zzz_tas_to_edges_2019_2021_twoway_proj_spb_gibdd.csv','w')
f.write('YEAR,DATE,TIME,ID,X,Y,LAT,LON,NODE1,NODE2,KEY,DIST\n')

# input TAs
ta_data = 'data/tas_raw_spb.csv'
fta = open(ta_data,'r')
ta_header = 'YEAR,DATE,TIME,LAT,LON,X,Y'
ta_items = ta_header.split(',')
fta.readline() # Skip header

def prepareRow(year, date, time, id, x, y, lat, lon, node1, node2, key, d):
    row = str(year) + ', '
    row += date + ', '
    row += time + ', '
    row += str(id) + ', '
    row += str(x) + ', '
    row += str(y) + ', '
    row += str(lat) + ', '
    row += str(lon) + ', '
    row += str(node1) + ', '
    row += str(node2) + ', '
    row += str(key) + ', '
    #row += str(d) + '\n'
    row += "{0:.5f}".format(d) + '\n'
    return row
```

Продолжение Приложения 2

```
#def write_csv_row(year, edge, ta_id, x, y, lat, lon, oneway):
def write_csv_row(year, date, time, edge, ta_id, x, y, lat, lon, oneway):
    node1 = edge[0]
    node2 = edge[1]
    key = edge[2]
    d = edge[3]
    row = prepareRow(year, date, time, ta_id, x, y, lat, lon, node1, node2, key,
d)

    f.write(row)
    if not oneway:
       #if the edge is two-way, add adge with opposite direction
       row = prepareRow(year, date, time, ta_id, x, y, lat, lon, node2, node1,
key, d)
       print('two way:', node1,node2,key)
       f.write(row)

def extract_year(date):
    return date.split('-')[0]


count = 0
ta_id = 126438
print(datetime.datetime.now().time())
while True:
    ta = fta.readline().split(',')
    if len(ta) < 7: break
    year = ta[ta_items.index('YEAR')]
    date = ta[ta_items.index('DATE')]
    time = ta[ta_items.index('TIME')]
    lat = ta[ta_items.index('LAT')]
    lon = ta[ta_items.index('LON')]
    point = (float(lat), float(lon))
    point_geom_proj, crs = ox.projection.project_geometry(Point(reversed(point)),\
        to_crs=Gp.graph['crs'])
    x, y = point_geom_proj.x, point_geom_proj.y
    u, v, key, dist = ox.get_nearest_edge(Gp, (y, x), return_dist=True)
    edge = (u, v, key, dist)
    if dist < max_dist:
      if save_oneway:
         write_csv_row(year, date, time, edge, ta_id, x, y, lat, lon, True)
      else:
         oneway = G.edges[u,v,key]['oneway']
         write_csv_row(year, date, time, edge, ta_id, x, y, lat, lon, oneway)
      ta_id += 1
      count += 1
      if count % 100 == 0:
         print(count)
f.close()
```

Приложение 3. Оценка точности вычисления расстояний в UTM координатах с помощью формулы Евклида

#_test_utm_eucl.py

```
import osmnx as ox
import pickle
import math
from shapely.geometry import LineString,Point
from pyproj import Proj

coords_1 = (float(60.098310), float(30.305087))
coords_2 = (float(59.814286), float(30.374339))

with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
    G = pickle.load(f)
Gp = ox.project_graph(G)


dist = geopy.distance.geodesic(coords_1, coords_2).m

point1 = (coords_1[0], coords_1[1])
point2 = (coords_2[0], coords_2[1])

point_geom_proj, crs = ox.projection.project_geometry(Point(reversed(point1)),\
    to_crs=Gp.graph['crs'])
x1, y1 = point_geom_proj.x, point_geom_proj.y

point_geom_proj, crs = ox.projection.project_geometry(Point(reversed(point2)),\
    to_crs=Gp.graph['crs'])
x2, y2 = point_geom_proj.x, point_geom_proj.y

eucl_dist = math.sqrt((abs(x1-x2)**2) + (abs(y1-y2))**2)
print(eucl_dist - dist)
```

Приложение 4. Подсчет числа ДТП для каждого ребра дорожного графа

