САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Направление: 02.03.02 «Фундаментальная информатика и информационные технологии»
ООП: «Программирование и информационные технологии»

Чакляров Георгий Олегович

Выпускная квалификационная работа

Поиск оптимального маршрута в условиях динамически меняющихся ограничений

Научный
руководитель: к.ф.-м.
н., доцент кафедры
КТС Погожев Сергей
Владимирович

Санкт-Петербург
2022

Содержание

Введение

Задача поиска оптимального маршрута в условиях динамически меняющихся ограничений возникает практически повсеместно. Особенно в наши дни, когда ведутся активные наработки в области искусственного интеллекта: создание бытовых роботов-пылесосов, автопилотов (к примеру, компания «Яндекс» уже запустила экспериментальные такси без водителя в Москве [1]) и т.п. В этой же работе будет рассмотрена задача, относящаяся к одной из важнейших сфер человеческой деятельности – к судоходству.

Важнейшим фактором конкурентоспособности в морской отрасли является минимизация расхода топлива на судоходных маршрутах. В то же время, это согласуется с усилением общемировой тенденции к сокращению выбросов вредных веществ в атмосферу в рамках смягчения последствий изменения климата (например, глобального потепления) [2]. Оба этих требования могут быть удовлетворены посредством поиска оптимального маршрута. Под критериями оптимальности можно понимать очень многое: время в пути (за опоздание или слишком раннее прибытие может быть наложен штраф), расход топлива, минимизация выбросов, безопасность судна и груза (боковая качка может привести к повреждениям судна или потере груза) и комфорт пассажиров (особенно страдающих морской болезнью). Выбор конкретных критериев зависит от поставленной задачи.

Кроме того, в последние годы в академических и в коммерческих кругах значительно возросло внимание к так называемым **метеорологическим маршрутам судов** (англ. *weather routing*). Основной задачей в этой области является поиск оптимального пути и скорости движения для заданного рейса в условиях динамически меняющихся ограничений (погодных условий, таких как ветер, высота и направление волн и т.п.). Данную тенденцию можно заметить на рис. 1, где отображено изменение количества статей в области weather routing за последние 60 лет [3].

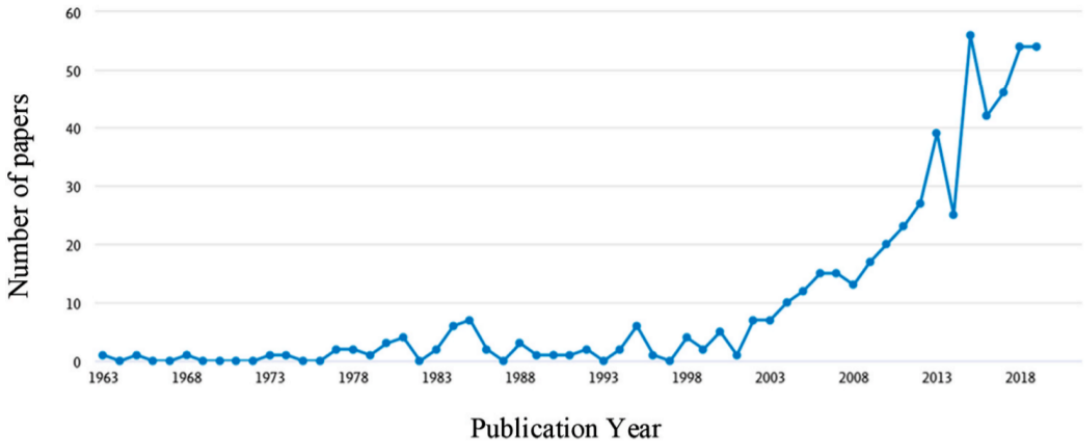

Рис. 1. Повышение интереса к области weather routing [3]

Большинство существующих разработок в этой области основывается либо на классических алгоритмах поиска кратчайших путей в графе (таких как алгоритм Дейкстры [4, 5, 6, 7], A* [2, 8]), либо на оптимизационных и математических алгоритмах: метод динамического программирования [4], метод многокритериальной оптимизации [9, 10], задача вариационного исчисления [6, 9], задача нелинейного программирования [4, 11], метод изохрон [4, 12], эволюционный алгоритм [4, 9, 10]. Кроме того, в последние годы также возросло количество приложений искусственного интеллекта и машинного обучения [3]. В работе [13] упоминается, что в некоторых системах планирования маршрутов используется метод Монте-Карло.

Однако не было найдено ни одного исследования, опирающегося на идеи гексагональной растеризации карты, математической морфологии и поиске в ширину. Вследствие чего этот подход к решению проблемы weather routing рассматривается в данной работе.

Постановка задачи

Дана некоторая прямоугольная карта местности с заданными длиной $width$ и шириной $height$. На ней присутствуют статические (т.е. не меняющимися со временем) ограничения, представляющие из себя набор из K областей в виде многоугольников:

$$S_n, n = 1,2, \dots, K.$$

На карте задаются точка старта (x_s, y_s) и точка финиша (x_f, y_f). Также на карте присутствуют сведения о динамических (т.е. меняющихся со временем) ограничениях – наборе из M областей в виде многоугольников

$$D_{im}, \qquad m = 1,2, \dots, M$$

в заданные моменты времени t_i. Многоугольники задаются последовательностью координат своих вершин:

$$(x_i, y_i),\ i = 1, 2, \dots, n + 1,$$

где $n > 2$ – число сторон в многоугольнике. Причём первая и последняя вершины совпадают, чтобы гарантировать замкнутость:

$$(x_0, y_0) = (x_{n+1}, y_{n+1}).$$

Статические ограничения представляют собой острова, береговые линии, зоны мелководья, зоны, ограниченные международными соглашениями, области пиратства.

Динамические ограничения определяют запретные для движения судна зоны, например, зоны с опасными погодными условиями. Данные ограничения изменяются через равные промежутки времени Δt по мере поступления новых данных о погоде.

В качестве критерия оптимальности было выбрано время. За начальный момент времени обозначим за t_0.

Задача – проложить маршрут от стартовой точки (x_s, y_s) до финальной точки (x_f, y_f), состоящий из последовательности точек

$$(x_j, y_j),\ j = 1, 2, \dots, l,$$

$$(x_1, y_1) = (x_s, y_s),$$
$$(x_l, y_l) = (x_f, y_f),$$

где (x_j, y_j) – координаты судна в момент времени

$$t_j = t_0 + j * \Delta t,$$

а l – длина маршрута. Причём ни в какой момент времени $t_i, i = 1,2, \dots, l$ маршрут не должен проходить через точки, принадлежащие статическим или динамическим запретным областям, то есть, никакая точка отрезка $[(x_i, y_i), (x_{i+1}, y_{i+1})], i = 1,2, \dots, l-1$ не должна принадлежать ни одной из областей, заданных многоугольниками:

$$S_n,\ D_{im},\ n = 1,2, \dots K,\ m = 1,2, \dots M.$$

При этом необходимо минимизировать время в пути, то есть

$$l \to min.$$

Предложенный в данной работе метод решения описанной задачи может быть разделен на следующие пункты:

1. Разработать и реализовать алгоритм генерации ограничений в виде многоугольников, заданных последовательностью координат своих вершин
2. Реализовать алгоритм гексагональной растеризации многоугольников
   2.1. Растеризация отрезков
   2.2. Заливка многоугольников
3. Разработать и реализовать алгоритм поиска оптимального маршрута на основе поиска в ширину и математической морфологии на гексагональной сетке
4. Реализовать вывод полученных результатов
5. Провести эксперименты на реальных данных, проанализировать алгоритм.

