

федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский национальный
исследовательский университет информационных технологий, механики и
оптики»

Факультет информационных технологий и программирования
Направление (специальность) Прикладная математика и информатика
Квалификация (степень) Бакалавр прикладной математики и информатики
Кафедра компьютерных технологий Группа 4537

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной квалификационной работе

Построение семейств оптимальных маршрутов на морских картах

Автор квалификационной работы Громаковский И.Е. _____

Научный руководитель Ковалев А.С.

Консультанты:

- а) По экономике и организации производства _____
 - б) По безопасности жизнедеятельности и экологии _____
 - в)

К защите допустить

« » 2015 г.

Санкт-Петербург, 2015 г.

Квалификационная работа выполнена с оценкой _____

Дата защиты «_____» 2015 г.

Секретарь ГАК _____

Листов хранения _____

Чертежей хранения _____

федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет информационных
технологий, механики и оптики»

АННОТАЦИЯ ПО ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Студент _____ Громаковский И.Е.
Факультет _____ информационных технологий и программирования
Кафедра _____ компьютерных технологий Группа _____ 4537
Направление (специальность) _____ Прикладная математика и информатика
Квалификация (степень) _____ Бакалавр прикладной математики и информатики
Наименование темы Построение семейств оптимальных маршрутов на морских картах
Руководитель _____ Ковалев А.С., магистр прикладной математики и информатики
Консультант _____

КРАТКОЕ СОДЕРЖАНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ И ОСНОВНЫЕ ВЫВОДЫ

объем _____ 46 _____ стр., **графический материал** _____ – _____ стр., **библиография** _____ 16 _____ наим.

Направление и задача исследований

Целью настоящей работы является разработка эвристического алгоритма для решения в режиме реального времени задачи построения семейств оптимальных маршрутов кораблей на морских картах, а также реализация предложенного алгоритма в виде программного модуля. Под семейством оптимальных маршрутов понимается множество локально оптимальных путей между начальной и целевой точкой, отличающихся способом обхода существенных препятствий.

Проектная или исследовательская часть (с указанием основных методов исследований, расчетов и результатов)

Проводились исследования существующих алгоритмов для поиска семейств маршрутов. Для предобработки карт с целью получения графа были использованы методы вычислительной геометрии. Для поиска маршрутов были разработаны эвристики для обновления весов в графе и проверки критериев похожести маршрутов. С помощью различных оптимизаций удалось добиться работы в режиме реального времени. Программный модуль был реализован на языке C++ с использованием библиотек ЗАО «Кронштадт Технологии».

Экономическая часть (какие использованы методики, экономическая эффективность результатов)

С помощью разработанного программного модуля капитаны кораблей могут более эффективно прокладывать маршруты, что может быть экономически выгодно.

Характеристика вопросов экологии, техники безопасности и др.

Данная работа не оказывает влияния на вопросы экологии и не требует соблюдения никаких особых техник безопасности.

Является ли работа продолжением курсовых проектов (работ), есть ли публикации

Данная работа не является продолжением курсовых работ, публикаций на её основе нет.

Практическая ценность работы. Рекомендации по внедрению

Предложенный алгоритм был реализован и в настоящее время входит в состав программного обеспечения ЗАО «Кронштадт Технологии». Рекомендуется использование для решения задач поддержки принятия решения в навигационно-тактических системах морского назначения.

Выпускник _____

Научный руководитель _____

«____» _____ 2015 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
1. Обзор предметной области	7
1.1. Планирование маршрутов	7
1.2. Существующие алгоритмы поиска семейств маршрутов	10
1.2.1. Lim, Kim, 2005	10
1.2.2. Dijkstra-Hyperstar	14
1.3. Формальная постановка задачи	15
2. Теоретическое решение.....	20
2.1. Предобработка данных	20
2.2. Поиск одного маршрута	22
2.3. Поиск нескольких маршрутов	23
3. Практическая реализация и результаты.....	33
3.1. Предобработка.....	33
3.2. Вычисление метрик	34
3.3. Запретные зоны.....	36
3.4. Визуализация.....	37
3.5. Результаты.....	38
ЗАКЛЮЧЕНИЕ.....	44
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	45

ВВЕДЕНИЕ

Задача поиска кратчайшего пути при наличии полигональных препятствий хорошо известна и исследована. Классический способ решения этой задачи описан в [1]. Одним из её практических применений является поиск кратчайшего маршрута кораблей на морских картах, то есть кратчайшего пути между двумя точками, проходящего по воде. Однако при нахождении единственного кратчайшего маршрута возникают различные проблемы.

Во-первых, кратчайший путь не всегда является действительно оптимальным, с точки зрения пользователя. Например, на пути могут быть каналы, проплыть через которые возможно только за большую плату, что сделает такой маршрут менее привлекательным, чем какой-то другой, более длинный маршрут. В каких-то местах может быть нежелательно плавать по политическим соображениям, также у капитана корабля могут личные предпочтения. Формализовать всё множество критерииев, описывающих оптимальность маршрута, едва ли представляется возможным.

Во-вторых, иногда может возникнуть ситуация, при которой воспользоваться кратчайшим маршрутом не представляется возможным. Например, где-то могут проходить военные учения, из-за чего движение судов в таких местах будет запрещено. Информация, заложенная в карту, по которой строился маршрут, могла устареть, и какая-то река или канал могли полностью высохнуть. В таком случае пользователю хочется иметь альтернативный маршрут.

В-третьих, иногда нужно направить большое количество кораблей из одной точки в другую. Если сотни кораблей пойдут одним маршрутом, то они могут потратить очень много времени в очереди, чтобы проплыть по какому-нибудь каналу. Если же корабли пустить разными маршрутами, то суммарные временные затраты могут быть существенно уменьшены, несмотря на то что часть кораблей поплывёт не по кратчайшему пути.

Таким образом, возникает задача поиска семейств маршрутов между двумя точками. При этом маршруты должны быть в некотором смысле оптимальны. Например, логично требовать, чтобы маршруты были не сильно длиннее кратчайшего пути и попарно непохожи. Непохожие маршруты должны отличаться способом обхода существенных препятствий. Также хочется, чтобы маршруты были локально оптимальны по длине, то есть не могли быть сокращены в некоторой небольшой области. Сама по себе данная задача не при-

водит к конечному результату, а лишь осуществляет поддержку для принятия решения. Окончательное решение принимается пользователем в голове, поэтому основное требование к задачам поддержки принятия решения состоит в том, что они должны решаться практически моментально, в режиме реального времени, чтобы не сбивать пользователя с мыслей.

Для задачи поиска нескольких маршрутов (*multipath planning*) также известны некоторые решения. Например, существуют различные алгоритмы решения задачи *K-shortest paths*, состоящей в поиске первых K путей в графе по возрастанию длины [2, 3]. Однако нетрудно понять, что обычно такие пути будут иметь много общих рёбер и проходить через одни и те же водоёмы. Также известны и другие алгоритмы *multipath planning* [4–6]. Например, алгоритм [4] находит пути, которые имеют как можно меньше общих рёбер, однако при его применении к имеющейся задаче получаются похожие маршруты. Это связано с тем, что если есть, скажем, два маршрута, один из которых проходит на километр южнее другого, то они, как правило, хоть и почти не имеют общих рёбер, по сути являются очень похожими, поскольку обходят все препятствия с одной стороны (пусть и на разном расстоянии).

В первой главе приведён подробный обзор предметной области и существующих алгоритмов, формализована постановка задачи. Во второй главе описано теоретическое решение поставленной задачи, рассмотрены вопросы предобработки исходных данных и поиска семейств маршрутов. В третьей главе рассмотрены вопросы практической реализации предложенного решения и приведены основные результаты работы.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Планирование маршрутов

Задача планирования маршрута (англ. path planning) заключается в поиске пути из точки A в точку B , удовлетворяющего некоторым ограничениям. В данной работе слова «маршрут» и «путь» используются взаимозаменяемо. При планировании маршрута по воде ограничениями являются полигональные препятствия (материки, острова и так далее), которые запрещено пересекать. Стандартный способ решения этой задачи состоит в построении графа, вершины которого соответствуют некоторым точкам в мире, а ребро между двумя вершинами может присутствовать в графе только при условии, что оно не пересекает ни одно из препятствий, и поиске кратчайшего пути в этом графе. Известно [7], что любой кратчайший путь между двумя вершинами представляет из себя ломаную, вершины которой являются вершинами препятствий. Потому для поиска кратчайшего пути требуется построить *граф видимости*.

Граф видимости (англ. visibility graph) — граф, вершины которого являются вершинами полигональных препятствий, а ребро между вершинами u и v принадлежит графу тогда и только тогда, когда отрезок uv не пересекает ни одно препятствие.

Таким образом, любой кратчайший путь между двумя точками является кратчайшим путём в графе видимости и может быть найден с помощью любого алгоритма поиска кратчайшего пути в графе. Одними из наиболее популярных алгоритмов являются алгоритм Дейкстры [8] и A^* [9]. Стоит отметить, что довольно часто не требуется находить кратчайший маршрут, достаточно найти какой-нибудь корректный маршрут, желательно близкий к кратчайшему. В этом случае используется *навигационный граф*.

Навигационный граф — граф, построенный по полигональным препятствиям, вершины которого соответствуют некоторым точкам в пространстве, а ребро между вершинами u и v может принадлежать графу только тогда, когда отрезок uv не пересекает ни одно препятствие.

Как было отмечено во введении, задача планирования единственного кратчайшего пути имеет множество недостатков при поиске маршрутов по воде. Поэтому возникает задача поиска семейств маршрутов. Данная задача исследована не так хорошо, как планирование единственного маршрута, однако известны различные алгоритмы, обзор которых приведён в следующем

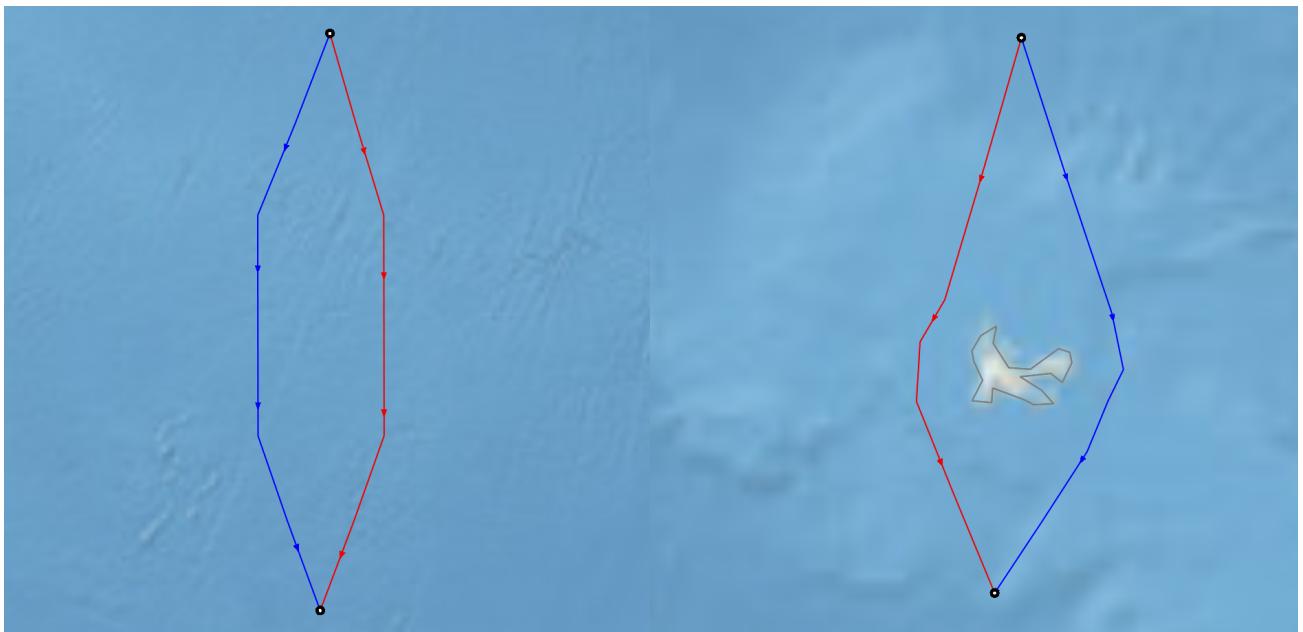


Рисунок 1 – Похожие маршруты

Рисунок 2 – Непохожие маршруты

разделе (1.2). При поиске семейств маршрутов по воде естественно требовать, чтобы все маршруты были, неформально говоря, попарно непохожи, отличаясь способом обхода существенных препятствий. Например, на рисунке 1 маршруты не имеют общих рёбер, но являются очень похожими. В то же время на рисунке 2 представлены непохожие маршруты, поскольку то, с какой стороны обходится остров, может быть достаточно существенно. Более наглядно неформальное описание похожести маршрутов проиллюстрировано на рисунках 3 и 4.

При решении такой задачи не следует использовать граф видимости по трём причинам:

1. Граф видимости позволяет найти кратчайший путь на плоскости, однако Земля не плоская, поэтому кратчайший путь в графе видимости не обязан являться кратчайшим маршрутом между двумя точками.
2. Данная задача выполняет поддержку принятия решения, заключающуюся в предложении альтернативных маршрутов, которые могут быть выбраны пользователем. При этом небязательно находить строго кратчайший путь, поскольку решение задачи в любом случае не является окончательным результатом.
3. В графе видимости, построенном по контурам, взятым с мелкомасштабной морской навигационной карты, будет слишком много рёбер, из-за чего поиск может выполняться неприемлимо долго.

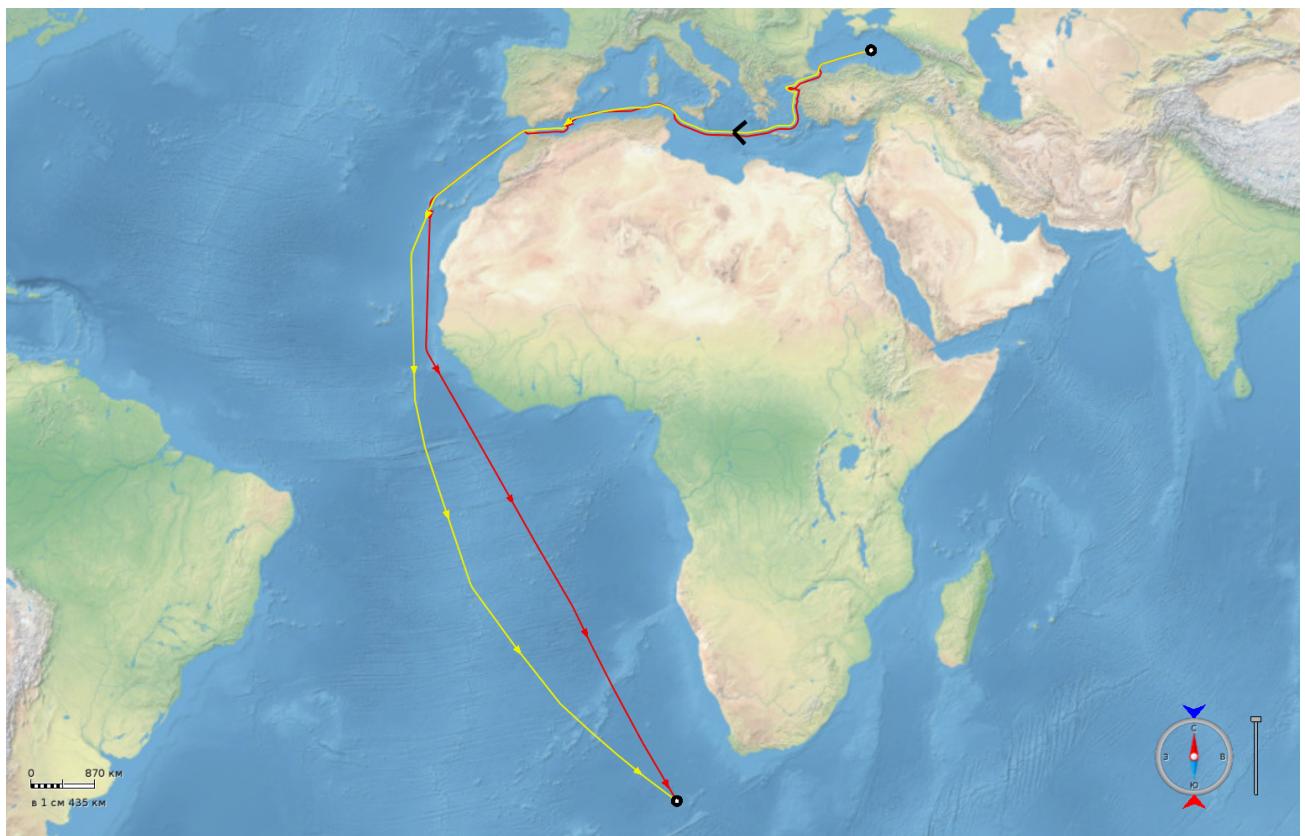


Рисунок 3 – Похожие маршруты

Поэтому для поиска используется навигационный граф, построение и структура которого описаны в главе 2.