```
#edges_most_populated.py
import pandas as pd
import networkx as nx
import osmnx as ox
from shapely.geometry import LineString
import geopandas as gpd
from ipyleaflet import *
import pickle
import matplotlib.pyplot as plt
import numpy as np
import datetime
from operator import itemgetter


with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
    G = pickle.load(f)
ta_edges =
pd.read_csv('data/tas_to_edges_2019_2021_twoway_proj_spb_gibdd_flt_35.csv')
avoidf = open('edges_most_populated_spb_sorted_by_ta_flt_2_debug.csv', 'w')
avoidf.writelines('cn,node_start,node_end,key,lat0,lon0,lat1,lon1,tas,\
       ta_density\n')

elfn = "edge_list.csv"
elf = open(elfn, 'w')
elf.writelines('NODE1,NODE2,KEY\n')

def tas_along_edge(u, v,  key):
    edges_outg = ta_edges.loc[ta_edges['NODE1'] == u]
    edge_tas = edges_outg.loc[(edges_outg['NODE2'] == v) &\
                              (edges_outg['KEY'] == key)]

    tas = edge_tas.shape[0]
    oneway = G.edges[u, v, key]['oneway']
    if not oneway:
       tas = tas/2.0
    return tas

def equal_nodes(edge1, edge2):
    ret = False
    if (edge1[0] == edge2[0]) and (edge1[1] == edge2[1]) and\
       (edge1[2] == edge2[2]):
       ret = True
    return ret

edge_list = []
def collect_edges(G):
    edges = G.edges(keys = True, data = True)
    for edge in edges:
        current_edge = [edge[0],edge[1],edge[2]]
        llen = len(edge_list)
        if llen != 0:
           pe = edge_list[llen-1]
           prev_edge = [pe[0],pe[1],pe[2]]
        if llen == 0:
           edge_list.append(edge)
```

Продолжение Приложения 4

```
        elif not equal_nodes(prev_edge, current_edge):
             edge_list.append(edge)


def save_paths_to_avoid(edges,n):
    for i in range(n):
        density = f"{edges[i][8]:.3f}"
        line = str(i) + ',' + str(edges[i][0]) + ',' + str(edges[i][1]) +\
               ','+ str(edges[i][2]) + ','+ str(edges[i][3]) +\
               ','+ str(edges[i][4]) +',' + str(edges[i][5]) +\
               ',' + str(edges[i][6])+ ',' + str(edges[i][7])+\
               ',' + str(density) + '\n'
        avoidf.writelines(line)
    avoidf.close()

ds = []
ts_edges = []
def tas_to_edges(G):
    for edge in edge_list:
        node0 = edge[0]
        node1 = edge[1]
        n_edges =  G.number_of_edges(node0, node1)
        for key in range(n_edges):
            tas = tas_along_edge(node0, node1, key)
            l = G.edges[node0, node1, key]['length']
            d = tas/l
            lon0 = G.nodes[node0]['x']
            lat0 = G.nodes[node0]['y']
            lon1 = G.nodes[node1]['x']
            lat1 = G.nodes[node1]['y']
            ts_edge = [node0, node1, key, lat0, lon0, lat1, lon1, tas, d]
            ts_edges.append(ts_edge)
collect_edges(G)
print('Total edges:', len(edge_list) )
for edge in edge_list:
    row = str(edge[0]) + ',' + str(edge[1]) + ',' + str(edge[2]) + '\n'
    elf.writelines(row)
elf.close()
tas_to_edges(G)
ts_sorted = sorted(ts_edges, key=itemgetter(7), reverse=True)
save_paths_to_avoid(ts_sorted, len(ts_edges))
```

Приложение 5. Статистические испытания на дорожном графе с равномерно распределенными точками