Обзор существующих методов

Согласно работе [14] первая служба маршрутизации судов прибрежного плавания была представлена в 1952 году. Её основной задачей было обеспечение безопасного плавания судна, чтобы команда и груз благополучно прибыли в место назначения без потерь и повреждений. В ней использовались первые базовые компьютеризированные модели прогноза погоды. Также в упомянутой работе можно найти наиболее известные современные системы планирования маршрутов.

Анализ публикаций по теме исследования, проведенный с помощью электронных ресурсов ScienceDirect [15], Connected Papers [16] и Elibrary [17], показал, что существует множество методов решений задачи weather routing, что лишний раз подчёркивает актуальность данной проблемы.

В работе [4] описываются, анализируются и сравниваются друг с другом основные методы решения задачи weather routing, такие как метод изохрон, метод динамического программирования, метод, основанный на вариационном исчислении, алгоритм Дейкстры, эволюционный алгоритм, метод DIRECT, разработанный Jones et al. в 1993 [13] году и смешанный подход, использующий несколько методов. Особенность метода DIRECT заключается в том, что он позволяет найти глобальный экстремум без вычисления градиента целевой функции. Это может быть очень полезно в случаях, когда вычисление градиента целевой функции ресурсозатратно или невозможно. Однако его недостатком является подверженность проблеме «проклятия размерности».

В целом, выбор конкретного решения сильно зависит от заданных требований и целей, от доступной информации, от выбора критерия оптимальности, от типа судна, от вычислительных возможностей и ограничений на время сходимости, от точности и др.

Каждый метод имеет свои преимущества и недостатки. Методы, основанные на динамическом программировании менее подвержены

опасности попасть в локальный экстремум, в отличие от методов, основанных на классическом исчислении вариаций [4]. Важным достоинством данных методов является возможность предоставить пользователю выбор оптимальных маршрутов (например, чтобы оптимизировать расход топлива). Основной недостаток – время вычислений зависит от степени детализации сетки.

Авторы статьи [4] замечают, что решения, использующие алгоритм Дейкстры, больше подходят для коротких прибрежных маршрутов в силу сравнительно больших временных затрат на вычисления. К их недостаткам можно отнести «негладкость» получаемого решения. Тем не менее, алгоритм Дейкстры использовался во многих исследованиях из-за его простоты и гибкости. Однако данный алгоритм имеет ещё один недостаток: при поиске выполняется больше действий, чем это необходимо. Для решения этой проблемы было предложено использовать алгоритм A*, основанный на использовании эвристической функции [2, 8].

В работе [2] рассмотрено применение алгоритма A* для построения оптимального по времени маршрута. Кроме того, используется подробная карта погодных условий, отражающая высоту и направление движения волн (рис. 2). Работа алгоритма тестируется на рейсах между Барселоной и Пальма-де-Майоркой. Даже при таком небольшом расстоянии авторам удалось сократить время плавания на 7% (по сравнению с минимальным по расстоянию маршрутом).

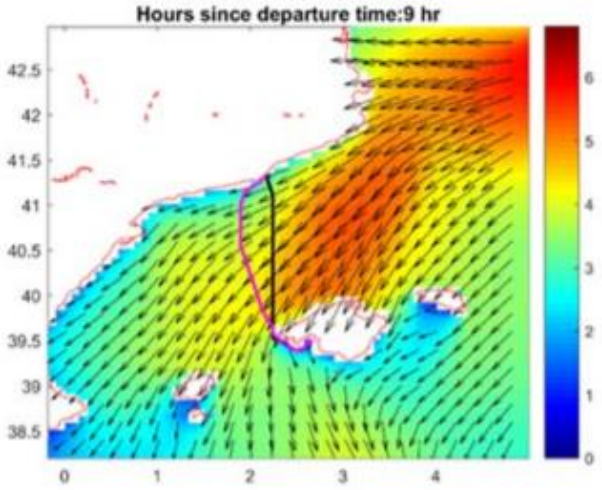

Рис. 2. Построение оптимального по времени маршрута (отмечен фиолетовым цветом) в условиях погодных ограничений. Чёрным показан кратчайший по расстоянию маршрут. Стрелки показывают направление движения волн, цвет – высоту волн [2]

Статья [6] освещает разработанный прототип оперативной системы поддержки принятия решений о маршрутах судов с использованием динамически меняющихся погодных областей. Кратчайший путь ищется с использованием модифицированного алгоритма Дейкстры на графе, где у каждой вершины – 24 соседа (рис. 3).

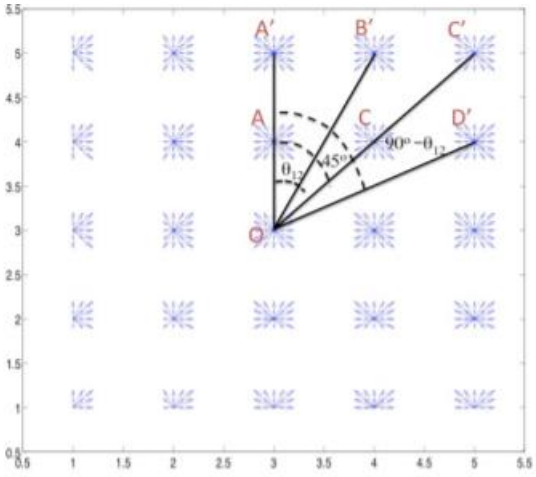

Рис. 3. Способ определение соседей узла в графе [6]

Модификация алгоритма заключается в использовании весов рёбер, как функций от времени [6]. Оптимизация маршрута происходит только по времени, однако авторы в будущем планируют реализовать оптимизацию по другим параметрам, в частности по расходу топлива.

Оба подхода – на основе алгоритма Дейкстры и на основе алгоритма A* – решают одноцелевую задачу, то есть предоставляют возможность оптимизировать только один критерий. Тем не менее, существует обобщенный алгоритм Дейкстры на случай двух целевых функций, рассматриваемый в статьях [7, 19].

Надо заметить, что в основе обоих алгоритмов (Дейкстры и A*) лежит поиск в ширину, который будет использоваться и в данной работе.