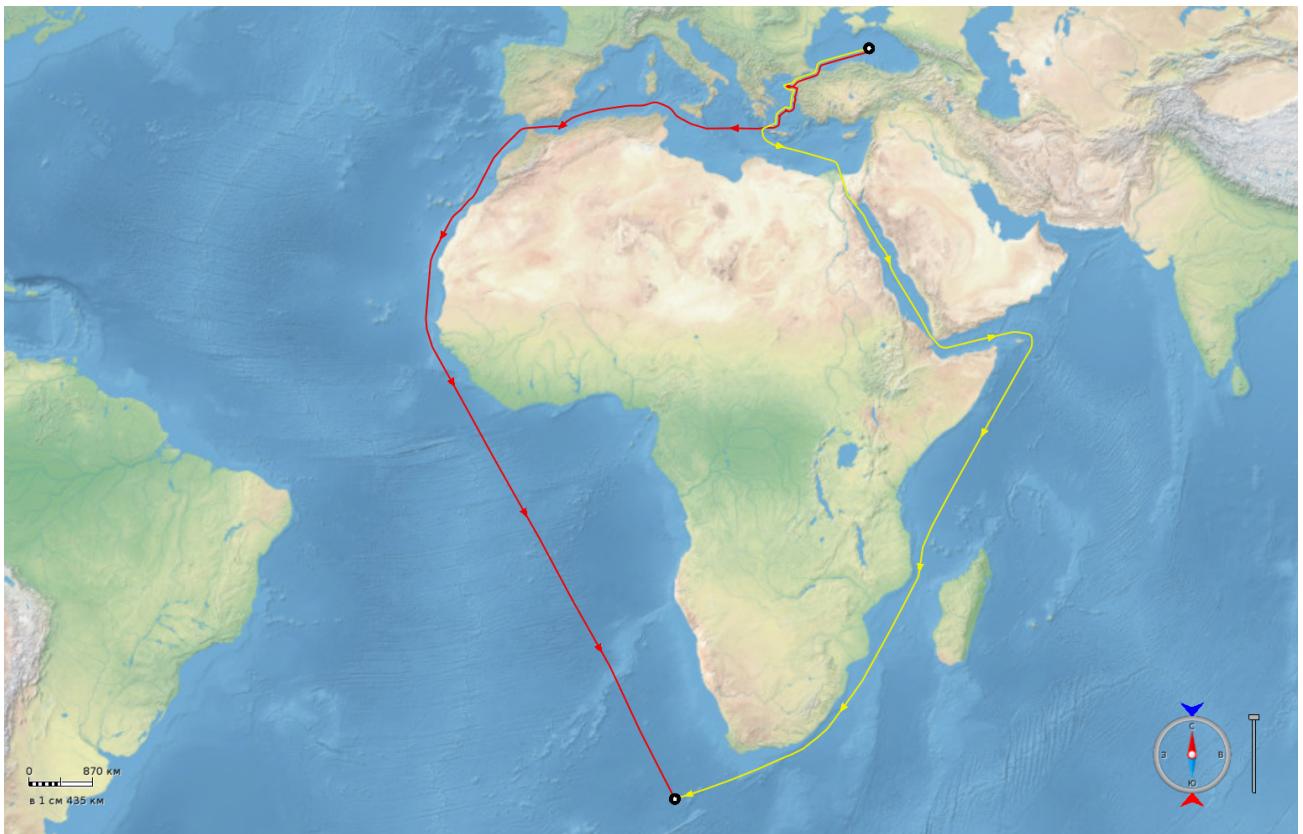


Рисунок 4 – Непохожие маршруты

1.2. Существующие алгоритмы поиска семейств маршрутов

Известно множество подходов к проблеме поиска семейств маршрутов [4–6]. Однако они обладают следующими недостатками:

- Все решения разрабатывались для других целей. Например, для поиска маршрутов по дорогам или в транспортных сетях. В этих случаях имеется граф дорог, в котором осуществляется поиск путей. Если два пути имеют мало общих рёбер, то они, как правило, имеют существенные различия. В то же время два маршрута по воде, не имеющие общих рёбер, могут проходить по одним и тем же морям и каналам.
- Предложенные алгоритмы работают с довольно абстрактными графиками и не учитывают привязанность вершин к реальным координатам в мире.
- Следствием первых двух пунктов является то, что при применении таких решений к задаче поиска семейств маршрутов кораблей находятся слишком похожие маршруты.

1.2.1. Lim, Kim, 2005

В [4] описывается алгоритм поиска непохожих маршрутов в сетях дорог. Непохожесть маршрутов является одной из главных целей алгоритма. В каче-

стве критерия, описзывающего похожесть маршрутов, используется отношение длины перекрывающейся части путей к длине одного из путей (степень перекрытия). Сам алгоритм устроен следующим образом: сначала происходит поиск кратчайшего пути в графе, затем веса всех рёбер на этом пути увеличиваются, считается степень перекрытия, и если она больше порогового значения, то процесс заканчивается, иначе найденный путь добавляется в результирующее множество, и процесс продолжается.

В статье приведён пример, демонстрирующий, что для графа дорог данное решение действительно находит непохожие пути. Однако при поиске семейств маршрутов в навигационном графе, построенном по полигональным препятствиям, недостаточно в качестве критерия, описывающего похожесть маршрутов, рассматривать лишь степень перекрытия. Рисунок 5 демонстрирует это. На рисунке представлено семь маршрутов, и нетрудно видеть, что любые два из них практически не имеют общих рёбер, то есть степень перекрытия действительно очень низкая. Однако все эти маршруты идут по одним и тем же морям и океанам, поэтому такие маршруты нельзя назвать непохожими. Второй недостаток данного подхода в том, что он находит слишком мало маршрутов. Например, на рисунке 6 показана ситуация, в которой данный алгоритм нашёл лишь маршрут через Панамский канал, тогда как хотелось бы видеть альтернативный маршрут через южную часть Южной Америки (рисунок 7). Это связано с тем, что после увеличения весов рёбер только на найденном маршруте следующий путь зачастую будет проходить очень близко к предыдущему, из-за чего алгоритм может сразу же прекратить работу.

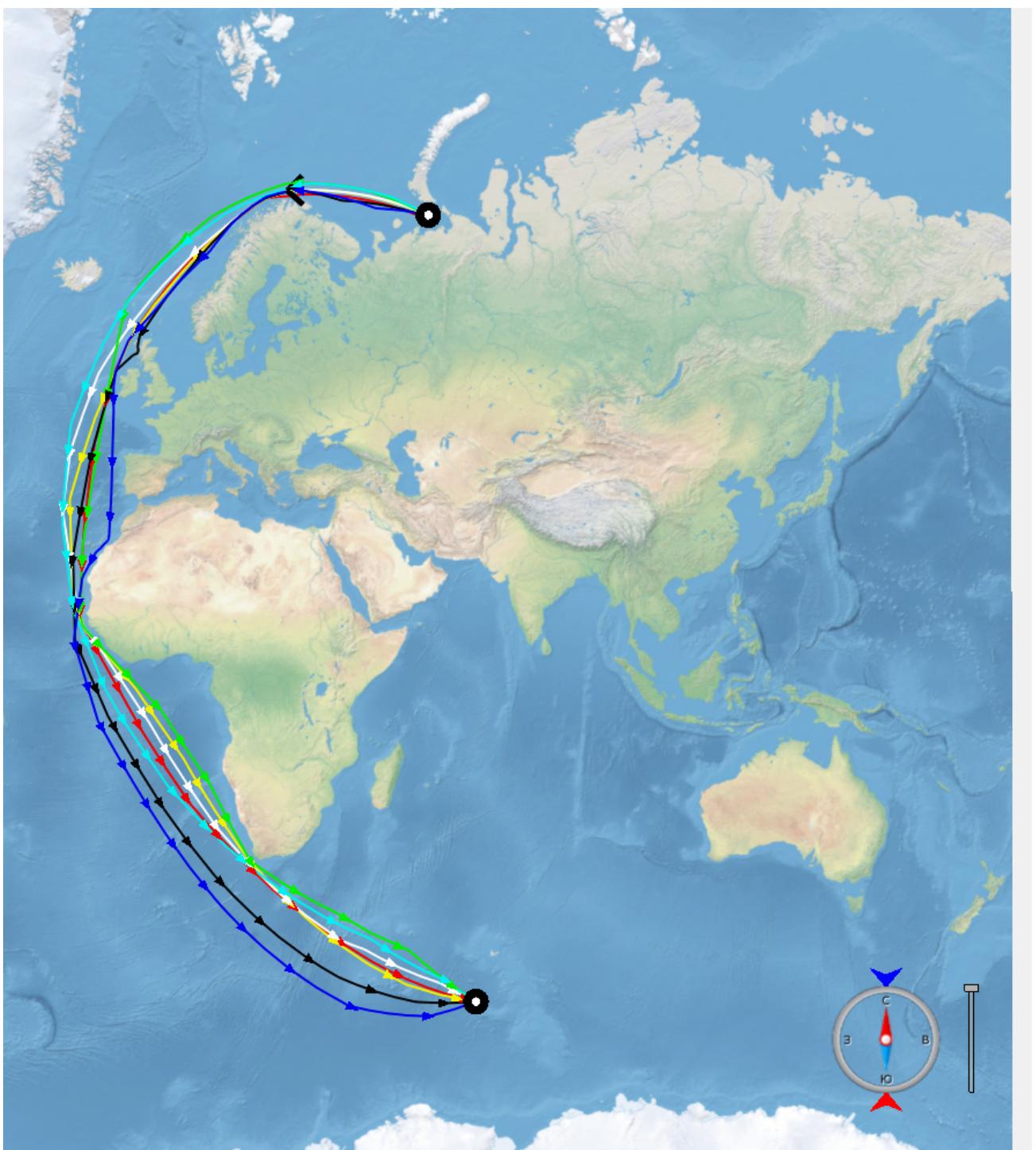


Рисунок 5 – Маршруты, найденные алгоритмом [4]

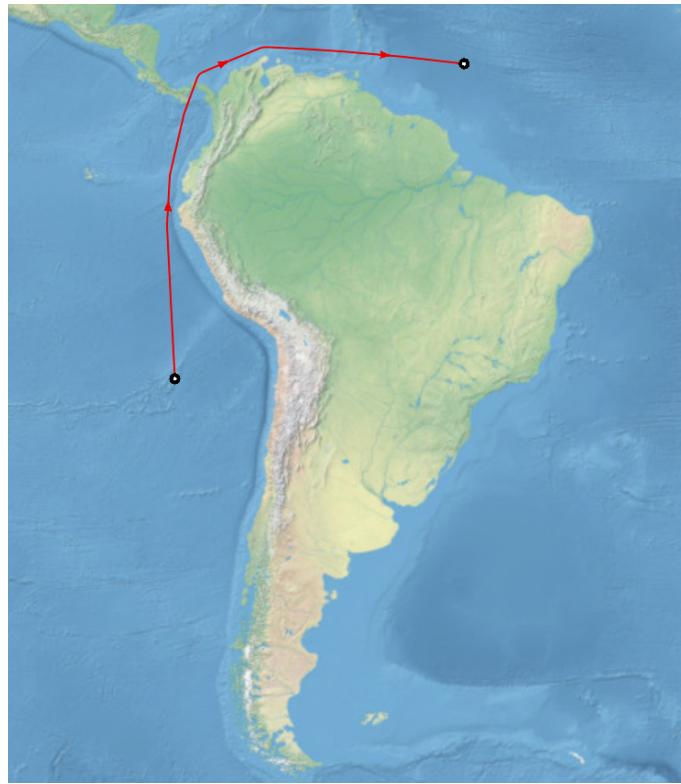


Рисунок 6 – Алгоритмом [4] найден лишь один маршрут

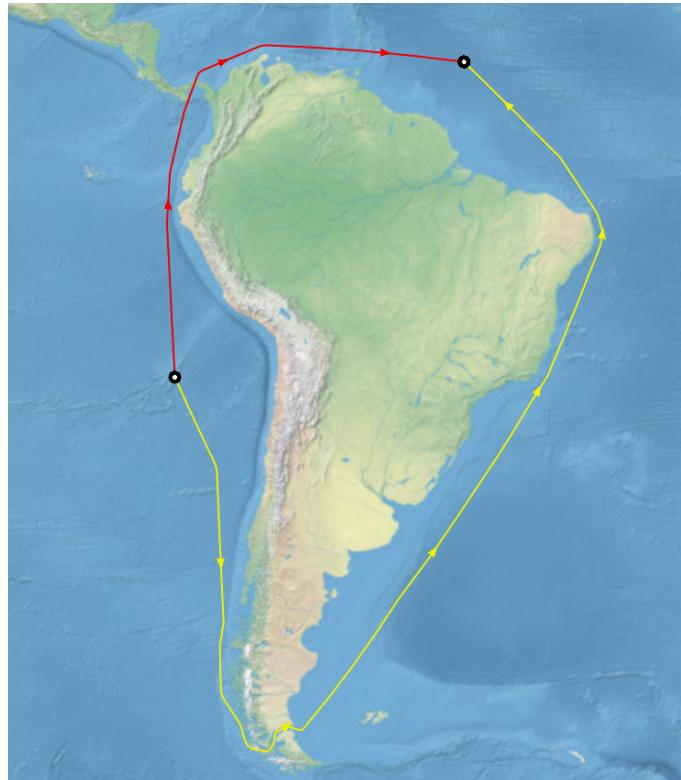


Рисунок 7 – Желаемые маршруты

1.2.2. Dijkstra-Hyperstar

В статье [6] описываются три алгоритма поиска нескольких маршрутов в графе, ориентированные на поиск путей в транспортных сетях. Данные алгоритмы приводят к похожим результатам и различаются в основном производительностью. Для анализа был выбран ASF (Adapted Spiess and Florian) алгоритм. В данном алгоритме каждое ребро графа помимо веса имеет некоторую частоту, которая соответствует тому, как часто ходит транспорт по этому ребру, поэтому напрямую он не может быть применён к задаче поиска семейств маршрутов по воде. Обратная величина частоты соответствует максимальному времени ожидания транспорта. Для исследования данного алгоритма частоты присваивались случайнным образом, чтобы максимальное время ожидания было того же порядка, что и веса рёбер. В результате данный алгоритм присваивает каждому ребру вероятность быть использованным в итоговом множестве маршрутов.