```
"""
Script:
collect_trials.py - performs Monte-Carlo trials on most TA-populated edges
Input:
Road network (.p file for example "Peterburg.p")
.csv list of Sanet-generated points and respective edges points belong to.
.csv list of the TA-populated edges sorted by TA.
Output:
.csv - TA-populated edges along with results of Monte-Carlo trials.
"""

import pandas as pd
import networkx as nx
import osmnx as ox
from shapely.geometry import LineString
import geopandas as gpd
#import geopy.distance
#import json
from ipyleaflet import *
import pickle
import matplotlib.pyplot as plt
import numpy as np
#import sys
import datetime



tsize = 13503

tot_trials = 1000
tot_edges = 2000

with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
    G = pickle.load(f)

#points uniformly distributed over edges
points_sanet_edges = pd.read_csv('data/sanet_points_to_edges_flt_35_spb.csv')
print(points_sanet_edges)

"""
def currentTime():
    e = datetime.datetime.now()
    time = "%s:%s:%s" % (e.hour, e.minute, e.second)
    return time
"""

sanet_trials =[]
def prepare_sanet_points(trials, size):
    #df.iloc[2:5]
    for i in range(trials):
        init = i * size
        end = (i+1) * size - 1
        trial = points_sanet_edges.iloc[init:end]
        sanet_trials.append(trial)
```

Продолжение Приложения 5

```
prepare_sanet_points(tot_trials, tsize)
print(len(sanet_trials))

ta_edges = open('data/edges_most_populated_spb_sorted_by_ta_flt_35_debug.csv' ,
'r')
mhdr = ta_edges.readline()
hp = mhdr.split(',')
hp[-1] = hp[-1].strip()
print(len(hp))
sanet_trials_file = open('sanet_trials_flt_35_half_two_way.csv', 'w')
hdr = 'node_start,node_end,key,tas,trial_values\n'
sanet_trials_file.writelines(hdr)

edge_cnt = 0
while True:
      if edge_cnt > tot_edges: break;
      line = ta_edges.readline()
      vals = line.split(',')
      if len(vals) < 10: break
      vals[-1] = vals[-1].strip() #delete last symbol '\n'
      edge1 = int(vals[hp.index('node_start')])
      edge2 = int(vals[hp.index('node_end')])
      key   = int(vals[hp.index('key')])
      tas = vals[hp.index('tas')]
      #oneway = G.edges[edge1,edge2,key]['oneway']

      row = str(edge1)+','+str(edge2)+','+str(key)+','+tas+','
      for trial in sanet_trials:
          pi = trial.loc[trial['EDGEI'] == edge1]
          pe = pi.loc[pi['EDGEE'] == edge2]
          pk = pe.loc[pe['EDGEK'] == key]
          pnts = pk.shape[0]
          #if not oneway: pnts = pnts/2.0
          row = row + str(pnts) + '#'
      row  += '\n'
      sanet_trials_file.writelines(row)
      print('edge number:', edge_cnt)
      edge_cnt += 1
sanet_trials_file.close()
```

Приложение 6. Представление равномерно распределенных точек, полученных с помощью программы SANET, в виде .csv файла

```
from configparser import InterpolationError
import shapefile
import networkx as nx
import osmnx as ox
from datetime import datetime
import numpy as np
import pandas as pd
import pickle


with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
    G = pickle.load(f)

#G_proj = ox.projection.project_graph(G)

def currentTime():
    now = datetime.now()
    current_time = now.strftime("%H:%M:%S")
    return now

fname = 'data/sanet/SANETRandomPoint20m.shp'
#fname = '../prepare_data/data/moscow_random30m.shp'

out_name = 'sanet_points_to_edges_flt_35_spb.csv'
outf = open(out_name,'w')
outf.writelines('N,LAT,LON,EDGEI,EDGEE,EDGEK\n')



print(currentTime(), ' Start reading sanet points')
sf = shapefile.Reader( fname )
mkrecords = sf.records()
print(currentTime(), ' End reading sanet points')
```

Продолжение Приложения 6