В статье [3] дан подробнейший обзор о достижениях в области weather routing, включая сравнение существующих систем и подходов.

Какие-то из этих методов были впервые описаны несколько десятилетий назад в довольно упрощенных формах, но всё возрастающая вычислительная мощность современных ЭВМ позволяет решать гораздо более сложные задачи. Кроме того, отмечается, что для большинства рассмотренных исследований целью было минимизировать либо расход топлива, либо время в пути (или и то, и другое в многокритериальных постановках задачи). Также, большинство исследований было выполнено с упором на океанские суда, и лишь несколько было проведено для прибрежных судов [3].

В работе приводится множество таблиц, сравнивающих различные подходы по таким параметрам, как критерий оптимизации, метод решения, учитываемые погодные данные, типа судна, зона плавания и др. (рис. 4).

Table 5g
Taxonomy part VII.

Taxonomy parameter/paper	Szlapczynska (2015)	Takashima et al. (2009)	Varelas et al. (2013)	Veneti et al. (2015)	Vettor and Guedes Soares (2016)
Optimization criterion	Multi-objective formulation on fuel consumption, time, and avoidance of unsecure areas	Fuel consumption	Time and fuel consumption	Total time, Fuel cost, and risk	Time and fuel consumption
Shipping sector	Ship type not specified	Ro-Ro and bulk carrier	Bulk carrier	Not specific	Liner shipping fishing vessel
Application area	North Atlantic	Japan	Trans-Atlantic	Aegean Sea	Transatlantic Mediterranean
Solution approach	Evolutionary algorithm	Dynamic programming, Dijkstra's	Dynamic programming	Genetic algorithm	Multi-objective evolutionary algorithm
Weather data	Waves, currents, tides, piracy and extreme wind areas	Wind and wave	Wave and wind	Wind speed and wave	Waves and weather conditions
Fuel efficiency considers	Calculated considering weather conditions	Increased resistance and speed loss	Increased resistance and speed loss (trim, draft environmental factors)	Increased resistance based on wind and wave	Weather conditions
Emissions considered	No	No	No	No	Yes
Fleet size	One ship	One ship	One ship	One ship	One ship
Type/field	Journal/Navigation	Journal/Navigation	Journal/Operations Research	Conference/ Transportation	Journal/Ocean Engineering

Рис. 4. Таблица сравнения различных подходов к решения задачи weather routing [3]

Как видно из таблицы, подходы могут довольно сильно отличаться. Например, Takashima рассматривает только ветер и волны [5], в то время как Szlapczynska учитывает ещё и течения, приливы, пиратство [9, 10]. Кроме того, метод, предложенный Takashima, основан на алгоритме Дейкстры и позволяет экономить топливо небольшим судам, совершающим рейсы у берегов Японии, а Szlapczynska рассматривает метод, основанный на эволюционном алгоритме, применимый к любым океанским судам и оптимизирующий сразу три критерия: расход топлива, время и нахождение в опасных зонах. В то же время, обе работы не затрагивают напрямую вопрос загрязнения окружающей среды.

В целом авторы сравнили 40 подходов, представленных научным сообществом в период с 1978 по 2019 года.

В статье [11] представлен краткий обзор существующих алгоритмов и дана подробная математическая составляющая алгоритма оптимизации маршрутов движения с учётом погодных условий, основанного на представлении маршрутов в трехмерном пространстве и сведения задачи к поиску траектории в трехмерном пространстве с обходом препятствий, представляющих статические и динамические ограничения.

Метод изохрон был впервые предложен в работе [20] и в дальнейшем развит в [21]. Недостатком этого подхода является то, что “он требует введения дополнительных упрощающих предположений, так как в противном случае вычислительные алгоритмы становятся слишком громоздкими для практического применения”, как замечает автор [11].

Идей классического вариационного исчисления описана в [22]. Однако применение этого подхода “для решения практических задач возможно только в случае предельно упрощенных постановок с примитивной математической моделью движения судна и без учета статических и динамических ограничений”, отмечает автор [23].

Применения методов теории динамического программирования для решения задачи weather routing рассматривается в [24, 25]. Однако автор [11] утверждает, что “ известная проблема «проклятия размерности» не позволяет использовать соответствующие алгоритмы на практике, так как вычислительные затраты при этом превосходят допустимые пределы, определяемые необходимостью постоянного пересчета маршрута при поступлении уточненных данных о прогнозе погодных условий”.

И наконец, в статьях [9, 10, 18] предлагается метод, заключающийся в рассмотрении задачи weather routing как задачи многокритериальной оптимизации, используя понятие оптимизации по Парето и идею генетических алгоритмов, с настраиваемым набором критериев и ограничений. Например, предложенный в [9, 10] алгоритм называется Multicriteria Evolutionary Weather Routing Algorithm (MEWRA). Особенностью данного подхода является возможность получить набор Парето-оптимальных решений, то есть таких, что улучшение одного из критериев невозможно без ухудшения какого-то другого критерия. Это позволяет выбирать подходящее решение в зависимости от ситуации.

Согласно [4] генетические подходы обеспечивают хорошую точность. Генетический алгоритм относительно прост в использовании, однако не

очень быстр [13]. Хиннеталь в [18] утверждает, что метод, основанный на генетическом алгоритме, никогда не достигнет вычислительной скорости детерминированного метода или метода, основанного на теории графов.

В целом наблюдается тенденция, что алгоритмы, склоняющиеся к использованию грубой силы, постепенно заменяются более “продвинутыми” алгоритмами, основанными, например, на методе динамического программирования или на эволюционном подходе. Они более эффективны в случаях многокритериальной оптимизации и ограниченности времени вычислений. Возможно, со временем будут описаны ещё более эффективные алгоритмы.

В данной же работе впервые рассматривается метод динамического программирования с использованием гексагональной растеризации, поиска в ширину и математической морфологии.

Глава 1. Реализация вспомогательных алгоритмов

1.1. Особенности реализации и хранения

В качестве языка программирования для реализации всех алгоритмов был выбран *Python*. Для демонстрации полученных результатов использовались библиотеки *Pillow* [26] и *Matplotlib* [27].

Код проекта находится в репозитории на github: https://github.com/4eckah78/weather_routing.

Данные о динамических и статических ограничениях-многоугольниках хранятся в виде .txt файлов, где каждая строчка представляет собой последовательность разделённых запятыми координат вершин многоугольника, причем первая и последняя вершины совпадают, чтобы гарантировать, что многоугольник замкнут. Координаты каждого многоугольника содержаться в отдельной строке.

1.2. Генерация случайных многоугольников

Изначально все ограничения планировалось генерировать случайно, впоследствии – использовать реальные данные о погодных условиях.

За основу алгоритма генерации случайных многоугольников взят алгоритм [28], идея которого заключается в соединении отрезками нескольких последовательных точек на окружности.