После присвоения вероятностей рёбрам необходимо построить сами маршруты. Для этого запускается случайный процесс, который на каждом шаге по текущей вершине выбирает следующую в зависимости от вероятностей, присвоенных исходящим рёбрам. Процесс начинается в стартовой вершине и заканчивается при достижении целевой точки. Если все исходящие рёбра имеют нулевую вероятность, то путь не может быть найден и алгоритм заканчивает свою работу, иначе процесс повторяется, пока не будет найдено максимальное число маршрутов или два одинаковых маршрута.

Главным минусом данного алгоритма является необходимость присвоения частот рёбрам. Если все частоты равны, то алгоритм присваивает нулевую вероятность всем рёбрам, не принадлежащим кратчайшему пути. Именно от частот зависит то, насколько различны будут итоговые маршруты. Также сам по себе алгоритм лишь находит вероятности рёбер, а значит, построению маршрутов с его использованием в любом случае будет свойственна недетерминированность. В большинстве же случаев маршруты получаются похожими (например, рис. 8).

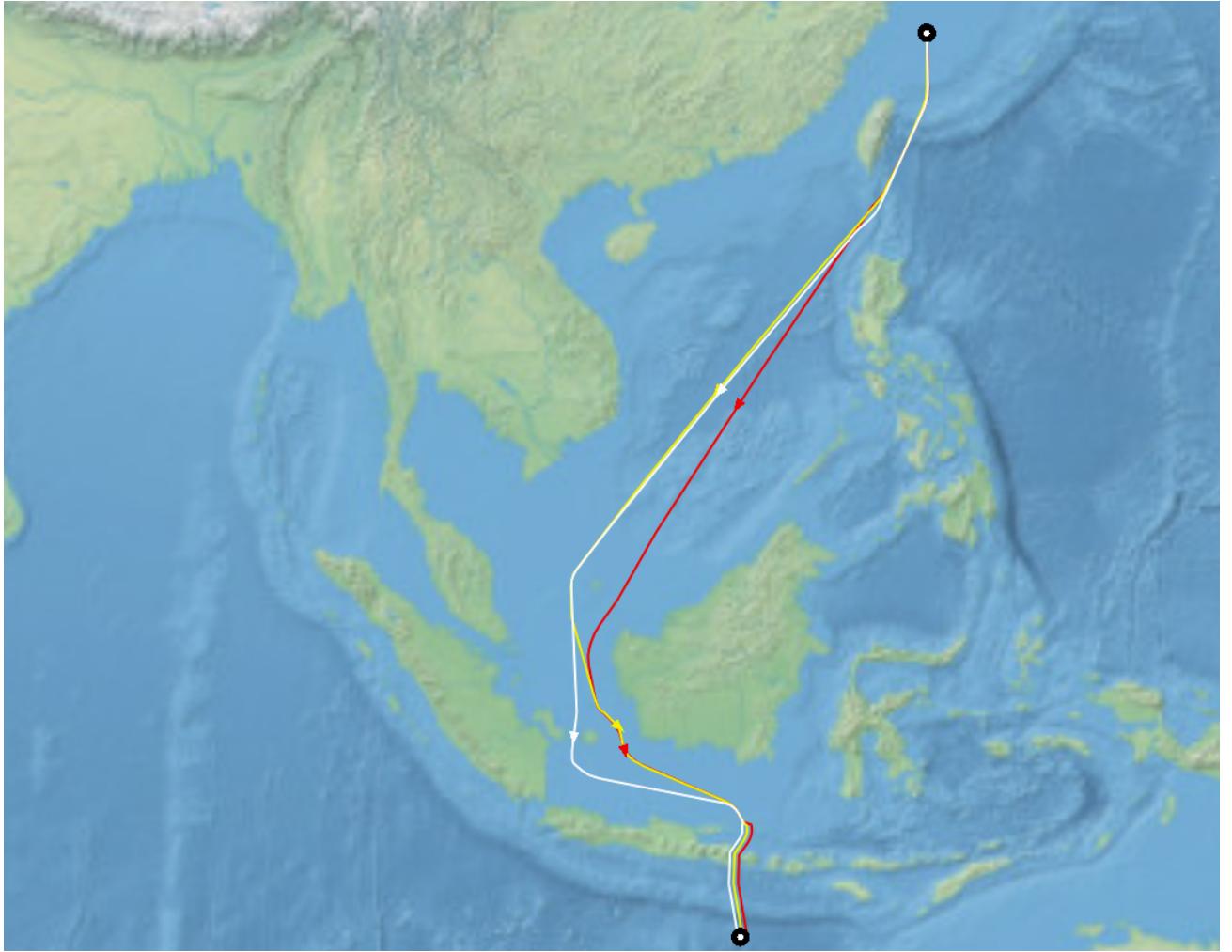


Рисунок 8 – Пример маршрутов, найденных ASF алгоритмом

1.3. Формальная постановка задачи

На основе проведённого обзора и неформальных требований можно формализовать постановку задачи. Дано множество полигональных препятствий C , начальная точка S и конечная точка D . По полигональным препятствиям строится навигационный граф. Требуется найти множество путей P из точки S в точку D , удовлетворяющих следующим свойствам:

- Каждый путь $p \in P$ представлен ломаной, любой отрезок которой не пересекает ни один контур из C .
- Пусть q — кратчайший путь. Обозначим как $\text{len}(p)$ длину пути p . Тогда $\forall p \in P : \text{len}(p) < C_0 \cdot \text{len}(q)$, где $C_0 > 1$ является параметром задачи.
- Любые два пути $p, q \in P$ не являются похожими в смысле критерия, сформулированного ниже.

Также неформальным требованием будет то, что маршруты должны быть локально оптимальны по длине. В разделе 2.2 описан подход, за счёт которого достигается такая оптимальность.

Для определения похожести маршрутов используются две метрики, заданные на множестве маршрутов:

$$\rho_1(P, Q) = \max\left(\max_{u \in P} \min_{v \in Q} \rho_g(u, v), \max_{v \in Q} \min_{u \in P} \rho_g(u, v)\right)$$

$\rho_g(u, v)$ означает длину кратчайшего пути в навигационном графе между вершинами u и v . В данной метрике для каждой вершины маршрута рассматривается длина кратчайшего пути до второго маршрута. Значением метрики является максимум всех таких длин. Большое значение этой метрики означает, что для какой-то вершины кратчайший путь в навигационном графе до второго маршрута имеет большую длину, что является хорошим показателем того, что маршруты непохожи.

$$\rho_2(P, Q) = \max\left(\max_{u \in P} \frac{\min_{v \in Q_u} \rho_g(u, v)}{\min_{v \in Q_u} \rho_r(u, v)}, \max_{v \in Q} \frac{\min_{u \in P_v} \rho_g(u, v)}{\min_{u \in P_v} \rho_r(u, v)}\right)$$

$$P_v = \{u \in P : \rho_r(u, v) > \varepsilon \cdot \text{len}(P)\}$$

$$Q_u = \{v \in Q : \rho_r(u, v) > \varepsilon \cdot \text{len}(Q)\}$$

$\rho_r(u, v)$ означает расстояние между u и v в мире. Данная метрика аналогична первой, однако максимум берётся не по длинам кратчайших расстояний в графе, а по отношению длины кратчайшего расстояния в графе к расстоянию до ближайшей вершины в мире. Большое значение этой метрики означает, что для какой-то вершины длина кратчайшего пути в навигационном графе до второго маршрута существенно больше, чем кратчайшее расстояние в мире. Это является показателем того, что между маршрутами имеется препятствие. При этом не рассматриваются пары слишком близких вершин, поскольку между ними может быть препятствие очень небольших размеров, которое не делает маршруты непохожими, однако делает значение метрики большим. Число ε также является параметром алгоритма.

На основании этих двух метрик формируется следующий критерий похожести маршрутов P и Q .

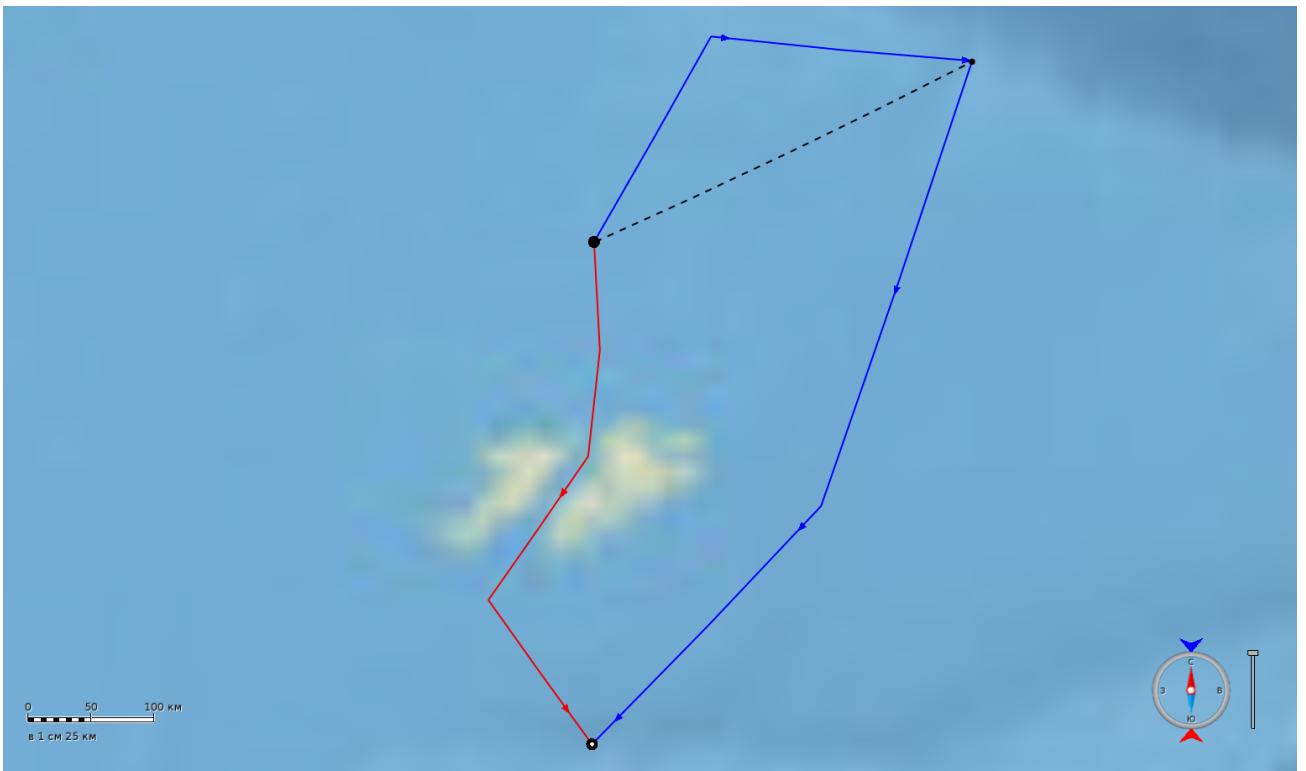


Рисунок 9 – Маршруты непохожи по первой метрике

- Если $\rho_1(P, Q) > C_{1max} \cdot \min(\text{len}(P), \text{len}(Q))$, то маршруты являются непохожими.
- Если $\rho_1(P, Q) < C_{1min} \cdot \min(\text{len}(P), \text{len}(Q))$, то маршруты являются похожими.
- Если ни один из первых двух пунктов не выполнен и $\rho_2(P, Q) > C_2 \cdot \min(\text{len}(P), \text{len}(Q))$, то маршруты являются непохожими.
- В противном случае маршруты являются похожими.

Константы C_{1max} , C_{1min} , C_2 также являются параметрами. В работе использовались следующие значения параметров, подобранные в результате экспериментов:

$$C_0 = 2, 5; C_{1max} = 0, 5; C_{1min} = 0, 2; C_2 = 1, 15; \varepsilon = 0, 1.$$

На рисунке 9 изображены два маршрута (синий и красный). Чёрная пунктирная линия соединяет две вершины, на которых достигается значение первой метрики (339 км). При этом длина красного маршрута составляет 435 км. Поскольку $\frac{339}{435} = 0, 78 > 0, 5$, то маршруты считаются непохожими.

Для маршрутов, изображённых на рисунке 10, значение первой метрики равно 502 км, а длина кратчайшего из двух маршрутов равна 1704 км. $0, 2 < \frac{502}{1704} = 0, 29 < 0, 5$, поэтому используется значение второй метрики. Поскольку между каждой парой вершин на маршрутах не имеется препятствий,

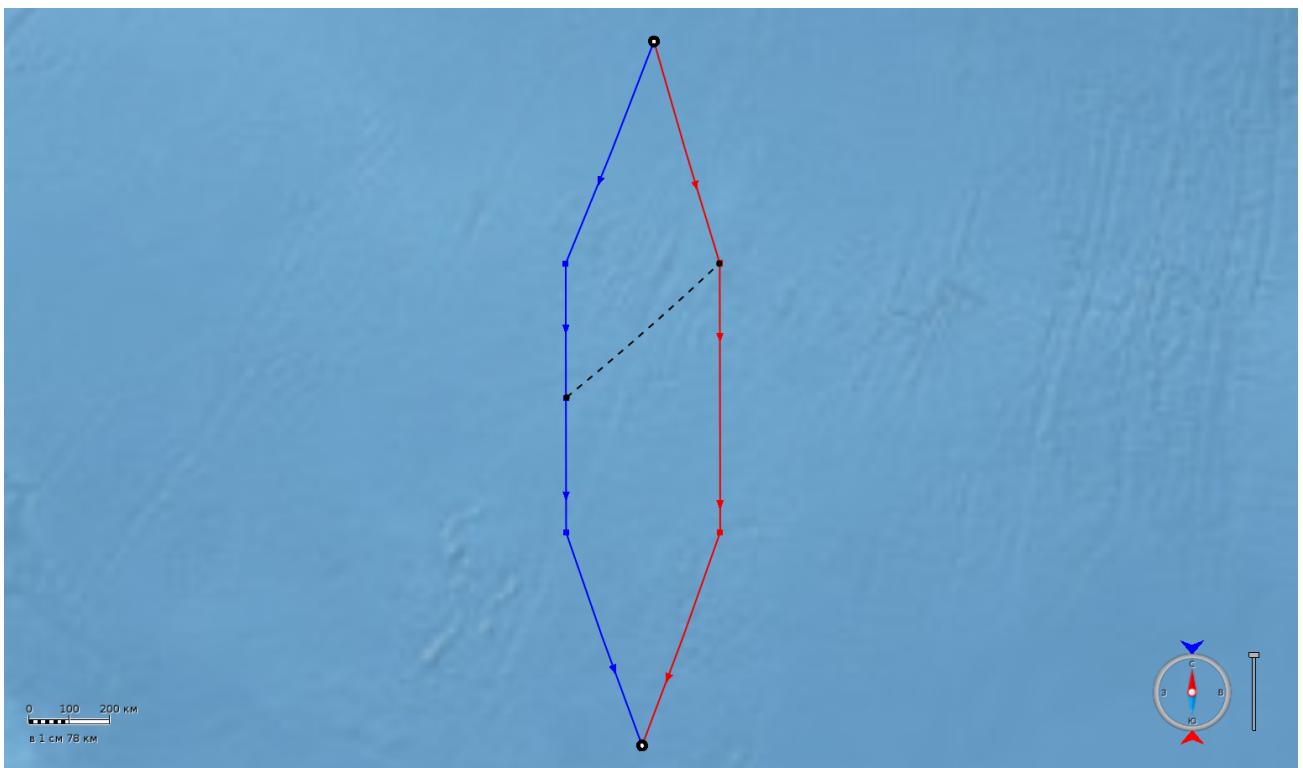


Рисунок 10 – Маршруты похожи по второй метрике

то значение этой метрики равно 1. Таким образом, эти два маршрута считаются похожими.