```
#trials = 1000
trials = 1000
tsize = 13503
#tsize = 10
tr = 0
cnt = 0
X = []
Y = []
for record in mkrecords:
    lat = record[2]
    lon = record[1]
    X.append(record[1])
    Y.append(record[2])
    cnt += 1
    if cnt >= tsize:
       edges =  ox.distance.get_nearest_edges(G,X,Y,method='balltree')
       for i, edge in enumerate(edges):
           out_str = str(tr) + ',' + str(Y[i]) + ',' + str(X[i]) + ',' +
str(edge[0]) + ',' + str(edge[1]) + ',' + str(edge[2]) + '\n'
           outf.writelines(out_str)
       X = []
       Y = []
       print(currentTime(), ' Trial: ', tr)
       tr += 1
       cnt = 0
       if tr >= trials: break
outf.close()
print(currentTime(), 'End')
```

Приложение 7. Нахожденрие статистически достоверных значений ДТП

```
"""
Script:
select_significant_edges.py – select statistically significant edges
Input:
sanet_trials_flt_35_half_two_way.csv – sanet points with results of trials
Output:
sanet_trials_flt_35_signifcant_<p>.csv – statistically significant edges used as
input for routing script
routing_finite_penalty.py.

"""
import numpy as np


p = 95

def percentile(vals,p):
    vals[-1] = vals[-1].strip()
    tot_vals = vals[hdrps.index('trial_values')][:-1]
    str_vals = tot_vals.split('#')
    str_vals[-1] = str_vals[-1].strip()
    ivals = [int(str_val) for str_val in str_vals]
    #vls = np.array(ivals)
    percentile = np.percentile(ivals,p)
    return percentile


trialsf = open('data/sanet_trials_flt_35_half_two_way.csv','r')
outpf = open('sanet_trials_flt_35_signifcant_' + str(p)+ '.csv','w')
hdr = trialsf.readline()
outpf.writelines(hdr)
hdrps = hdr.split(',')
hdrps[-1] = hdrps[-1].strip()
while True:
    row = trialsf.readline()
    rowps = row.split(',')
    if len(rowps) < 5: break
    tas = rowps[hdrps.index('tas')]
    perc = percentile(rowps,p)
    if float(tas) > perc:
       outpf.writelines(row)
trialsf.close()
outpf.close()
```

Приложение 8. Создание квадратной решетки, наложенной на дорожный граф

```
import pickle
import osmnx as ox
import utm
import sys

with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
     G = pickle.load(f)


# upper left 55,915650900000003, 37,366904099999999
ilat = 60.05
ilon = 30.14
width = 50000
height = 50000
step = 3000

csv_header = 'num,X,Y,LAT,LON\n'
gname = 'spb_grid_' + str(step)+ '.csv'
gf = open(gname,'w')
gf.write(csv_header)

# X,Y
ini_coords = utm.from_latlon(ilat, ilon)
print (ini_coords)
x0 = ini_coords[0] - 3000
y0 = ini_coords[1] + 4500

irange = int(width/step)
jrange = int(height/step)

k = 0
for i in range(irange):
    x = x0 + i*step
    for j in range(jrange):
        y = y0 - step*j
        #latlon = utm.to_latlon(x,y, 37, 'U')
        latlon = utm.to_latlon(x,y, 36, 'U' )
        row = str(k) + ',' + str(x) + ',' + str(y) + ',' + str(latlon[0]) + ','\
              + str(latlon[1]) + '\n'
        gf.write(row)
        k += 1
gf.close()

#Filter grid poionts
fname = 'grid_filtered_' + str(step) + '.csv'
ff = open(fname,'w')
fg = open(gname,'r')
header = fg.readline()
ff.write(header)
max_dist  = step/2
```

Продолжение Приложения 8

```
while True:
    row = fg.readline()
    rvals = row.split(',')
    if len(rvals) < 5: break
    lat = float(rvals[3])
    lon = float(rvals[4])
    nodeDist = ox.get_nearest_node(G, (lat,lon), method='haversine',\
               return_dist=True)
    print(nodeDist)

    if nodeDist[1] < max_dist:
       ff.write(row)

ff.close()
```

Приложение 9. Программа для собирания статистики относительного риска