Алгоритм генерации N случайных многоугольников следующий:

- Внутри заданного прямоугольника (карты местности) случайно генерируются N точек центров окружностей. Для этого используется *метод Монте-Карло* моделирования случайных равномерно распределенных точек на плоскости. Пусть высота и ширина карты заданы и равны соответственно $height$ и $width$. Тогда значение равномерно распределенной в прямоугольнике $R = [0, width] \times$

$[0, height]$ случайной величины может быть разыграно по следующей формуле:

$$\begin{cases} x_0 = \gamma_1 * width, \\ y_0 = \gamma_2 * height, \end{cases}$$

где γ_1, γ_2 – реализации непрерывной равномерной случайной величины, распределённой на отрезке [0,1]. Далее для каждого сгенерированного центра выбирается случайный радиус r из заданного диапазона значений $[r_0, r_1]$;

- На каждой окружности генерируется последовательность случайных точек: (x_i, y_i), $i = 1, 2, \dots, n$, где $n > 2$ – число сторон в многоугольнике. Число n выбирается случайно из заданного диапазона значений $[n_0, n_1]$;
- Полученные вершины последовательно соединяются отрезками и последняя точка соединяется с первой.

Ниже представлен алгоритм генерации точек на окружности с центром (x_0, y_0) и радиусом r.

1. Вводятся угол φ и равномерная непрерывная случайная величина μ, распределенная на отрезке $[a, b]$, где

$$a = \frac{2\pi}{n_1}, \; b = \frac{2\pi}{n_0}.$$

Она означает возможный диапазон поворота полярного радиуса (см. рис. 1.1).

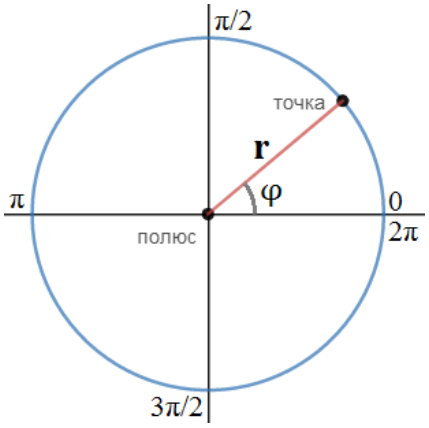

Рис. 1.1. Полярные координаты

По методу Монте-Карло моделируется угол

$$\varphi_0 = (b - a)\gamma + a,$$

где γ – реализация равномерно распределенной на отрезке [0, 1] непрерывной случайной величины. Далее инициализируется угол φ:

$$\varphi = \varphi_0.$$

2. Моделируется новый угол

$$\varphi_{new} = (b - a)\gamma + a, \quad (1)$$

где γ – реализация равномерно распределенной на отрезке [0, 1] непрерывной случайной величины.

После этого вычисляется новая точка (x_i, y_i) по формулам

$$x_i = r\cos\varphi + x_0,$$

$$y_i = r\sin\varphi + y_0$$

и увеличивается φ:

$$\varphi = \varphi + \varphi_{new}.$$

3. Шаг 2 повторяется, пока $\varphi \leq 2\pi$.

Примечания:

- Центры многоугольников моделируются равномерно по всей площади заданной области;
- При увеличении числа сторон n многоугольник стремится к окружности;
- Несмотря на то, что таким образом генерируются только выпуклые многоугольники, потенциально они могут перекрываться, образуя невыпуклые формы (это хорошо видно на рисунке 1.5).

Результаты работы алгоритма представлены на рисунках 1.2-1.5 ($width$ = 640 пикселей, $height$ = 480 пикселей).

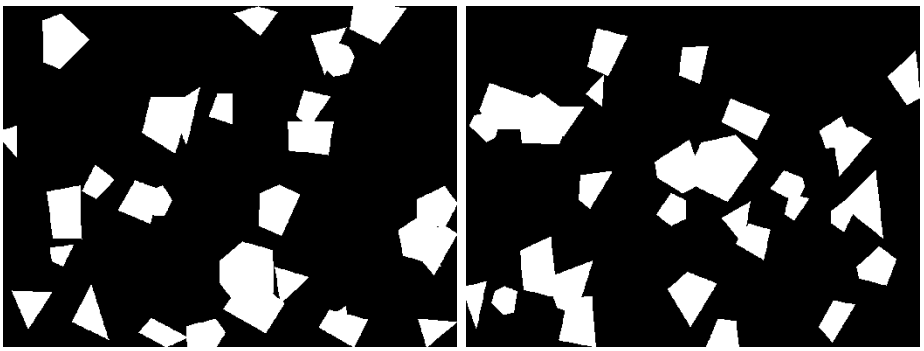

Рис. 1.2. 30 случайных многоугольников, $[r_0, r_1]$ = [20, 50], $[n_0, n_1]$ = [3, 12]

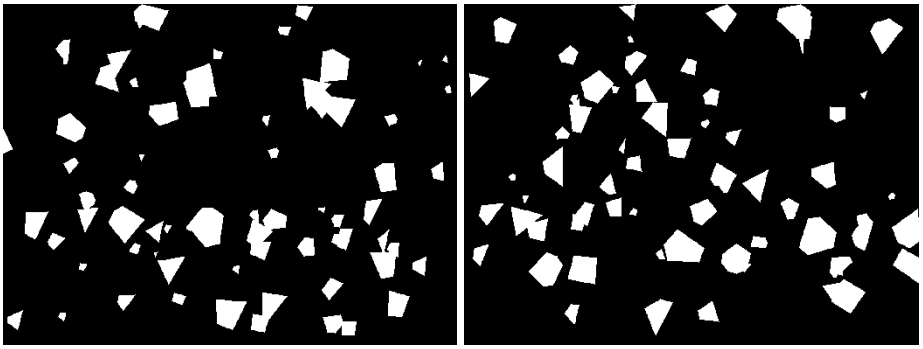

Рис. 1.3. 70 случайных многоугольников, $[r_0, r_1]$ = [5, 30], $[n_0, n_1]$= [3, 12]

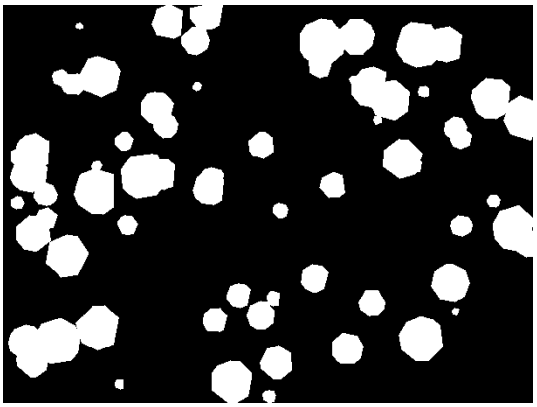

Рис. 1.4. 70 случайных многоугольников, $[r_0, r_1]$ = [5, 30], $[n_0, n_1]$ = [6, 17]