Для маршрутов, изображённых на рисунках 11 и 12, значение первой метрики равно 307 км (на первом рисунке пунктиром показано, для каких двух точек оно достигается). Длина кратчайшего из двух маршрутов равна 814 км. $0,2 < \frac{307}{814} = 0,38 < 0,5$, поэтому в данном случае тоже используется вторая метрика. На втором рисунке пунктиром показана ближайшая вершина второго пути для той же вершины первого. Расстояние до неё составляет 251 км. Значение второй метрики достигается между теми же двумя вершинами и равно $\frac{307}{251} = 1,22 > 1,15$. Поэтому данные два маршрута считаются непохожими, что полностью согласуется с неформальными требованиями, описанными ранее.

Наконец, на рисунке 13 представлены маршруты, для которых значение первой метрики составляет 641 км, а длина кратчайшего из них – 4059 км. Таким образом, поскольку $\frac{641}{4059} = 0,16 < 0,2$, то эти маршруты считаются похожими по первой метрике. Хоть между ними и имеется препятствие, его размеры пренебрежимо малы, поэтому логично считать такие маршруты похожими.



Рисунок 11 – Маршруты «возможно, похожи» по первой метрике...

Рисунок 12 – ... и точно непохожи по второй из-за острова между ними.

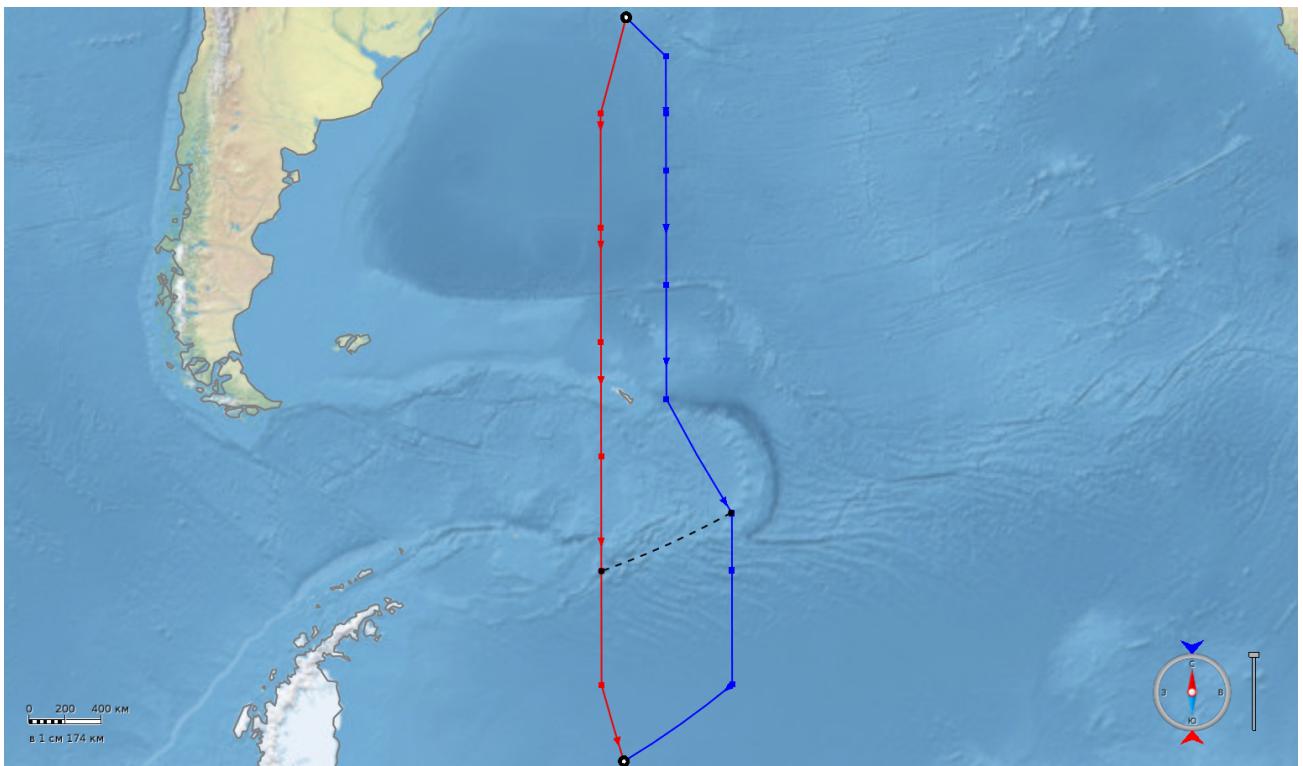


Рисунок 13 – Маршруты похожи по первой метрике

ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ РЕШЕНИЕ

2.1. Предобработка данных

Для решения поставленной задачи первым делом необходимо построить навигационный граф. Подразумевается, что карта содержит множество контуров, которые можно классифицировать на ограничивающие воду и ограничивающие сушу. Контуры заданы как последовательности точек в плоской проекции Меркатора [10]. Для построения графа выполняется определённый набор действий.

Первым делом происходит упрощение водных контуров, полученных из карты, и объединение их в один полигон. Упрощение выполняется по двум причинам:

- Данные с мелкомасштабной морской карты содержат слишком много точек, из-за чего в навигационном графе будет крайне много вершин и рёбер, если не упрощать контура. Это приведёт к тому, что граф будет занимать слишком много памяти, а поиск путей в нём будет занимать слишком много времени.
- Хоть такая модификация и может незначительно повлиять на корректность маршрута, действительно сильно некорректных маршрутов от неё появиться не может. Неформальная постановка задачи допускает такие модификации. Кроме того, планируется искать маршруты по всему земному шару, поэтому мелкомасштабные детали не несут никакой информации, а скорее наоборот, делают маршруты более сложными и менее естественными.

Для упрощения используется алгоритм Дугласа-Пекера [11].

После построения единого полигона, описывающего контуры воды, для него строится straight skeleton [12], с помощью которого выполняется смещение полигона внутрь. Это делается по следующим причинам:

- Корабли, как правило, не плавают слишком близко к суше (в двенадцатимильной зоне) [13]. При этом, даже если кораблю разрешено плавать где угодно, стоит отметить, что в данной задаче не требуется искать кратчайший маршрут, а требуется получать естественные маршруты, которые потенциально могут использоваться для реальных кораблей.
- Маршруты, проходящие по границе суши, хуже визуализируются, поскольку перемешиваются с сушей. В то же время при визуализации

маршрутов, проходящих на небольшом расстоянии от суши, таких проблем не возникает.

- Такая модификация нейтрализует возможные нарушения корректности маршрута, возникающие из-за упрощения контуров. Если при упрощении любой точка смещается на расстояние не больше l , а смещение контура происходит на расстояние $d > l$, то про любую точку, принадлежащую итоговому полигону, можно сказать, что в исходной карте она точно принадлежит контуру с водой.

По смещённому полигону строится навигационный граф. Для этого на плоскости, на которую спроектирована карта, строится сетка с некоторым постоянным шагом. Для каждого ребра сетки проверяется принадлежность полигону, ограничивающему воду. Если ребро полностью принадлежит полигону, то оно добавляется в граф (вместе с инцидентными вершинами). Помимо этого в граф добавляются все вершины полигона и рёбра до видимых вершин (то есть рёбра, полностью принадлежащие полигону) в некотором радиусе. При этом важно ограничить максимальную длину ребра. Это связано с тем, что график строится по полигону в плоской проекции, и проверка принадлежности отрезка полигону выполняется на плоскости. В то же время кратчайшее расстояние между двумя точками на Земле достигается не для отрезка, а для дуги. Поэтому такая проверка, вообще говоря, некорректна. Однако, если ограничить длину ребра, то отклонение дуги от отрезка будет невелико. Учитывая то, что полигон смещён внутрь, можно утверждать, что любое ребро действительно будет проходить по воде, не пересекая препятствия.

Также в полученный график добавляются дополнительные рёбра. Во-первых, поскольку график строится по контурам в плоской проекции Меркатора, то в нём отсутствуют рёбра через 180-ый меридиан. Такие рёбра добавляются в график. Во-вторых, добавляются рёбра, получившиеся в результате «схлопывания» контуров полигона при построении straight skeleton'a. Это необходимо сделать, потому что при смещении полигона некоторые контуры «схлопываются», объединяясь в один. Если между ними был, скажем, какой-нибудь канал, то по нему нельзя будет пройти в итоговом графике. Именно поэтому такие рёбра из straight skeleton'a добавляются в график. При этом они объединяются в цепочки рёбер, и к каждой цепочке применяется упрощение с помощью модифицированного алгоритма Дугласа-Пекера. Модификация заключается в

изменении предиката, проверяющего возможность замены ломаной на отрезок и выбирающего вершину для разбиения, если замена невозможна. Для проверки рассматривается не только максимальное расстояние от вершины ломаной до отрезка, но и корректность отрезка по исходному полигону. Если замена невозможна, то в качестве следующей вершины для разбиения берётся не самая далёкая, а такая, что получающиеся два отрезка будут корректны. Если такой вершины нет, берётся самая дальняя, как в оригинальном алгоритме. Помимо этого предоставляется возможность вручную добавлять рёбра в граф, поскольку исходные данные не всегда полностью корректны и актуальны.

2.2. Поиск одного маршрута

Прежде чем перейти к описанию алгоритма поиска семейств маршрутов, необходимо детально описать процесс поиска одного маршрута в построенном навигационном графе. Первым делом для двух заданных точек проверяется принадлежность полигону, ограничивающему воду. Если точки действительно находятся на воде, то по имеющейся сетке для них находятся ближайшие вершины графа, и все корректные рёбра до ближайших вершин добавляются в граф. После этого используется алгоритм Дейкстры [8] для поиска кратчайшего пути в графе между этими двумя вершинами. Поиск останавливается, когда кратчайший путь до нужной вершины найден.

Поскольку найденный путь не является действительно кратчайшим маршрутом между двумя точками, производится попытка его сократить. Для каждой вершины пути проверяется, можно ли её убрать из маршрута, не нарушив корректность рёбер маршрута и не увеличив длину пути. Корректность ребра определяется не только тем, пересекает ли оно какое-либо препятствие, но и, как отмечалось ранее, его длиной, которую важно ограничить сверху. Если условия выполнены, то такая вершина убирается из маршрута. Описанный процесс может повторяться несколько раз, для чего каждый отрезок пути разбивается на подотрезки меньшей длины. После этого опять производится попытка выкинуть точки. И так далее. За счёт такого приёма маршрут становится локально оптимальным в том смысле, что никакой подпуть не может быть сокращён в некоторой области.

Ещё одна модификация маршрута вызвана тем, что в связи со структурой графа в маршруте могут появляться слишком острые углы (если в каком-нибудь контуре, образованном при смещении полигона, есть острый угол). В

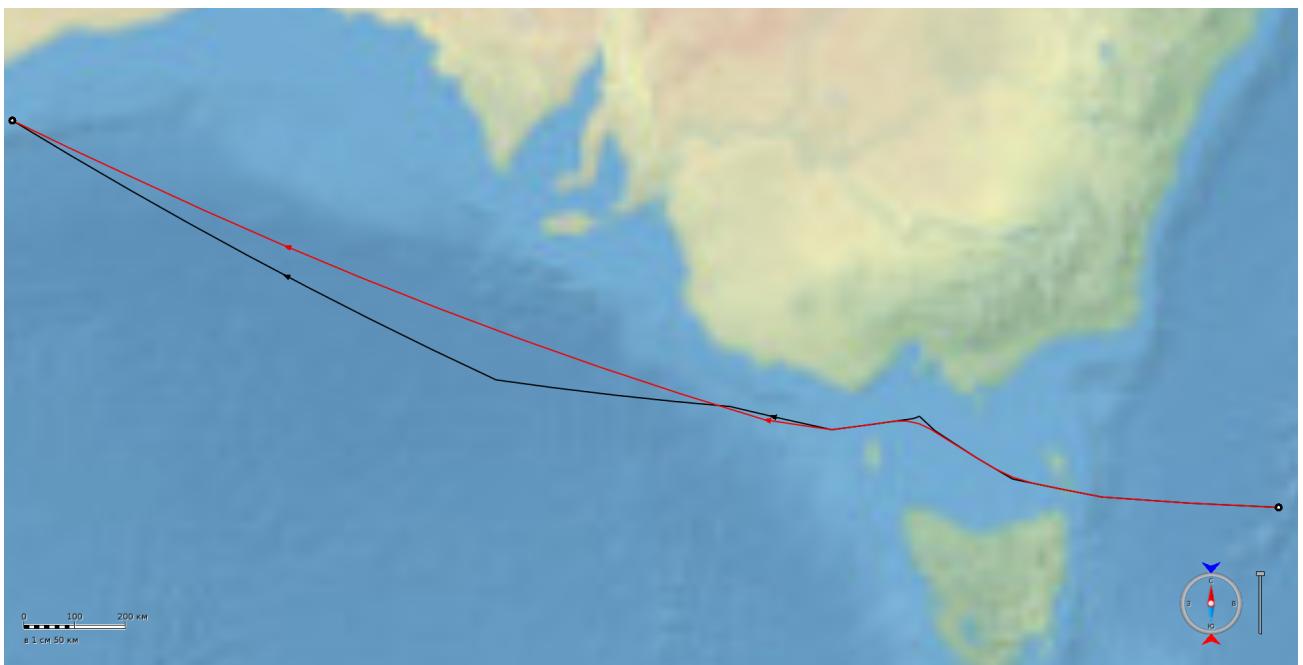


Рисунок 14 – Сокращение и сглаживание маршрута

реальных маршрутах, как правило, не бывает острых углов, поэтому выполняется сглаживание маршрута, то есть замена угла, величина которого меньше определённого значения, ломаной, аппроксимирующей дугу эллипса. При этом корректность такой замены проверяется по исходному полигону, не смещённому внутрь. Если ни один отрезок ломаной не пересекает границы исходного полигона, то замена действительно производится, иначе в маршруте остаётся острый угол.

На рисунке 14 продемонстрирован эффект от сокращения и сглаживания маршрута. Чёрной линией изображён кратчайший путь в графе, а красной — итоговый маршрут. Нетрудно видеть, что левая часть маршрута сократилась, а в правой части произошло сглаживание, что сделало маршрут более естественным.

2.3. Поиск нескольких маршрутов

Для поиска нескольких маршрутов используется итеративный алгоритм. На листинге 1 представлен псевдокод алгоритма. Первым делом выполняется поиск первого маршрута описанным выше способом, найденный маршрут добавляется в результирующее множество, запоминается его длина, приближённо равная длине кратчайшего пути. Затем, пока количество найденных маршрутов не достигнет максимально возможного, выполняются следующие действия: обновление весов рёбер в графе, поиск очередного маршрута, проверка

критерия остановки. Критерий остановки был сформулирован в формальной постановке задачи и состоит в том, что длина найденного маршрута должна быть не более чем в C_0 раз больше длины кратчайшего маршрута, и не один из имеющихся маршрутов не должен быть похожий на последний найденный. Функция $similar(p, q)$ проверяет, похожи ли маршруты в соответствии с формализацией, приведённой в постановке задачи. Практическая реализация этой функции описана в следующей главе.