```
import numpy as np
import pandas as pd
import networkx as nx
import osmnx as ox
from shapely.geometry import LineString
import geopandas as gpd
import geopy.distance
from ipyleaflet import *
import pickle
import matplotlib.pyplot as plt
import datetime
from itertools import product
import sys


# To run with command line

if len(sys.argv) < 2:
   print ('Usage: routing_finite_penalty.py <p>, <max_path_to_avoid> <obstacles>
<penalty>')
#example: routing_finite_penalty.py 95 800
data/sanet_trials_flt_35_signifcant_95.csv 200
   sys.exit()

p = int(sys.argv[1])
max_paths_to_avoid = int(sys.argv[2])
fobst = sys.argv[3]
penalty = int(sys.argv[4])
print('penalty=', penalty)



"""
p = 95
max_paths_to_avoid = 800
fobst = 'data/sanet_trials_flt_35_signifcant_95.csv'
penalty = 1000
"""


debug = True
#fobst = 'data/paths_to_avoid_moscow_sorted_by_ta.csv'
#node_start,node_end,key,tas,trial_values
#fobst = 'data/paths_to_avoid_moscow_sorted_by_ta_flt_35_half_two_way.csv'
grid_step = 3000
#max_paths_to_avoid = 500
#penalty = 500
obst_edges = []



def read_obst_edges(fname, maxp):
    cnt = 0
    fl = open(fname, 'r')
    hdr = fl.readline().split(',')
    hdr[-1] = hdr[-1].strip()
```

Продолжение Приложения 9

```
    while(True):
         ln = fl.readline()
         if ln[0] == '#': continue
         print(ln)
         if(ln == ''): break
         lnp = ln.split(',')
         if len(lnp) < 5: break
         start_node = int(lnp[hdr.index('node_start')])
         end_node = int(lnp[hdr.index('node_end')])
         key = int(lnp[hdr.index('key')])
         edge = [start_node, end_node, key]
         obst_edges.append(edge)
         cnt += 1
         if cnt == maxp: break
    fl.close()

def split_row(header):
    hdrps = header.split(',')
    hdrps[-1] = hdrps[-1].strip()
    return hdrps,len(hdrps)


def set_obstacles(fname, penalty, maxp):
    global GO # with obstacles
    global G  # original graph
    fl = open(fname, 'r')
    hd = fl.readline()
    hdr,hl = split_row(hd)
    cnt = 0
    while(True):
         ln = fl.readline()
         if ln[0] == '#': continue
         #print(ln)
         lnp, lcs = split_row(ln)
         if len(lnp) < hl: break
         start_node = int(lnp[hdr.index('node_start')])
         end_node = int(lnp[hdr.index('node_end')])
         key = int(lnp[hdr.index('key')])
         origEdgeLength = G.edges[ start_node, end_node, key]['length']
         GO.edges[ start_node, end_node, key]['length'] = origEdgeLength + penalty
         cnt += 1
         if cnt == maxp: break
    fl.close()

with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
    G = pickle.load(f)

with open("data/Petersburg.p", 'rb') as f:  # notice the r instead of w
    GO = pickle.load(f)

# Read grid points
gridfn = 'data/grid_filtered_' + str(grid_step) + '.csv'
df = pd.read_csv(gridfn)
print(df.shape)

# Read TAs with the edges TAs belong to
ta_edges =
pd.read_csv('data/tas_to_edges_2019_2021_twoway_proj_spb_gibdd_flt_35.csv')
print(ta_edges)
```

Продолжение Приложения 9