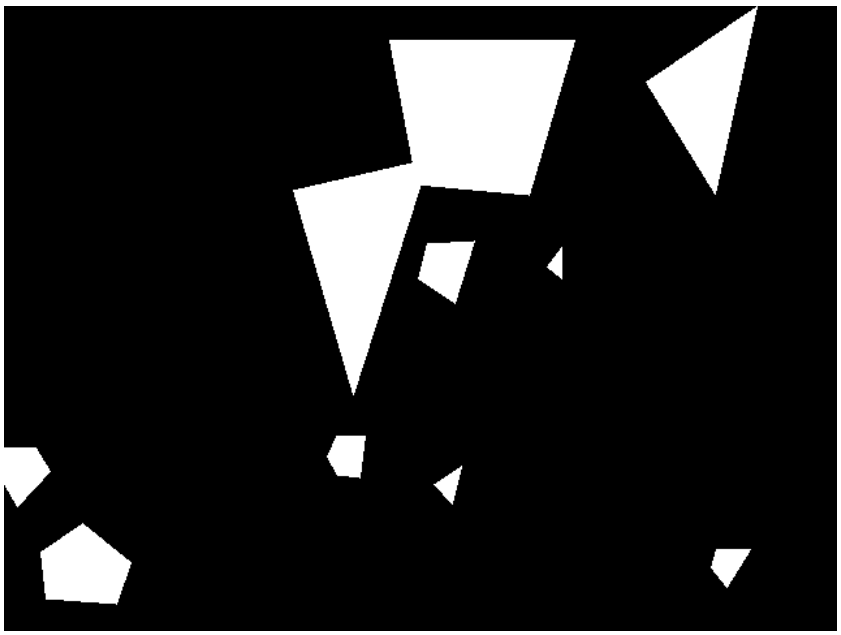

Рис. 1.5. 10 случайных многоугольников, $[r_0, r_1]$ = [10, 100], $[n_0, n_1]$ = [3, 10]

Полученные результаты можно назвать удовлетворительными. Многоугольники генерируются в целом равномерно по всей площади заданного прямоугольника в соответствии с заданными параметрами (рис. 2.2-2.3). Однако, как уже замечалось выше, уже при диапазоне сторон $[n_0, n_1]$ = [6, 15] многоугольники похожи на окружность (рис. 1.4).

Кроме того, на рис. 1.2 можно заметить интересный факт. Хотя диапазон возможного количества сторон $[n_0, n_1]$ равен [3, 12], на обеих картинках лишь малая часть многоугольников имеют пять-шесть сторон, остальные – меньше, а многоугольников с семью и более сторонами вообще не наблюдается.

Был проведён эксперимент, показавший, что на 100,000 тестов с параметрами N = 10, $[r_0, r_1]$ = [10, 100], $[n_0, n_1]$ = [3, 8], основная масса многоугольников имела 4 стороны, в то время как всего 4 многоугольника имели 7 сторон, 8 же сторон не имел ни один многоугольник (табл. 1).

Таблица 1. Результаты эксперимента

Количество сторон:	**3**	**4**	**5**	**6**	**7**
Количество многоугольников:	251899	604638	139255	4204	4

Причины такого распределения кроются в используемом способе генерации точек на окружности, а именно в выборе случайного угла φ_{new} с помощью формулы (1). На каждой итерации этот угол является реализацией непрерывной равномерно распределенной на отрезке [a, b] случайной величины, и именно факт равномерности делает невозможным появление многоугольника с восемью сторонами и уменьшает вероятность появления многоугольников с 6-7 сторонами. Полученное распределение скорее нормальное, чем равномерное. В алгоритме неявно используется сумма равномерных непрерывных случайных величин, а по центральной предельной теореме чем их больше, тем больше эта сумма стремится к нормальному распределению. Но для рассматриваемой задачи данная особенность не имеет значения.

Последним отметим, что многоугольники с рисунка 1.5 были взяты в качестве основных тестовых статических ограничений.

1.3. Гексагональный растр

Рассмотрим причины выбора именно гексагонального растра.

Существует только три правильных многоугольника, которыми можно замостить всю плоскость: треугольник, квадрат и гексагон (правильный шестиугольник) (рис. 1.6). Объяснение этого факта можно найти в [29].

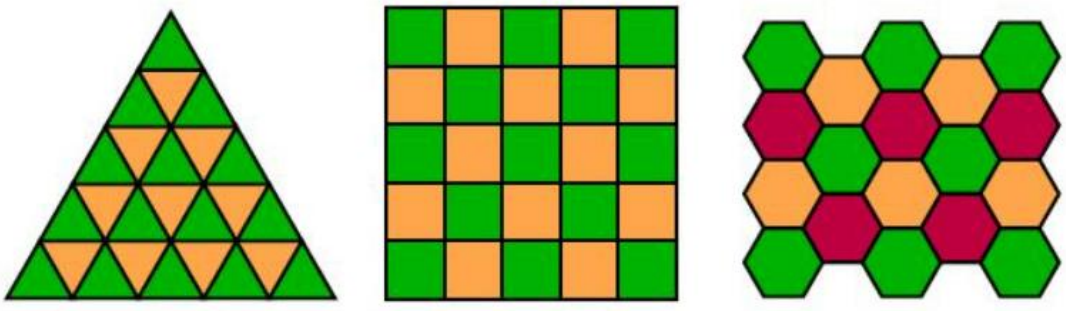

Рис. 1.6. Три способа замощения плоскости правильными многоугольниками

Некоторым недостатком квадратного растра можно считать разное расстояние от центра пикселя до центров соседних пикселей (расстояние до диагональных пикселей будет больше, чем до остальных, см. рис 1.7, картинка слева), в то время как в гексагональном растре расстояния до центров всех соседних пикселей одинаковое (см. рис. 1.7, картинка справа).

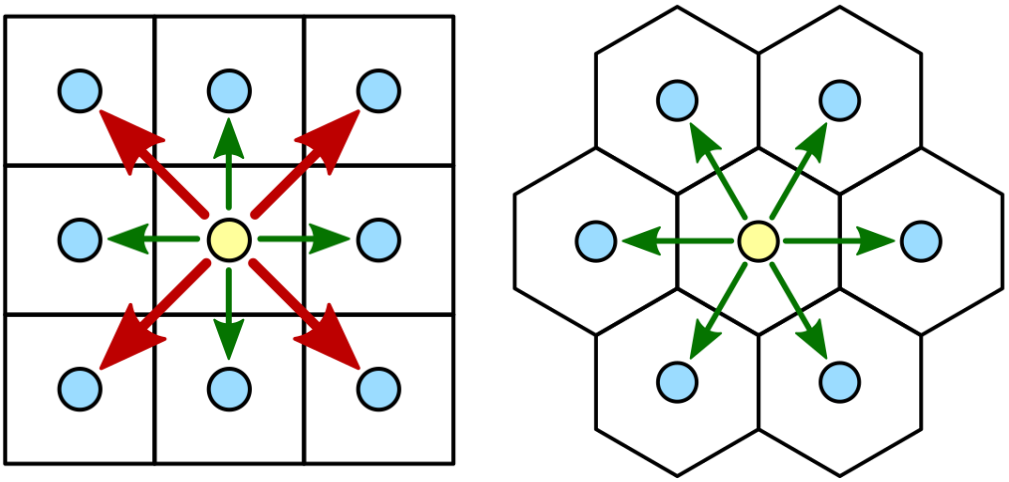

Рис. 1.7. Расстояния до центров соседних пикселей в квадратном и в гексагональном растрах [30]

Треугольная сетка обладает слишком сложной структурой по сравнению с двумя остальными (в ней присутствуют два типа треугольников: обращенные вершинами вверх и вниз, см. рис. 1.6, слева). Кроме того, возникают трудности с выделением соседей для пикселя.