Остановимся поподробней на процессе обновления весов. Процесс основывается на введении для каждой вершины графа некоторой величины, названной потенциалом. Чем больше потенциал вершины, тем сильнее увеличиваются веса инцидентных ей рёбер. Чем ближе вершина к найденному маршруту, тем сильнее должен увеличиваться вес инцидентных ей рёбер, чтобы находились непохожие маршруты. Для поиска потенциалов первым делом строится граф с фиктивной вершиной, из которой проводятся рёбра нулевого веса во все вершины очередного пути (функция `graph_with_fake` в псевдокоде). Вводится максимальное расстояние, равное $\frac{len}{C_1}$, где константа C_1 является параметром алгоритма. Для всех вершин, до которых длина кратчайшего пути в графе меньше этого расстояния, могут быть увеличены потенциалы. После этого используется алгоритм Дейкстры для поиска кратчайших расстояний из фиктивной вершины, при этом поиск заканчивается, когда достигнуто максимальное расстояние (функция `shortest_in_radius` в псевдокоде). Кратчайшее расстояние до непосещённых вершин принимается равным максимальному. Следующий вопрос состоит в вычислении потенциалов вершин по найденным расстояниям и обновлении весов рёбер по введённым потенциалам. Наиболее очевидным и простым решением является вычислять значение потенциала, используя убывающую линейную функцию от кратчайшего расстояния (например, $\frac{limit-dist}{A}$, где $limit$ — введённое выше максимальное расстояние, $dist$ — найденное кратчайшее расстояние от фиктивной вершины, $A > 0$ — некоторая константа), и прибавлять к весу ребра среднее арифметическое потенциалов инцидентных ему вершин. Однако такой подход обладает определённым недостатком.

Рассмотрим рисунок 15. На данном рисунке между крайней правой и крайней левой точкой есть два альтернативных пути. Предположим, что до этого были найдены длинные маршруты (хотя бы один), в окрестность которых

Листинг 1 – Псевдокод поиска нескольких маршрутов

```

function FIND_PATHS
    p, len  $\leftarrow$  find_path()
    paths  $\leftarrow \{p\}$ 
    min_len  $\leftarrow$  len
    while paths.size() < max_paths_count do
        update_weights(p, len)
        p, len  $\leftarrow$  find_path()
        if len >  $C_0 \cdot \min\_len$  then
            return paths
        end if
        for q in paths do:
            if similar(p, q) then
                return paths
            end if
        end for
        paths.insert(p)
    end while
end function

function UPDATE_WEIGHTS(p, len)
    g'  $\leftarrow$  graph_with_fake(p)
    limit  $\leftarrow \frac{\text{len}}{C_1}$ 
    d  $\leftarrow$  shortest_in_radius(g', fake, limit)
    for all v in p do
        d[v] =  $\frac{\text{limit}}{M}$ 
    end for
    for all v in vertices do
        potential =  $1 + (\max\_potential - 1) \cdot \frac{\text{limit} - d[v]}{\text{limit}}$ 
        potentials[v] = max(potentials[v], potential)
    end for
    for all e in edges do
        e.weight =  $e.\text{initial\_weight} \cdot \sqrt{potentials[e.\text{from}] \cdot potentials[e.\text{to}]}$ 
    end for
end function

```

попадают изображённые вершины. В этом случае, поскольку эти вершины находятся близко друг к другу, их потенциалы будут примерно равны некоторому C . Тогда, если использовать предложенный выше подход, то увеличение веса верхнего пути за счёт потенциалов будет на C больше, чем увеличение веса нижнего пути. Из-за этого будет выбран более длинный нижний путь. Таким образом, возникает проблема в том, что вес пути начинает зависеть от числа

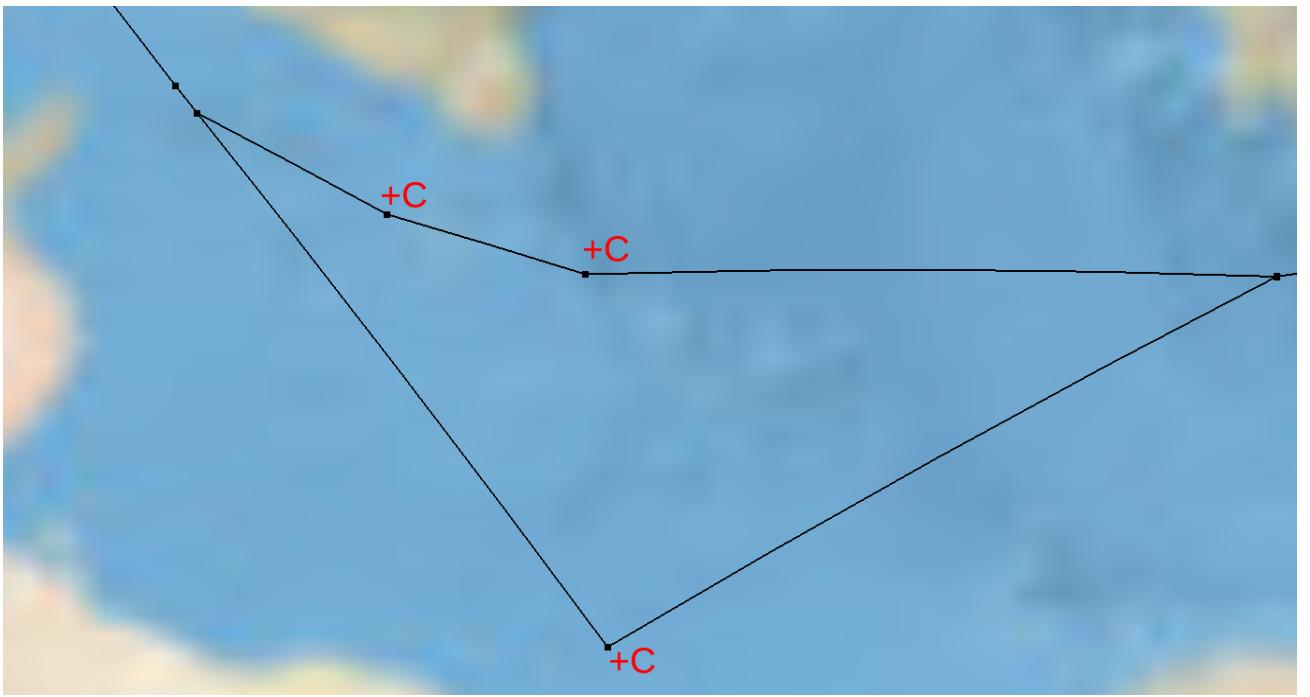


Рисунок 15 – Потенциалы должны быть множителями

вершин в нём. Более осмысленно использовать потенциалы как множители, например, домножать веса рёбер на среднее геометрическое потенциалов инцидентных вершин. В этом случае увеличение веса подпути при примерно равных потенциалах не будет зависеть от количества вершин. При этом значение потенциала в таком случае должно быть безразмерной величиной, и изначально для каждой вершины оно равно единице (что соответствует немодифицированным весам рёбер). Поскольку потенциал должен увеличивать вес ребра, то его значение не может быть меньше единицы. Для максимального значения потенциала вводится отдельный параметр *max_potential*. Как уже отмечалось, потенциал вершины v должен убывать с ростом расстояния от фиктивной вершины до неё ($d[v]$). Поэтому для вычисления потенциала используется эвристическая функция: $1 + (\max_potential - 1) \cdot \frac{\text{limit} - d[v]}{\text{limit}}$. Она удовлетворяет следующим свойствам:

- Значение потенциала принадлежит отрезку $[1..max_potential]$.
- Значение потенциала убывает с ростом расстояния от фиктивной вершины.
- Для вершин, расстояние до которых равно максимальному, значение потенциала минимально.

Описанный выше способ обновления весов рёбер обладает недостатком, проиллюстрированным на рисунке 16. Проблема заключается в том, что для



Рисунок 16 – Маршруты несущественно разошлись



Рисунок 17 – Маршруты совпадают и оптимальны в начале

вершин найденного маршрута значение потенциалов максимально (поскольку расстояние от фиктивной вершины равно нулю). В связи с этим следующий найденный маршрут будет проходить на небольшом расстоянии от предыдущего. При этом он станет длиннее, но не станет непохожим в смысле сформулированного ранее критерия. Разумеется, такое поведение является неестественным и не удовлетворяет неформальной постановке задачи. Поэтому кратчайшие расстояния от фиктивной вершины до вершин на маршруте принимаются равными $\frac{limit}{M}$, где $M > 1$ — параметр алгоритма. За счёт этого в приведённом примере южная часть белого маршрута совпала с той же частью красного, проходящего по кратчайшему пути (рисунок 17).

Наконец, последний вопрос состоит в обновлении потенциалов, после того как найден очередной маршрут. Понятно, что потенциалы должны как-то накапливаться, чтобы учитывать влияние всех предыдущих маршрутов и не допускать прохождение нового маршрута вблизи старых. Например, можно для каждой вершины в качестве нового потенциала брать сумму или произведение имеющегося потенциала и вновь посчитанного. Также можно брать максимум. На рисунках (18–23) проиллюстрированы различия между двумя вариантами: перемножением и максимумом. Красные квадраты визуализируют значения потенциалов: чем больше и ярче квадрат, тем больше потенциал. После нахождения первого маршрута потенциалы, очевидно, распределены одинаково (рис. 18 и 19). Затем, если перемножать потенциалы, то для вершин в окрестностях начальной и конечной точек они оказываются сильно больше, чем для остальных вершин (рис. 20). Если же брать максимум, то распространение потенциалов происходит равномерно: во всех точках, расположенных близко к какому-либо найденному маршруту, потенциал оказывается большой (21). Рисунки (22–23) демонстрируют, что в дальнейшем описанный эффект лишь усиливается. В результате проведённого эксперимента при использовании максимума для обновления потенциалов было найдено на один маршрут больше (рис. 24–25). Поэтому в предложенном алгоритме используется максимум для обновления потенциалов.

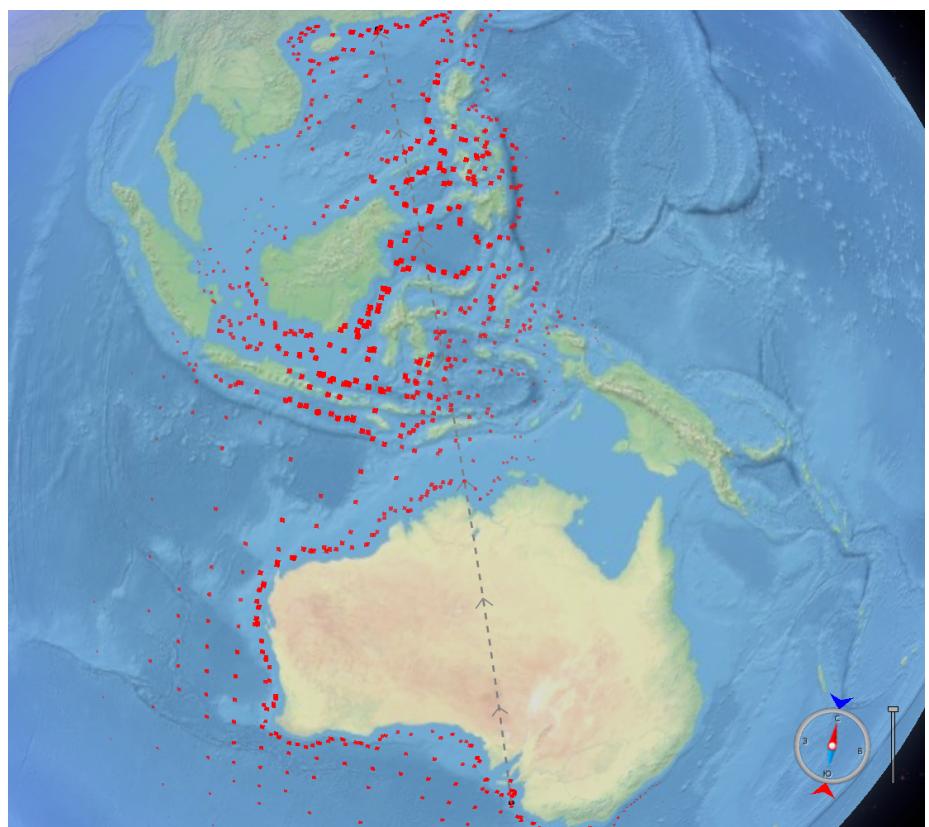


Рисунок 18 – Обновление потенциалов: умножение (1)

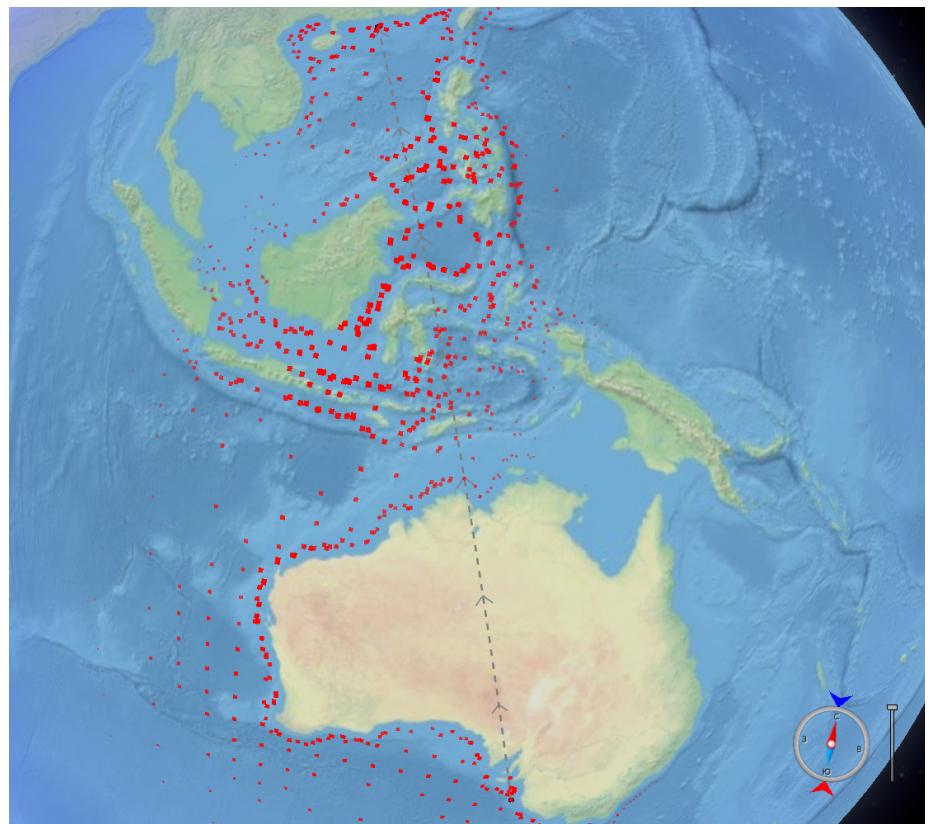


Рисунок 19 – Обновление потенциалов: максимум (1)