```
def tas_along_edge(u, v,  key, oneway, year=0):
    if year == 0:
       edges_outg = ta_edges.loc[ta_edges['NODE1'] == u]
    else:
       edges_outg = ta_edges.loc[ta_edges['NODE1'] == u & ta_edges['YEAR'] ==\
                    year]
    edge_tas = edges_outg.loc[(edges_outg['NODE2'] == v) & (edges_outg['KEY'] ==\
               key)]
    tot_tas = edge_tas.shape[0]
    if not oneway:
       tot_tas = 0.5 * edge_tas.shape[0]
    return tot_tas


def tas_along_route(G, route, year=0):
    total_tas = 0
    rl = len(route)
    node0 = route[0]
    for i in range(1,rl):
        node1 = route[i]
        n_edges =  G.number_of_edges(node0, node1)
        if n_edges == 1:
           edge = 0
        else:
           #Find edge with min length in case there are several edges between two
           # nodes
           edge = 0
           l_dest = G.edges[node0, node1, 0]['length']
           for k in range(1, n_edges):
               l_attr = G.edges[node0, node1, k]['length']
               if l_attr  < l_dest:
                  l_dest = l_attr
                  edge = k
        oneway =  G.edges[node0, node1, edge]['oneway']
        if year == 0:
           total_tas += tas_along_edge(node0, node1, edge, oneway)
        else:
           total_tas += tas_along_edge(node0, node1, edge, oneway, year)
        node0 = node1    # next edge begins where previous one ends
    return total_tas

# Compare two routes
def routesAreEqual(route1,route2):
    result = True
    if len(route1) != len(route2): return False
    for i in range(len(route1)):
        if route1[i] != route2[i]:
           result = False
           break
    return result

# Check if an edge is in list of "prohibited" edges
def edgeWithPenalty(node1, node2, key):
    ne = obst_edges_n.shape[0]
    for i in range(ne):
        v = obst_edges_n[i]
        trio = np.array([node1,node2,key])
        if (v==trio).all():
           return True
    return False
```

Продолжение Приложения 9

```
def inspect_route(G,route):
   extra_length = 0
   penalties = 0
   for i in range(len(route)-1):
        u = int(route[i])
        v = int(route[i + 1])
        n_edges =  G.number_of_edges(u, v)
        ek = 0
        if n_edges > 1:
           # Find edge with min length
           minl =  G.edges[u,v,0]['length']
           for i in range(1, n_edges):
              li = G.edges[u,v,i]['length']
              if li < minl:
                 minl = li
                 ek = i
        if edgeWithPenalty(u,v,ek):
              extra_length += penalty
              penalties += 1
   return extra_length, penalties

read_obst_edges(fobst,max_paths_to_avoid)
obst_edges_n = np.array(obst_edges)

set_obstacles(fobst, penalty, max_paths_to_avoid)
grd_pnt = df.shape[0]

f_tar_name = 'routes_' + str(grid_step) + '_pen_' + str(penalty) +\
             '_edges_sorted_by_ta_sign' + str(p) + '_' + str(max_paths_to_avoid)+\

             '.csv'
f_tar = open(f_tar_name, 'w')
f_tar.write("BEGIN,END,ROUTE_LENGTH,ROUTE_O_LENGTH,ROUTE_NODES,ROUTE_O_NODES,\
             TA_DIRECT,TA_AVOID,AFFECTED\n")

print(datetime.datetime.now().time())

cin = 0
cend = 900000000
penalties_G = []
penalties_GO = []
penalties_G = []
for rin,rend in product(range(grd_pnt), range(grd_pnt)):
    ind = rin * grd_pnt + rend
    print(ind, ' of ', grd_pnt * grd_pnt )
    if ind < cin: continue
    if ind >= cend: break
    if rend == rin: continue
    row_from = df.loc[rin,:]
    row_to = df.loc[rend,:]
    start = (row_from[3],row_from[4])
    end = (row_to[3],row_to[4])
    dist = geopy.distance.distance(start, end).km
    start_node = ox.get_nearest_node(G, start)
    end_node = ox.get_nearest_node(G, end)
    try:
        route = nx.shortest_path(G, start_node, end_node, weight='length')
        penalties_G.append(inspect_route(G, route)[1])
```

Продолжение Приложения 9