По этим причинам было решено провести исследование работы алгоритмов именно с гексагональным растром.

В ходе исследований статьи [31] в качестве гексагональной системы координат были выбраны так называемые ***double height*** (удвоенные по высоте) гексагональные координаты, так как по сравнению с альтернативными вариантами: 1) они интуитивно понятны и 2) с ними легче проделывать разнообразные преобразования. В качестве положения гекса было выбрано положение типа ***flat topped***, то есть стороной вверх (см. рисунок 1.8).

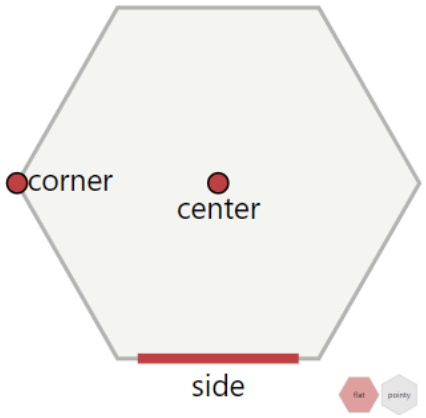

Рис. 1.8. Положение *flat topped* [31]

В выбранной системе координат, сетка будет выглядеть, как показано на рис. 1.9. Первая координата ячейки в гексагональном растре – номер столбца получившейся сетки $\boldsymbol{col}$, вторая – номер строки $\boldsymbol{row}$. Формулы преобразования декартовых координат в *double height* и обратно можно найти в [31].

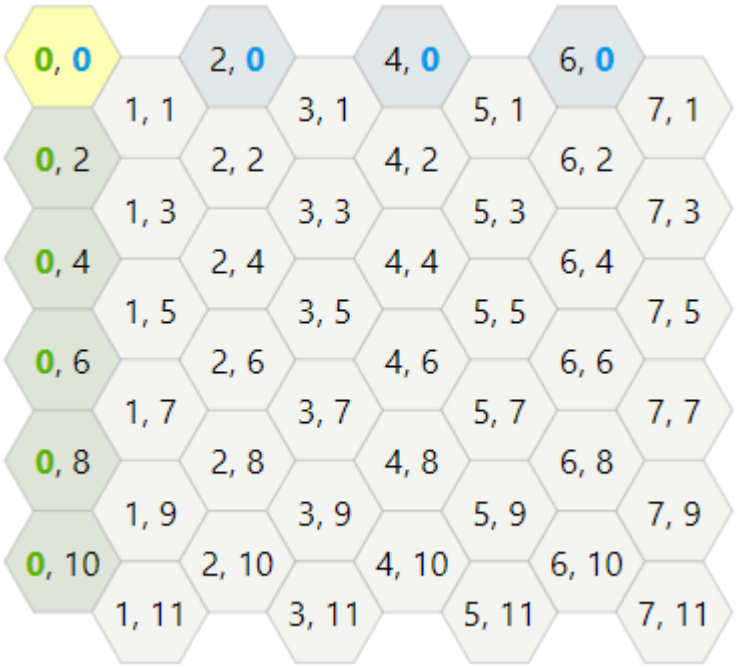

Рис. 1.9. Гексагональная сетка системы координат типа *double height* [31]

Данные о гексагональной карте хранятся в numpy массиве map с размерами $\left[\frac{width}{3a} \times \frac{heigth}{2\sqrt{3}a}\right]$. В каждой строке i массива хранятся гексы с координатой

$$row = i,$$

При необходимости массив можно сохранить в файл.

Также, реализована возможность настройки длины стороны гексагона a. В качестве замечания стоит отметить, что в теории a может быть любым вещественным неотрицательным числом, но при отображении будет происходить округление до целого количества пикселей.

1.4. Растеризация отрезков

Для растеризации сторон многоугольника использовался **алгоритм Брезенхема** растеризации отрезков, модифицированный для гексагонального растра. Его описание можно найти в [32]. Одним из главных преимуществ этого алгоритма является то, что в нем не используются вычисления с плавающей точкой, что делает его крайне эффективным.

Результаты растеризации сторон многоугольников с рис. 1.5. при выборе разной длины стороны гекса a представлены на рис. 1.10-1.11. Для проверки корректности растеризации наложены стороны многоугольников в квадратном растре.

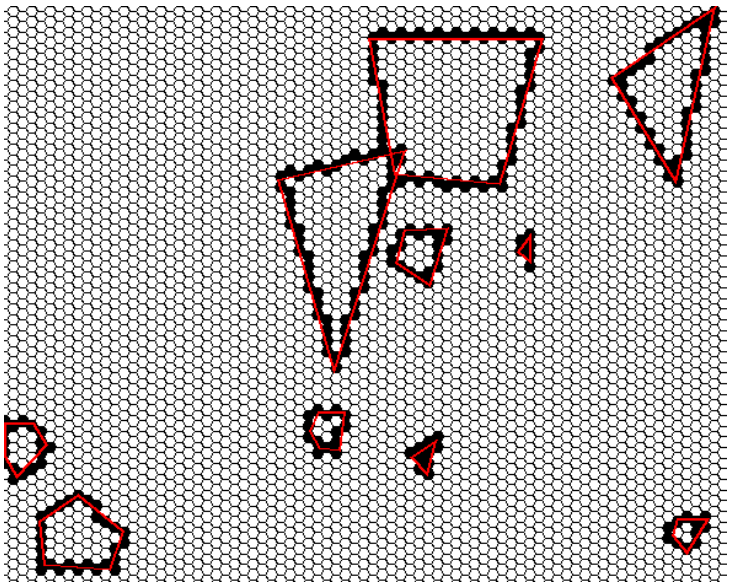

Рис. 1.10. Многоугольники в гексагональном растре с наложенными сторонами в квадратном растре, сторона одного гекса a равна 5 пикселям