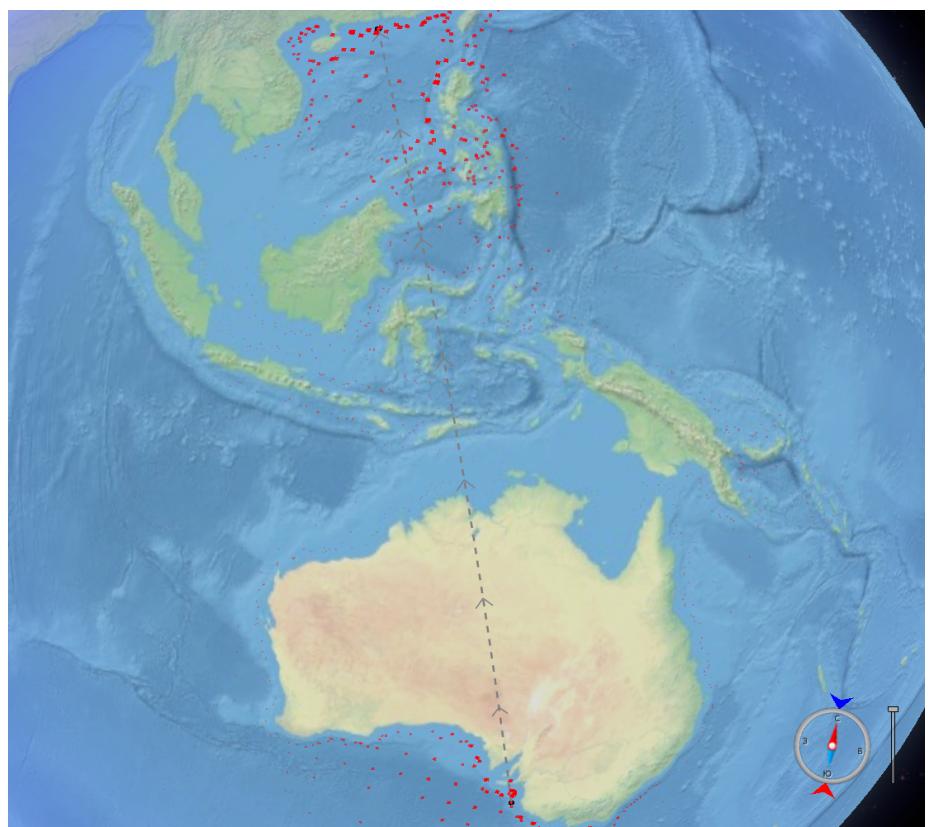


Рисунок 20 – Обновление потенциалов: умножение (2)

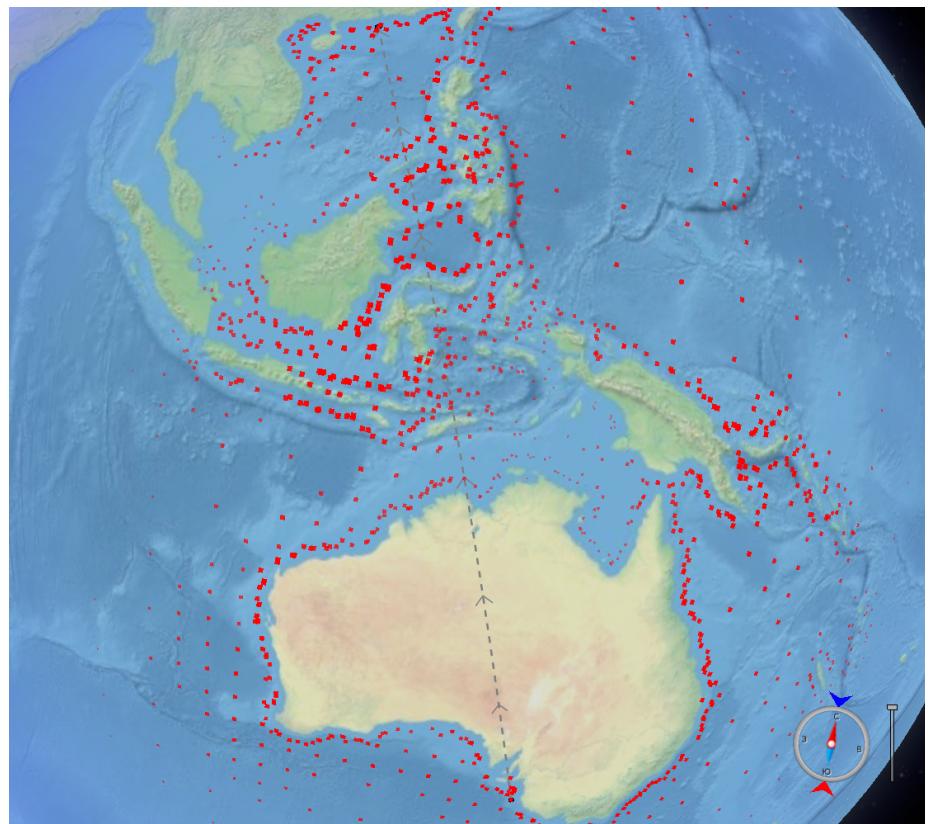


Рисунок 21 – Обновление потенциалов: максимум (2)

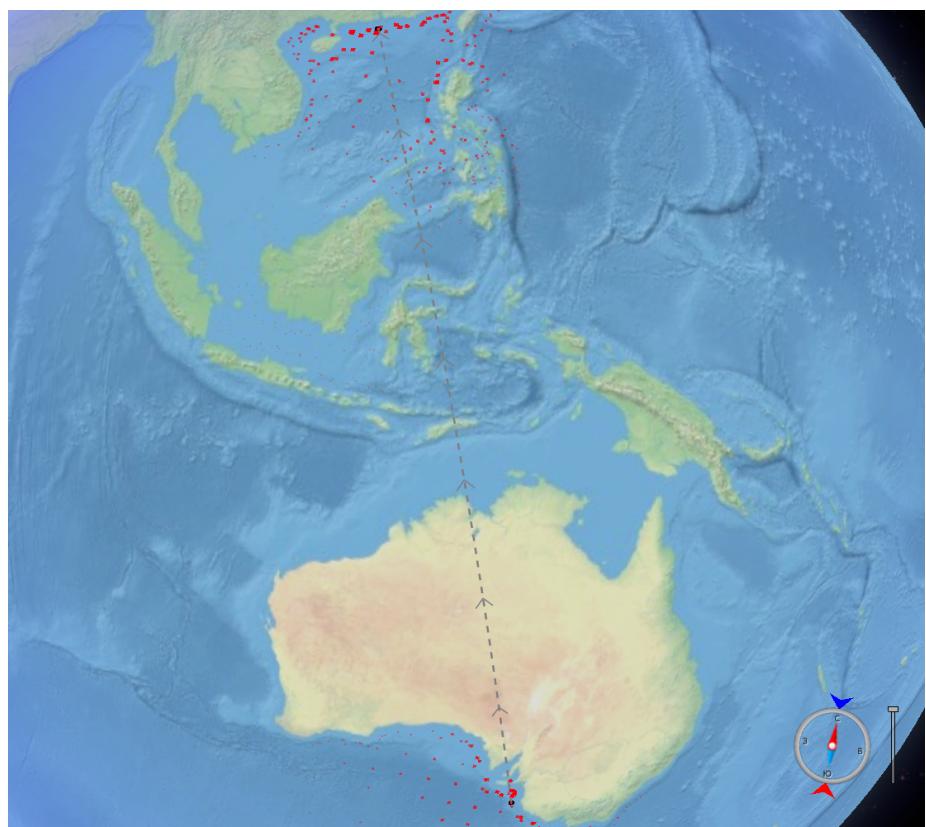


Рисунок 22 – Обновление потенциалов: умножение (3)

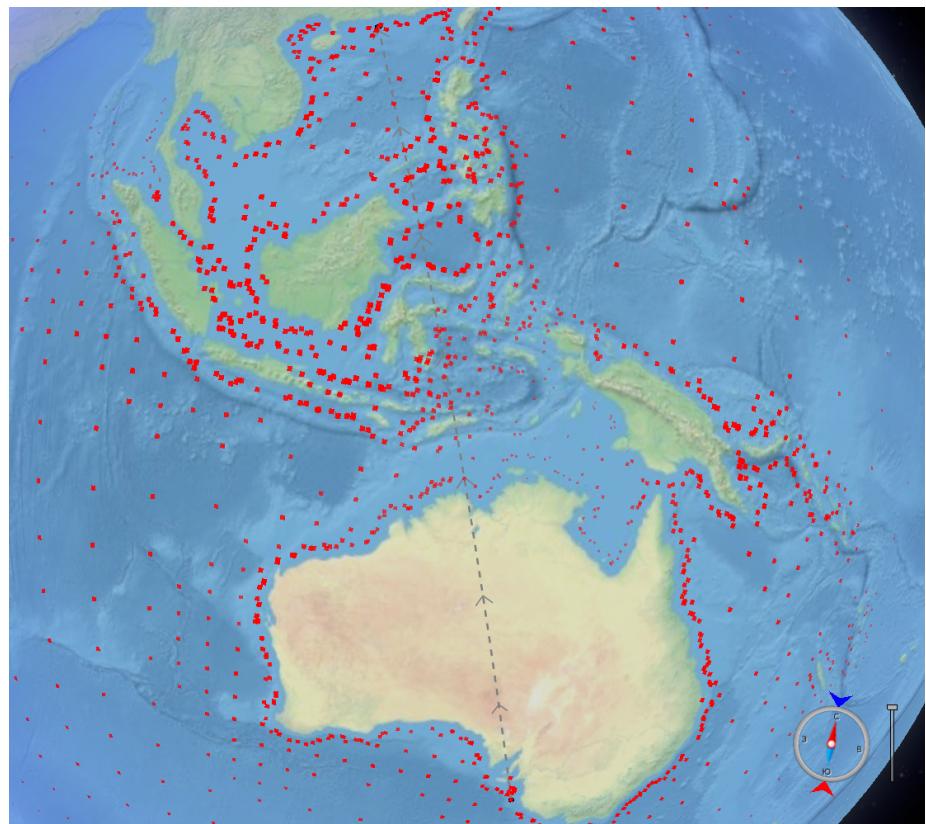


Рисунок 23 – Обновление потенциалов: максимум (3)

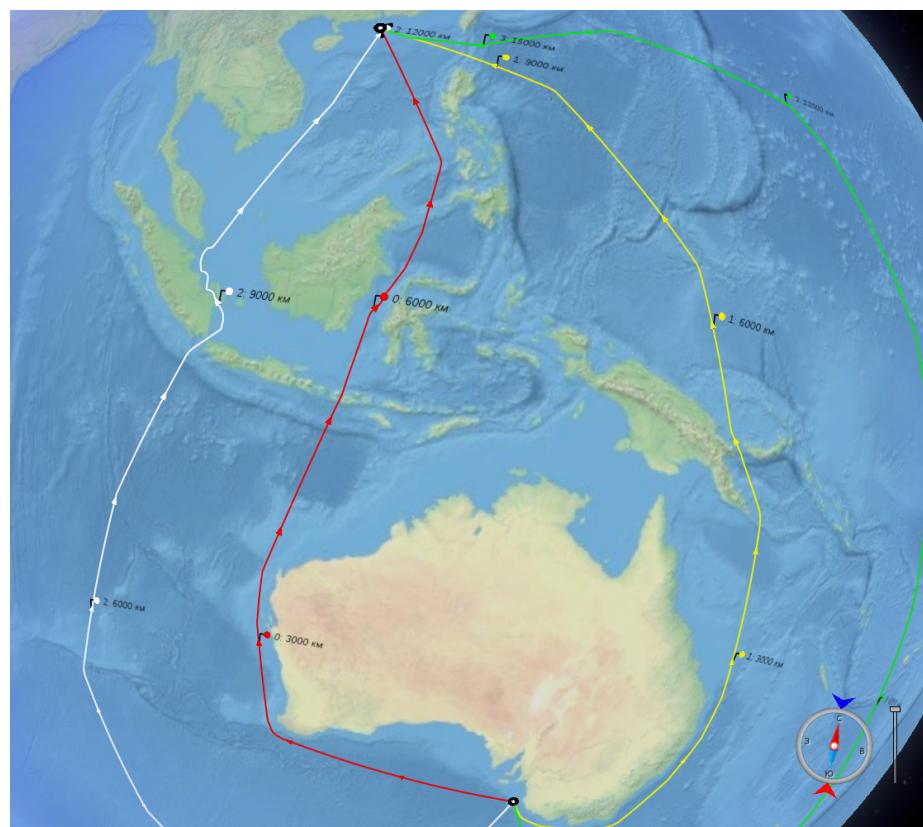


Рисунок 24 – Найденные маршруты при использовании умножения

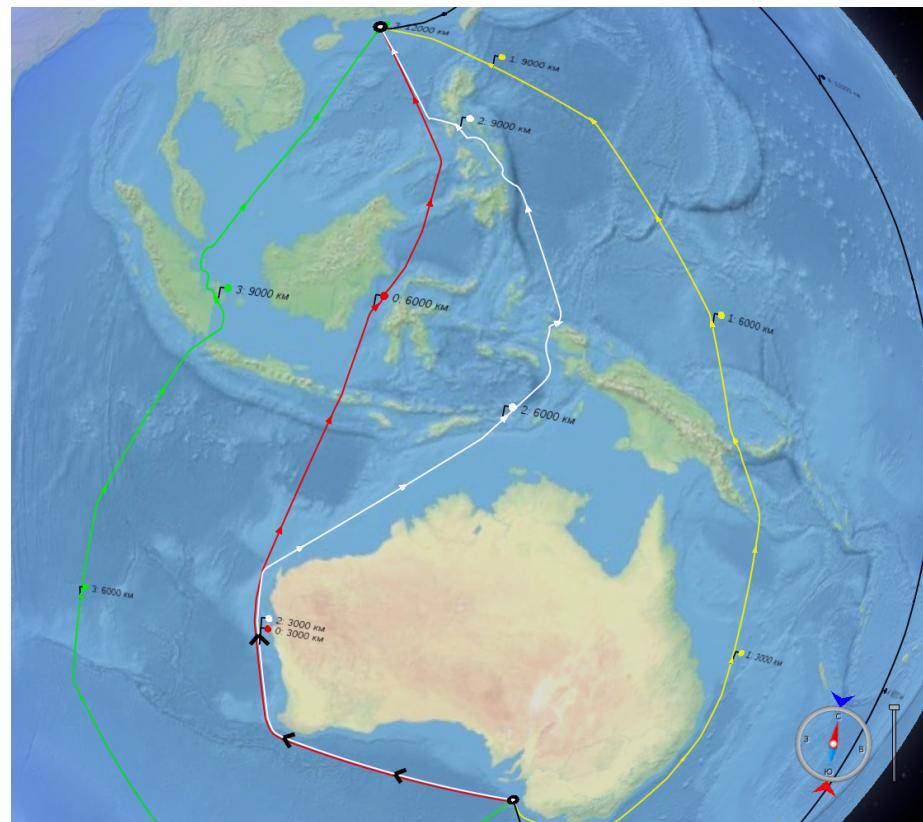


Рисунок 25 – Найденные маршруты при использовании максимума

ГЛАВА 3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ И РЕЗУЛЬТАТЫ

В данной главе рассмотрены некоторые аспекты практической реализации предложенного алгоритма и визуализации. Программный модуль, решающий поставленную задачу, был реализован на языке C++ в рамках имеющегося фреймворка для решения задач. В заключение приводятся основные результаты работы.

3.1. Предобработка

Для решения большей части задач по предобработке данных использованы библиотеки ЗАО «Кронштадт Технологии». Чтение исходных данных и классификация объектов карты осуществляется с помощью имеющихся утилит импорта картографических данных. Для выполнения геометрических операций, таких как объединение контуров, упрощение полилиний, построение straight skeleton, смещение полигона и т. д., используется библиотека геометрических примитивов и алгоритмов ЗАО «Кронштадт Технологии». Также в этой библиотеке есть специальная структура данных, названная *Contours Set*, для эффективной проверки пересечения отрезка с полигоном. Эта структура данных активно используется для всех проверок корректности рёбер (при добавлении рёбер, сглаживании маршрута, упрощении цепочек рёбер straight skeleton’а).