```
        route_length = nx.shortest_path_length(G, start_node, end_node,
weight='length')
        route_nodes = len(route)
    except:
        continue
    try:
        route_o = nx.shortest_path(GO, start_node, end_node, weight='length')
        total_penalty = inspect_route(GO, route_o)[0]
        penalties_GO.append(inspect_route(GO, route_o)[1])
        if total_penalty != 0:
           print('penalty=', total_penalty)
        route_o_length = nx.shortest_path_length(GO, start_node, end_node,
weight='length') - total_penalty
        route_o_nodes = len(route_o)
    except:
        continue

    tas_direct = tas_along_route(G, route)
    equal_routes =  routesAreEqual(route, route_o)
    if not equal_routes:
        affected = ', yes'
        tas_avoid = tas_along_route(G, route_o)
    else:
        affected = ', no'
        tas_avoid = tas_direct

    row = str(start[0])+'#'+str(start[1]) + ', ' + str(end[0])+\
          '#'+str(end[1]) + ', ' + str(route_length) + ', ' +\
           str(route_o_length) +\
           ', '
    row += str(route_nodes) + ', ' + str(route_o_nodes) + ', ' +\
           str(tas_direct) + ', ' + str(tas_avoid) + affected + '\n'
    f_tar.write(row)
s = pd.Series(penalties_GO)
print('Modified route penalties')
print(s.describe())


s = pd.Series(penalties_G)
print('Original route penalties')
print(s.describe())

f_tar.close()
print(datetime.datetime.now().time())
```

Приложение 10. Визуализация данных

```
from os import stat_result
import geopy.distance
from math import sqrt
import numpy as np
import pandas as pd
from scipy import stats
import bootstrapped.bootstrap as bs
import bootstrapped.stats_functions as bs_stats
import matplotlib.pyplot as plt
import matplotlib.markers as markers



params = {'xtick.labelsize':'x-large',
          'ytick.labelsize':'x-large'}
plt.rcParams.update(params)

colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
'#e377c2', '#7f7f7f', '#bcbd22', '#17becf']


edges = ['200','500','1000','2000','5000']
fnames = ['routes_3000_pen_200_edges_sorted_by_ta_sign95_800.csv',
          'routes_3000_pen_500_edges_sorted_by_ta_sign95_800.csv',
          'routes_3000_pen_1000_edges_sorted_by_ta_sign95_800.csv',
          'routes_3000_pen_2000_edges_sorted_by_ta_sign95_800.csv',
          'routes_3000_pen_5000_edges_sorted_by_ta_sign95_800.csv',
          ]
pnlts = ['200','500','1000','2000','5000']
paths = open('routes_3000_pen_500_edges_sorted_by_ta_sign95_800.csv', 'r')

header = paths.readline()
hv = header.split(',')

markers = [".","o","v","^","+","h","s","*",]
```

Продолжение Приложения 10

```
def plot_dependency(ylabel, column,label,legend):

    rheader = 'eucl_dist','length_before','route_length_after/before',\
              'tas_after_before','nodes_after_before','penalty','affected']
    plots = []
    for i, pnlt in enumerate(pnlts):
        mean_ta_rt = []
        mean_ta_ci_l = []
        mean_ta_ci_u = []
        x = []
        xax = []
        start = 1
        total_routes = 0
        affected = 0
        print(pnlt)
        while start < 50:
            xax.append(str(start) + '-' + str(start + step))
            x.append(stat_result)

            ta_ratios = []
            for r in routes:

                val = r[1]/1000
                if r[5] == pnlt:
                    total_routes += 1
                    aff = r[rheader.index('affected')].strip()
                    if aff == 'yes':
                       affected += 1
                    if val > start and val < start + step:
                       ta_ratios.append(r[rheader.index(column)])
            stats = interval_stat(ta_ratios)
            mean = stats[1]
            mean_ta_rt.append(mean)
            mean_ta_ci_l.append(stats[2])
            mean_ta_ci_u.append(stats[3])
            print(start,start+step, ' ', len(ta_ratios), "{:.3f}".format(mean),\
                  "{:.3f}".format(stats[2]), '-', "{:.3f}".format(stats[3]))
            start = start + step
        print('Affected routes:', affected/total_routes)
        print('\n')