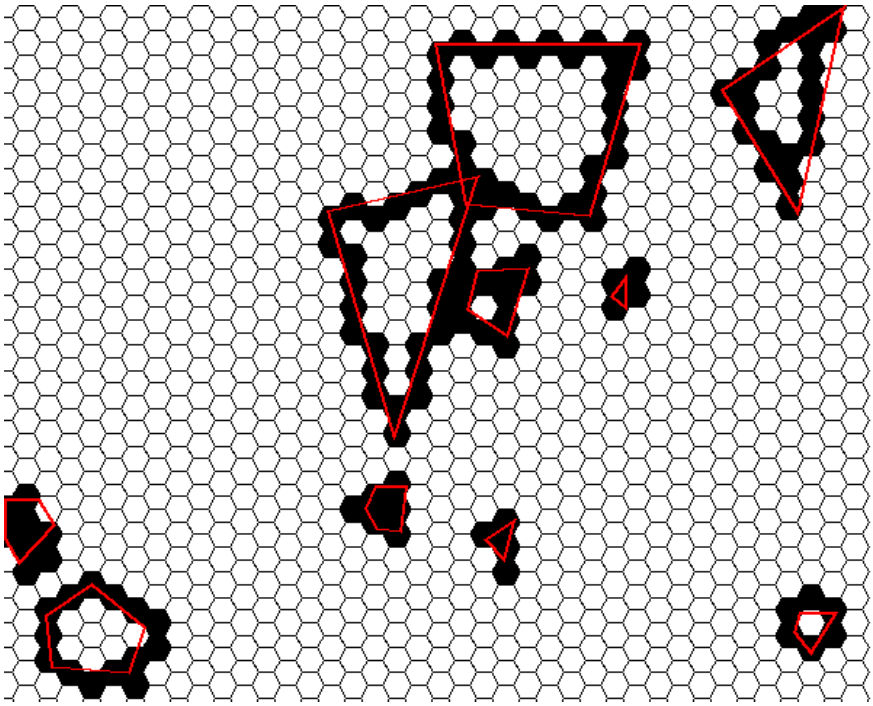

Рис. 1.11. Многоугольники в гексагональном растре с наложенными сторонами в квадратном растре, сторона одного гекса a равна 10 пикселям

На рис. 1.12 представлена растеризация сторон более сложного многоугольника, отражающего реальную зону опасно высокой скорости ветра.

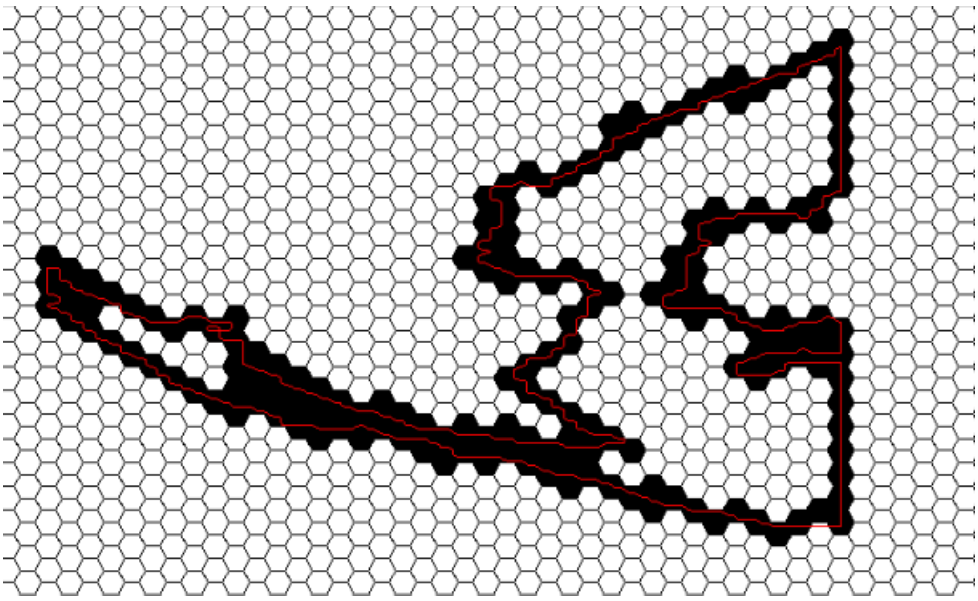

Рис. 1.12. Растеризация реальных погодных данных. Красным выделены стороны многоугольников в квадратном растре

Заметим, что отрезки в гексагональном и в квадратном растрах практически совпадают на рис. 1.10, в то время как на рис. 1.11 наблюдается погрешность, связанная с увеличенной стороной гекса a.

1.5. Заливка многоугольников

Заливка многоугольников происходит по модифицированному под гексагональный растр алгоритму со списком рёберных точек [33].

Модификация заключается в следующем. В отличие от квадратного растра, многоугольник заливается по столбцам гексагональной сетки сверху вниз. То есть по таким гексагональным линиям, у которых значения координаты столбца col одинаковы (см. рис. 1.9). Соответственно, пересечения сторон многоугольника ищутся с линиями, параллельными не оси Ox, а оси Oy с шагом $1.5a$, а не 1, как в классическом алгоритме. Причём линии проходят через центры гексов соответствующего столбца. В остальном же всё аналогично алгоритму, описанному в [33].

Результаты заливки тестовых многоугольников представлены на рис. 1.13.

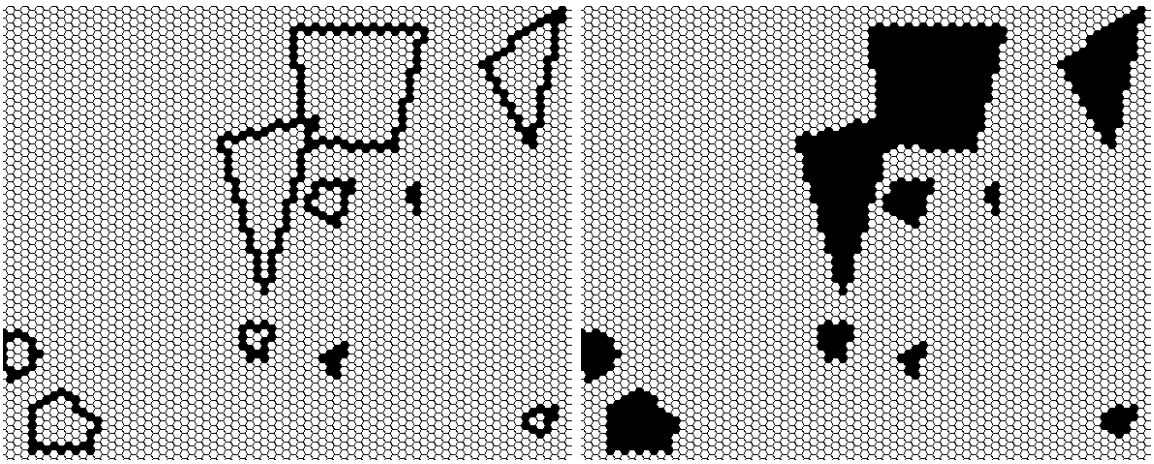

Рис. 1.13. Результаты заливки многоугольников

Как уже замечалось ранее, все сгенерированные случайным образом многоугольники являются выпуклыми. В связи с чем, для демонстрации корректности заливки любых многоугольников на рис. 1.14 проиллюстрированы некоторые этапы успешного заполнения невыпуклого многоугольника.