Определённую сложность представляет добавление рёбер через 180-ый меридиан. Для этого имеющийся полигон, ограничивающий воду, объединяется со своей копией, сдвинутой влево на длину плоскости проекции. В результате получается полигон, изображённый на рисунке 26. По такому полигону можно легко проверять корректность ребра, проходящего через 180-ый меридиан, переместив его восточную точку влево на длину плоскости проекции.

Для тестирования работы алгоритма использовались свободно распространяемые данные OpenStreetMap [14]. Получившийся граф содержит 13344 вершины и 124324 ребра.

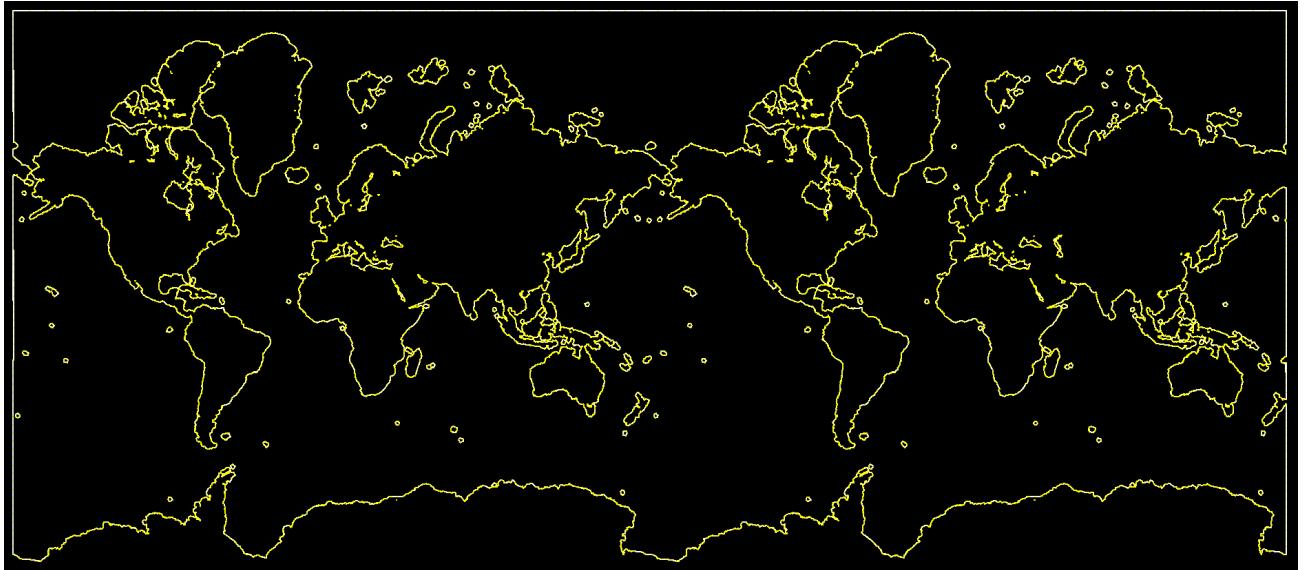


Рисунок 26 – Полигон, объединённый со смещённой копией

3.2. Вычисление метрик

Поскольку в процессе работы алгоритма для всех пар путей проверяется критерий похожести, необходимо добиться высокой скорости вычисления метрик на маршрутах. Для вычисления обеих метрик используются длины кратчайших расстояний между парами вершин в графе. Длины кратчайших расстояний между всеми парами вершин могут быть найдены с помощью алгоритма Флойда [15]. Однако алгоритм Флойда работает за время порядка $\Theta(n^3)$, что неприемлемо для графа, содержащего 13344 вершины. Также нетрудно заметить, что знать длины кратчайших путей в графе между всеми парами вершин не нужно, поскольку в определении обеих метрик присутствуют лишь длины кратчайших путей для всех пар вершин, принадлежащих первому и второму маршруту. Поэтому можно для каждой вершины первого пути использовать алгоритм Дейкстры и останавливать поиск при достижении какой-либо вершины второго пути, затем сделать то же самое для вершин второго пути. Поскольку в этом случае время работы сильно зависит от конкретных маршрутов (не только от числа вершин, но и от их расположения, поскольку каждый поиск алгоритмом Дейкстры может заканчиваться очень рано или, наоборот, спустя большое число итераций), то теоретические оценки времени работы такого подхода не несут практической информации. На практике же при таком способе вычисления метрик поиск маршрутов, как правило, занимал 2–10 секунд. Однако поставленная задача требует выполнять запросы за время, не превышающее одной секунды.

Для улучшения производительности было сделано две оптимизации. Нетрудно заметить, что если в процессе вычисления первой метрики находится пара вершин из первого и второго маршрута, между которыми длина кратчайшего пути в графе больше порогового значения, то можно заключить, что маршруты непохожи. При вычислении второй метрики верен аналогичный факт. Таким образом, поскольку метрики считаются только для проверки критерия похожести маршрутов, то вычисление можно останавливать, как только найдена пара вершин, гарантирующая непохожесть маршрутов.

Вторая оптимизация затрагивает сам процесс вычисления метрик. Вместо поиска кратчайших расстояний с помощью алгоритма Дейкстры из всех вершин пути используется другой подход. Сначала строится граф с фиктивной вершиной (f), из которой проведены рёбра нулевого веса во все вершины первого маршрута (P). Затем с помощью алгоритма Дейкстры находятся длины кратчайших расстояний из фиктивной вершины до всех вершин второго маршрута (Q) (поиск заканчивается, когда посещены все вершины, принадлежащие Q). Поскольку из фиктивной вершины проведены рёбра нулевого веса во все вершины, принадлежащие P , и только в них, то кратчайший путь от неё до вершины $v \in Q$ будет иметь вид $fu \dots v$ для какой-то вершины $u \in P$, и его длина будет равна $\rho_g(u, v)$. Как и в определении метрик за $\rho_g(u, v)$ обозначена длина кратчайшего пути между вершинами u и v в графе. Нетрудно заметить, что в этом случае $\rho_g(u, v) = \min_{w \in P} \rho_g(w, v)$. Действительно, пусть существует вершина $u' \in P$, такая что $\rho_g(u', v) < \rho_g(u, v)$. Обозначим кратчайший путь в графе между u' и v как $R = u'r_0r_1 \dots r_nv$. Тогда длина пути $fu'r_0 \dots r_nv$ равна $\rho_g(u', v) < \rho_g(u, v) = \rho_g(f, v)$, что противоречит тому, что $\rho_g(f, v)$ является длиной кратчайшего пути из f в v . Значит, предположение о существовании u' неверно. Таким образом, сразу для всех вершин $v \in Q$ находится величина $\min_{u \in P} \rho_g(u, v)$ за один обход алгоритма Дейкстры. Аналогичный поиск можно выполнить, добавив фиктивную вершину и рёбра нулевого веса во все вершины маршрута Q , тем самым найдя для всех $u \in P$ величину $\min_{v \in Q} \rho_g(u, v)$. После этого достаточно взять максимум всех найденных величин, который будет значением метрики. При этом, как было сказано ранее, если хоть одно из найденных значений больше определённого порога (описанного в критерии похожести), то вычисление сразу же закончить, заключив, что маршруты непохожи. Для вычисления второй метрики помимо максимального минимального

расстояния между парами вершин нужно находить расстояние до ближайшей вершины в мире. Данный поиск был реализован простым перебором, работающим в худшем случае за время порядка $\Theta(l_1 \cdot l_2)$, где l_1 и l_2 — длины первого и второго маршрута соответственно. Однако, во-первых, вторая метрика вычисляется, как правило, реже, чем первая, во-вторых, зачастую не нужно просматривать все вершины, поскольку пара вершин, для которых выполнен критерий непохожести, встречается раньше. Помимо этого при вычислении второй метрики не рассматриваются пары вершин, ребро между которыми не пересекает ни одно препятствие, поскольку между такими вершинами можно пройти по прямой. В результате проведённых оптимизаций время обработки запроса существенно уменьшилось и стало составлять порядка 0,2–0,8 секунды. При этом профилирование показало, что основную часть времени стало занимать не вычисление метрик, а сокращение маршрутов. Поскольку итоговое время работы удовлетворяет поставленным требованиям, дальнейшие оптимизации не проводились.

3.3. Запретные зоны

Помимо поиска маршрутов по имеющейся карте, важно предоставить пользователю возможность самостоятельно добавлять и удалять запретные зоны, которые являются дополнительными полигональными препятствиями. Во-первых, в реальной жизни действительно в различных регионах может быть запрещено проплыть (например, из-за военных учений). Во-вторых, за счёт этого пользователь может влиять на семейство находимых маршрутов, например, запрещать маршрут, который по тем или иным причинам не подходит. Такая важная функциональность, однако, реализуется очень просто. Для этого используется обёртка над графом, хранящая уже упомянутую структуру *Contours Set* с запретными зонами. При запросе рёбер они фильтруются по предикату, проверяющему, что ребро не пересекает запретные контуры. Нетрудно заметить, что для предложенного алгоритма такие запретные зоны, по сути, ничем не отличаются от исходных полигональных препятствий, поскольку они просто запрещают определённые рёбра в графе, тем самым увеличивая длину кратчайшего пути в графе между определёнными вершинами. Таким образом, способ обхода запретных зон также будет влиять на похожесть маршрутов. На рисунке 27 представлен пример маршрутов при наличии запретной зоны.

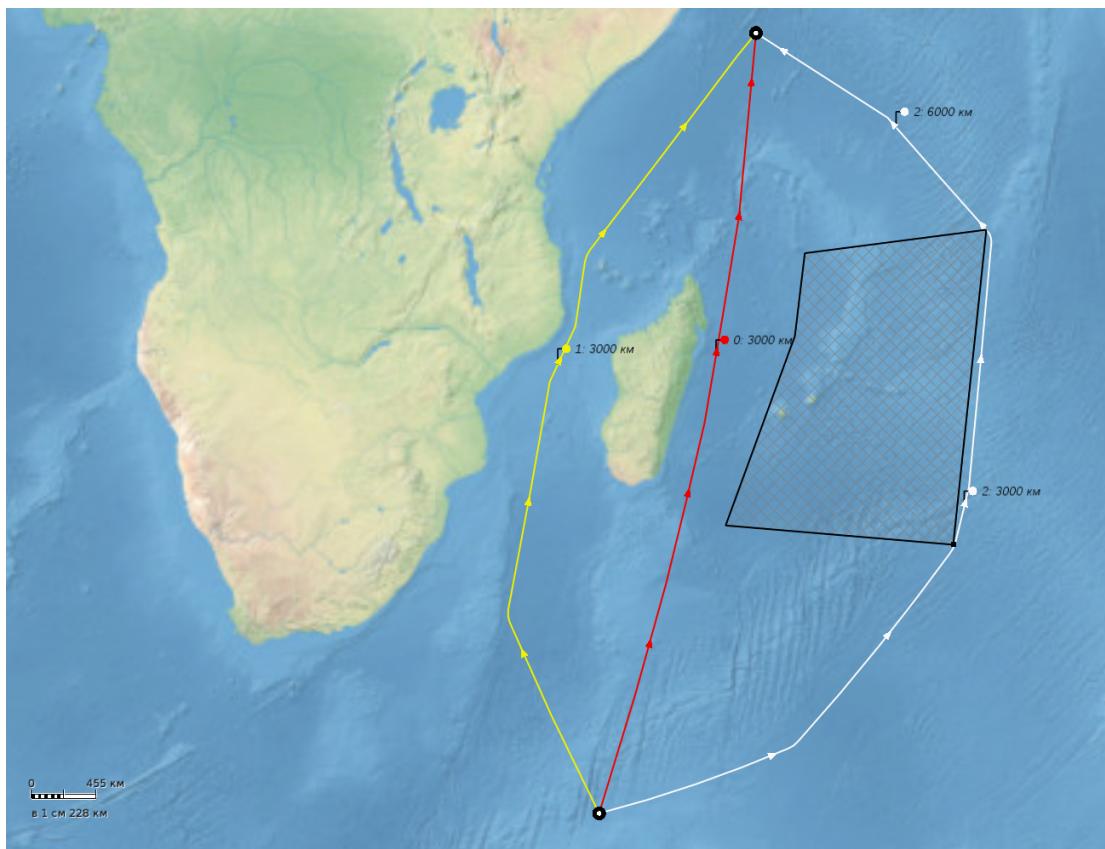


Рисунок 27 – Маршруты при наличии запретной зоны

3.4. Визуализация

Для визуализации маршрутов была использована библиотека, разработанная в ЗАО «Кронштадт Технологии» и основанная на OpenGL [16]. Данная библиотека позволяет отображать различные примитивы на плоскости и сфере, в том числе полилинии на сфере. Основная проблема при визуализации семейств маршрутов состоит в том, что некоторые маршруты накладываются друг на друга. Решение такой проблемы состоит в смещении перекрывающихся частей на небольшое расстояние в пикселях. Для этого выделяются перекрывающиеся части имеющихся маршрутов (имеющие одинаковые рёбра) с помощью сохранения рёбер в хеш-таблицу. При визуализации каждое ребро маршрута рассматривается отдельно. Если оно принадлежит более чем одному маршруту, то берётся порядковый номер текущего пути в отсортированном списке маршрутов, содержащих данное ребро, и вычисляется величина смещения путём домножения этого номера на текущий размер пикселя в мире. Затем каждая вершина ребра смещается на эту величину в направлении бисектрисы, проведённой в треугольнике, образованном текущей вершиной и её соседними

вершинами. Если одной из соседних вершин не существует (вершина является начальной или конечной точкой), то в качестве направления берётся нормаль к соответствующему ребру. В следующем разделе (3.5) приведены рисунки с примерами работы алгоритма, на которых можно видеть смещённые маршруты.

3.5. Результаты

По итогам данной работы получены следующие результаты:

- Исследованы имеющиеся подходы к решению аналогичных задач, выявлены и продемонстрированы их недостатки.
- Дано формальное описание поставленной задачи.
- Предложен эвристический алгоритм для решения данной задачи.
- Реализован программный модуль, решающий поставленную задачу в режиме реального времени.
- Получено заключение экспертов ЗАО «ОСК-Транзас» о том, что находимые маршруты удовлетворяют поставленным требованиям. Эвристики, разработанные для поиска различных маршрутов, признаны эффективными и разумными. Алгоритм и критерии, заложенные в программный модуль, рекомендованы для решения задач поддержки принятия решения в навигационно-тактических системах морского назначения.
- Разработанный программный модуль внедрён в имеющийся 3D-клиент ЗАО «Кронштадт Технологии».