        if pnlt == '10000000000000': pnlt = '∞'
        y = np.array(mean_ta_rt)
        yl = np.array(mean_ta_ci_l)
        yu = np.array(mean_ta_ci_u)
        plot = [y,yl,yu]
        plots.append(plot)
    fig, ax = plt.subplots()
    ax.labelsize:'medium'  # fontsize of the x any y labels
    for j in range(len(pnlts)):
        ax.plot(xax,plots[j][0],label = label[j], color = colors[j],\
                marker=markers[j])
        ax.fill_between(xax, plots[j][1], plots[j][2], color=colors[j], alpha=.1)
    ht = np.ones(20)
    l = np.linspace(0,10,20)
    ax.plot(l,ht, color = 'lightgrey')
    lg = ax.legend(loc='upper right', fontsize=16)
    lg.set_title(legend,prop={'size':16})
    plt.xlabel('Длина первоначального (без препятствий) маршрута, км',fontsize=18)
    plt.ylabel(ylabel, fontsize=18)
```

Продолжение Приложения 10

```
def interval_stat(values):
    ntest = stats.kstest(values, 'norm')
    i_len = len(values)
    lr = np.array(values)
    samples = lr
    mean_ci = bs.bootstrap(samples, stat_func=bs_stats.mean, alpha=0.05)
    med = np.median(values)
    med_ci = bs.bootstrap(samples, stat_func=bs_stats.median, alpha=0.05)
    istat = [i_len, mean_ci.value , mean_ci.lower_bound, mean_ci.upper_bound,\
             med, med_ci.lower_bound, med_ci.upper_bound]
    return istat

def geoDist(col1,col2, prts, header):
    latlon1 = parts[hv.index(col1)].split('#')
    latlon2 = parts[hv.index(col2)].split('#')
    lat1 = float(latlon1[0])
    lon1 = float(latlon1[1])
    lat2 = float(latlon2[0])
    lon2 = float(latlon2[1])
    return geopy.distance.distance((lat1,lon1), (lat2,lon2)).km


routes = []
cnt = 0
for i in range(len(pnlts)):
    pnlt = pnlts[i]
    fname = fnames[i]
    paths = open(fname, 'r')
    header = paths.readline()
    hv = header.split(',')
    hv[-1] = hv[-1].strip()
    while(True):
        line = paths.readline()
        parts = line.split(',')
        if len(parts) < 9: break
        dist = geoDist('BEGIN', 'END', parts, hv)
        d = round(dist,3)
        tas_after = float(parts[hv.index('TA_AVOID')])
        tas_before = float(parts[hv.index('TA_DIRECT')])
        if tas_before != 0:
            tas_after_before = tas_after/tas_before
        else:
            continue
        length_after = float(parts[hv.index('ROUTE_O_LENGTH')])
        length_before = float(parts[hv.index('ROUTE_LENGTH')])
        length_after_before = length_after/length_before

        nodes_after = float(parts[hv.index('ROUTE_O_NODES')])
        nodes_before = float(parts[hv.index('ROUTE_NODES')])
        nodes_after_before = nodes_after/nodes_before
        affected = parts[hv.index('AFFECTED')].strip()
        r = [dist, length_before, length_after_before,\
             tas_after_before, nodes_after_before, pnlt, affected]
        routes.append(r)
    paths.close()

step = 5
```

Продолжение Приложения 10

```
plot_dependency('Относительный риск ДТП - ОРДТП','tas_after_before', edges, 'Штраф для грани-препятствия, м')
plt.show()

plot_dependency('Средняя длина пути после/до','route_length_after/before', edges, 'Штраф для грани-препятствия, м')
plt.show()

plot_dependency('Среднее число вершин после/до','nodes_after_before', edges, 'Штраф для грани-препятствия, м')
plt.show()
```