Для примера проведём сравнение маршрутов, находимых разработанным алгоритмом, с маршрутами, находимыми алгоритмом [4]. На рисунках 28 и 29 показана ситуация, в которой алгоритм, описанный Лимом и Кимом, нашёл лишь один маршрут, а алгоритм, описанный в данной работе, нашёл три различных маршрута. В ситуации, показанной на рисунках 30 и 31, оба алгоритма нашли три маршрута. Однако в первом случае маршруты, не являются существенно различными, хоть и обходят препятствия разными способами, поскольку размеры этих препятствий невелики. К тому же найденные маршруты не являются локально оптимальными, так как в конце пути имеются некоторые различия. Во втором случае было найдено три существенно разных маршрута, различающихся способами обхода материков. В алгоритме Кима и Лима увеличение весов происходит только для рёбер, принадлежащих найденному маршруту. Данный приём хорошо работает при поиске маршрутов в



Рисунок 28 – Найден лишь один маршрут



Рисунок 29 – Найдено три различных маршрута

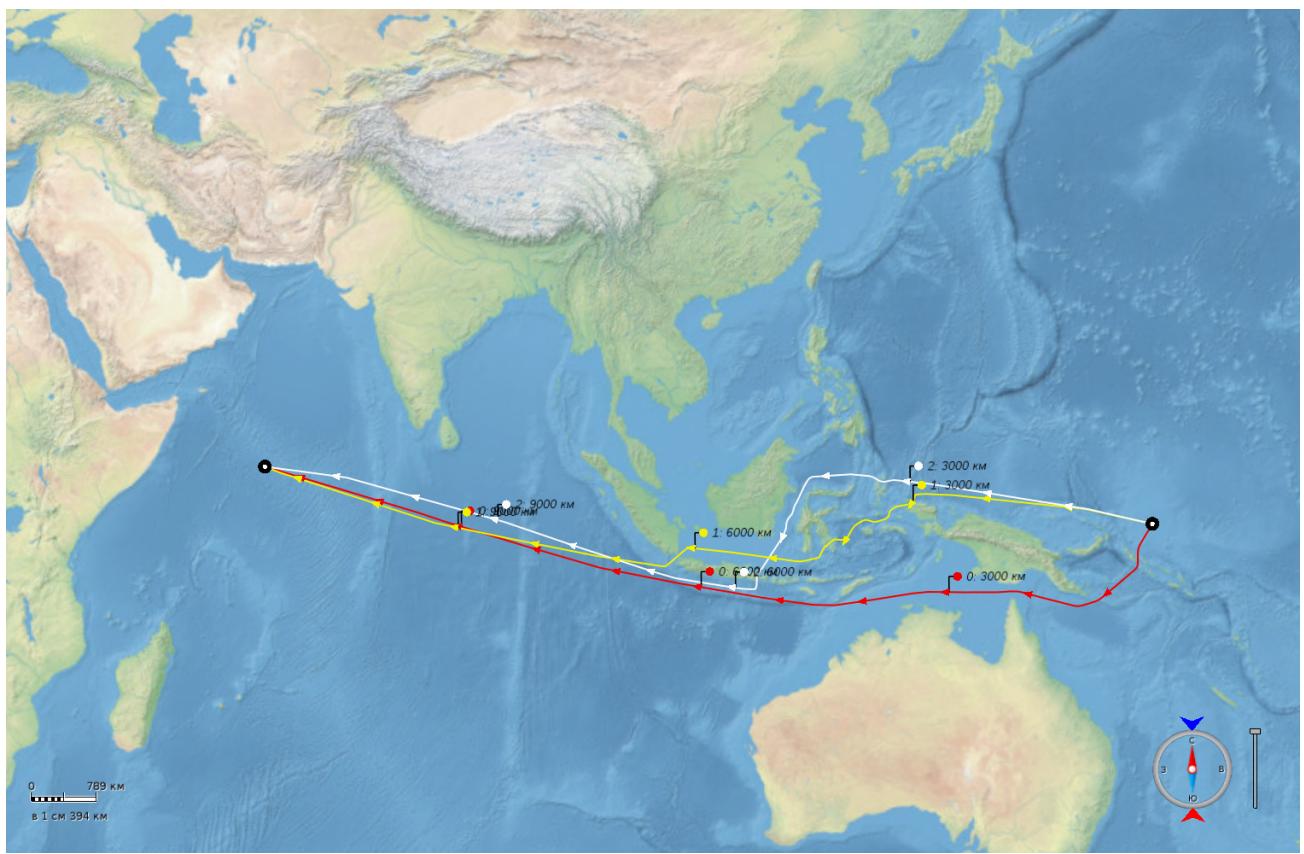


Рисунок 30 – Найдены похожие маршруты



Рисунок 31 – Найдены маршруты с разными способами обхода материков

графе дорог, поскольку в таком случае для непохожести маршрутов достаточно просто пройти по разным дорогам. Однако для поиска маршрутов по воде он, как было показано, в большинстве случаев не подходит. В то же время в предложенном алгоритме обновление весов происходит в области, а критерий остановки учитывает гораздо больше факторов. Таким образом, показано, что разработанный алгоритм лучше подходит для решения поставленной задачи, чем алгоритм, описанный в [4], хотя общая идея у них похожа.

В заключение приведено несколько рисунков, демонстрирующих результаты работы алгоритма в 3D-клиенте (рисунки 32, 33).

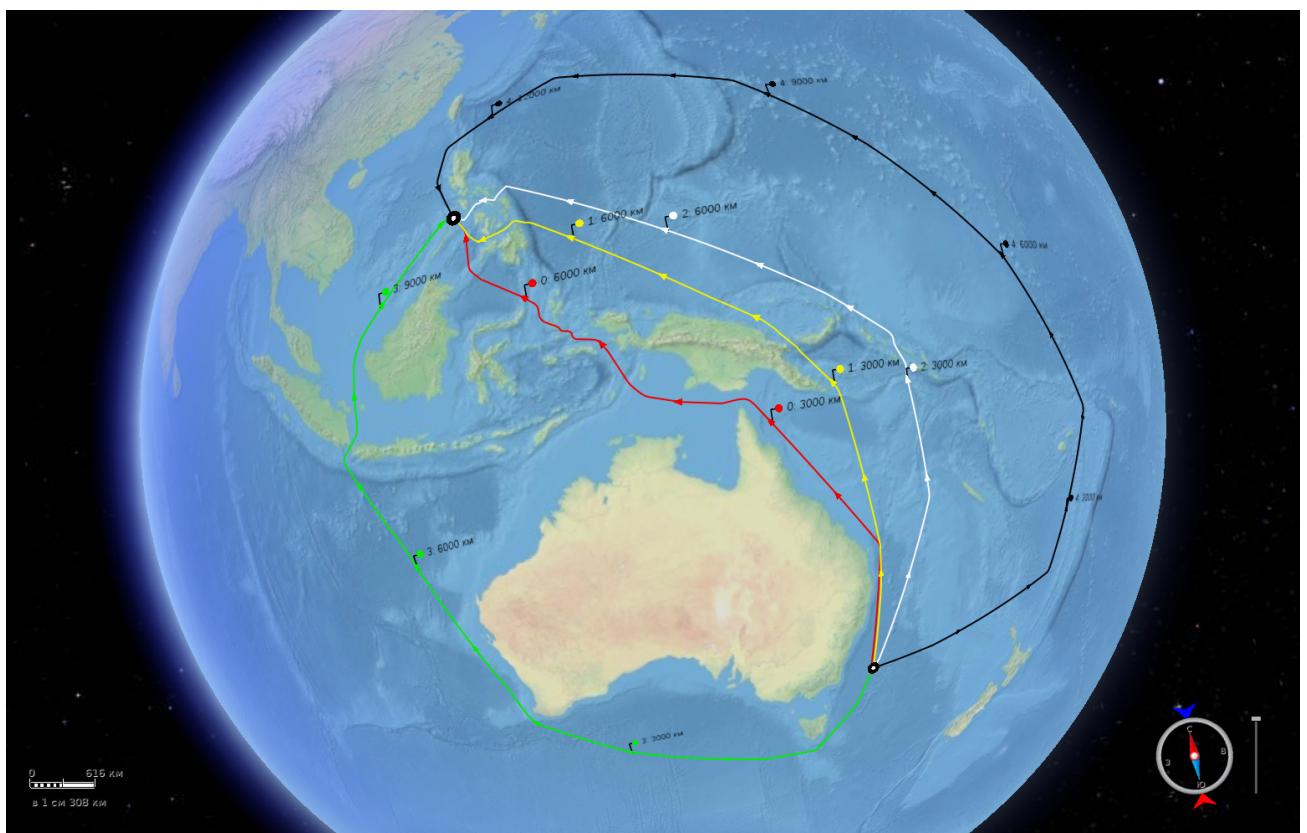


Рисунок 32 – Маршруты на сфере

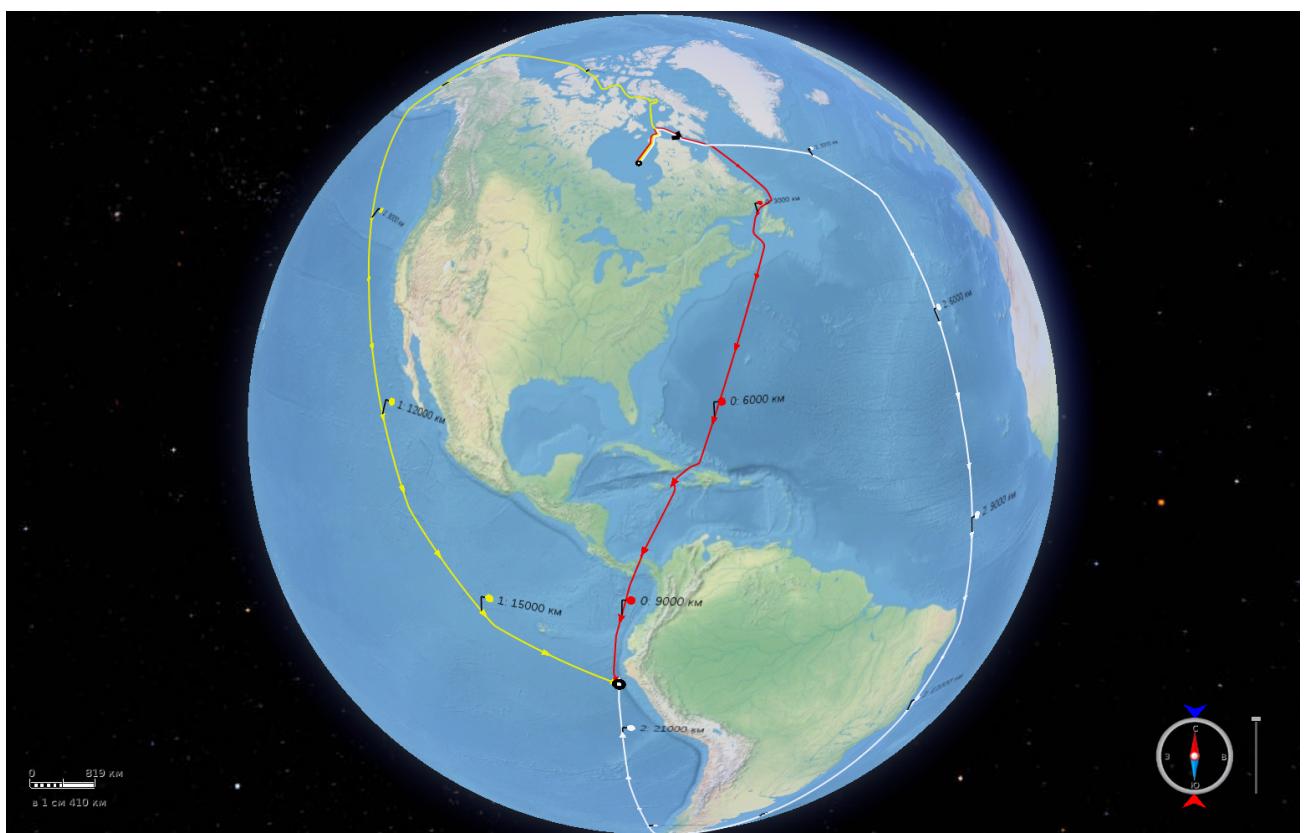


Рисунок 33 – Маршруты на сфере

ЗАКЛЮЧЕНИЕ

Как было показано в данной работе, задача построения семейств оптимальных маршрутов кораблей представляет достаточный практический интерес. Были рассмотрены подходы к решению аналогичных задач, выявлены их недостатки. После этого было formalизовано описание семейства оптимальных маршрутов. На основе формальной постановки задачи были предложены различные эвристики для её решения. В результате был разработан алгоритм для поиска семейств маршрутов и реализован в виде программного модуля для решения данной задачи в режиме реального времени.

Стоит заметить, что предложенный алгоритм является преимущественно эвристическим, большинство принятых решений не имеют строгих обоснований и основаны на различных исследованиях. Для оценки проделанной работы разработанный программный модуль был отправлен на экспертизу в ЗАО «ОСК-Транзас». Разработанные эвристики признаны эффективными и разумными. Отмечено, что находимые маршруты соответствуют заявленным требованиям. Эксперты рекомендуют использовать алгоритм и критерии, заложенные в реализованный программный модуль, для решения задач поддержки принятия решения в навигационно-тактических системах морского назначения.

Также стоит отметить возможные направления дальнейших работ для улучшения имеющегося решения:

- В алгоритме активно применяется алгоритм Дейкстры для различных целей (поиск маршрута, вычисление потенциалов, вычисление метрик). Возможно, имеет смысл рассмотреть вопрос использования алгоритма A^* с целью потенциального улучшения производительности.
- Более сложной является задача поиска маршрутов при заданных запретных зонах в зависимости от времени. В этом случае известно, что в некоторые временные диапазоны в определённых местах будет нельзя плавать. Требуется находить маршруты, учитывая подобные ограничения.

Разработанный программный модуль был внедрён в имеющееся программное обеспечение ЗАО «Кронштадт Технологии».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Computational geometry / M. De Berg [et al.]. — Springer, 2000. — Chap. 13, 15.
- 2 *Eppstein D.* Finding the k shortest paths // SIAM Journal on computing. — 1998. — Vol. 28, no. 2. — P. 652–673.
- 3 *Yen J. Y.* Finding the k shortest loopless paths in a network // management Science. — 1971. — Vol. 17, no. 11. — P. 712–716.
- 4 *Lim Y., Kim H.* A shortest path algorithm for real road network based on path overlap // Journal of the Eastern Asia Society for Transportation Studies. — 2005. — Vol. 6. — P. 1426–1438.
- 5 *Dial R. B.* A probabilistic multipath traffic assignment model which obviates path enumeration // Transportation research. — 1971. — Vol. 5, no. 2. — P. 83–111.
- 6 *Ma J., Fukuda D., Schmöcker J.-D.* A fast node-directed multipath algorithm: Dijkstra-Hyperstar.
- 7 Computational geometry / M. De Berg [et al.]. — Springer, 2000. — Chap. 15. P. 325.
- 8 *Dijkstra E. W.* A note on two problems in connexion with graphs // Numerische mathematik. — 1959. — Vol. 1, no. 1. — P. 269–271.
- 9 *Hart P. E., Nilsson N. J., Raphael B.* A formal basis for the heuristic determination of minimum cost paths // Systems Science and Cybernetics, IEEE Transactions on. — 1968. — Vol. 4, no. 2. — P. 100–107.
- 10 *Thomas P. D.* Conformal projections in geodesy and cartography. Vol. 4. — US Government Printing Office, 1952.
- 11 *Douglas D. H., Peucker T. K.* Algorithms for the reduction of the number of points required to represent a digitized line or its caricature // Cartographica: The International Journal for Geographic Information and Geovisualization. — 1973. — Vol. 10, no. 2. — P. 112–122.
- 12 *Aichholzer O., Aurenhammer F.* Straight skeletons for general polygonal figures in the plane. — Springer, 1996.

- 13 *O'Connell D. P., Shearer I. A.* The international law of the sea. — Oxford University Press, USA, 1984.
- 14 URL: www.openstreetmap.org.
- 15 *Floyd R. W.* Algorithm 97: shortest path // Communications of the ACM. — 1962. — Vol. 5, no. 6. — P. 345.
- 16 URL: <https://www.opengl.org/>.