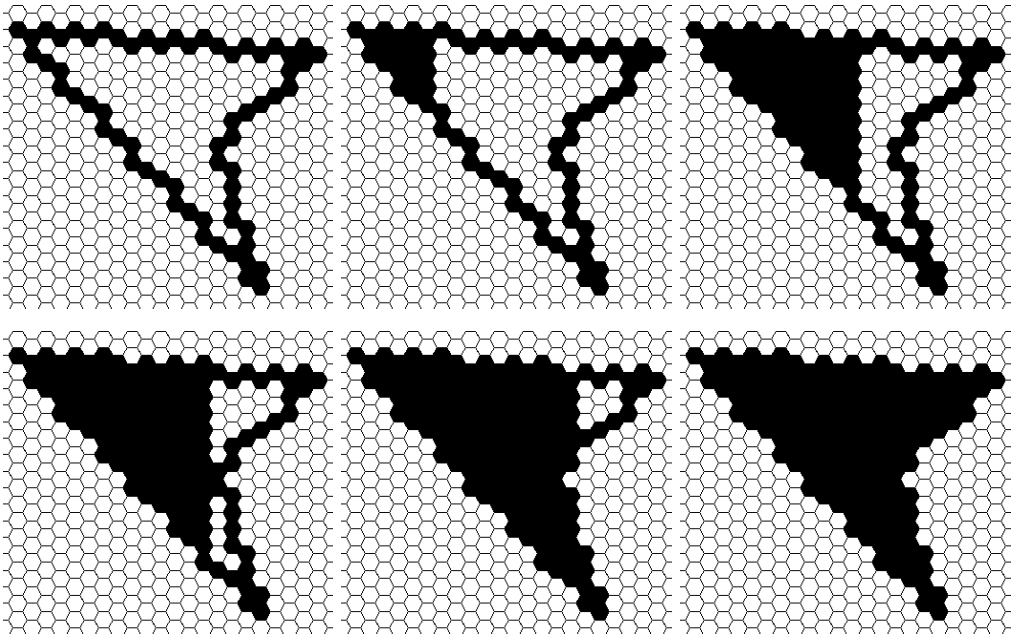

Рис. 1.14. Этапы заливки невыпуклого многоугольника

Глава 2. Алгоритм поиска оптимального маршрута

После гексагональной растеризации заданной карты необходимо приступить к решению основной задачи: найти и отобразить оптимальный маршрут (или несколько, если таковой не один) от старта до точки назначения, учитывая статические и динамические ограничения.

Точка старта (x_s, y_s) и точка финиша (x_f, y_f) преобразуются в гексагональные координаты *double height* – (row_s, col_s) и (row_f, col_f). Здесь и далее, если не указано иное, будем предполагать, что

- Точка старта находится в левом верхнем углу;
- Точка финиша находится в правом нижнем углу;
- Промежуток времени, через которые меняются динамические ограничения – один час.

Замечание. В реальных рейсах судна имеют конечный набор фиксированных скоростей: полный ход, малый ход и т.д. Если данные о погоде обновляются раз в час, то сторону гекса a имеет смысл брать так, чтобы расстояние между центрами соседних гексов было равно расстоянию $minDist$, которое судно пройдет за один час с минимальной скоростью. То есть, так как расстояние между центрами соседних гексов равняется $\sqrt{3}a$ (это легко показать, исходя из свойств правильного шестиугольника), то искомая длина стороны гекса a находится по формуле

$$a = \frac{minDist}{\sqrt{3}}.$$

1.1. Идея и описание алгоритма

Исследуемый в данной работе алгоритм, реализующий поиск в ширину и идею математической морфологии, интуитивно понятен и прост.

Его можно сравнить с эволюцией развития плесени в чашке Петри: из одной точки плесень растет во все стороны. С каждым моментом времени

она занимает всё больше пространства. Если на её пути будет лежать какое-либо препятствие, она его обойдет, а если на плесень капнуть гипохлорит натрия, она в этой области погибнет. Но как только капля исчезнет, плесень тут же займет освободившийся участок. Плесень символизирует точки, в которых может находиться корабль, препятствие означает некое статическое ограничение, гипохлорит натрия – динамическое.

Идея алгоритма находится в гармонии с реалиями судоходства, где существует возможность некоторое время стоять на месте, пережидая некие динамические ограничения, препятствующие дальнейшему продвижению, за счёт чего точка назначения может быть достигнута в более ранние сроки, чем если бы судно пыталось обойти препятствие.

Предлагаемый алгоритм состоит из **двух проходов**.

Каждому гексагону hex_i гексагональной карты сопоставляется свой набор меток времени $marks_i$, который содержит те моменты времени, в которые судно могло находиться в этом гексагоне.

1. На нулевой итерации в начальный момент времени t_0 для всех гексагонов этот набор пустой, кроме стартовой точки, у которой он состоит из одной метки

   $$t = t_0.$$

   Вводится множество $visited_0$ посещенных в начальный момент времени t_0 гексов, содержащее стартовую точку (x_s, y_s).
2. На каждой следующей итерации i в момент времени t_i происходит два действия:

   1) гексагональная карта map обновляется в соответствии с изменениями динамических ограничений в момент времени t_i и

   2) множество допустимых гексагонов из множества $visited_{i-1}$ объединяется с множеством допустимых соседей этого множества, образуя новое множество $visited_i$. **Допустимыми гексагонами** считаются гексагоны, не являющиеся в данный момент времени t_i

ограничениями. В наборы всех гексов получившегося множества $visited_i$ добавляется метка t_i.

3. Шаг 2 повторяется, пока в момент времени t_{end} не будет достигнута конечная точка (x_f, y_f). На этом заканчивается первый, **прямой проход** алгоритма.
4. Далее следует второй, **обратный проход**, позволяющий найти только кратчайшие маршруты, и не рассматривать все остальные, тупиковые, не ведущие к конечной точке. Проход начинается с конечной точки, у которой в наборе единственная метка

$$t = t_{end}.$$

Далее рассматриваются все её соседи, у которых в наборе присутствует метка

$$t = t_{end} - 1,$$

а потом все их соседи, у которых присутствует метка

$$t = t_{end} - 2,$$

и так далее, пока не будет достигнута начальная точка (x_s, y_s) с меткой

$$t = t_0.$$

Все пройденные таким образом гексы будут часть кратчайших маршрутов из начальной точки в финальную. Итоговый маршрут строится путём последовательного соединения центров гексов.

На случай, если пути из стартовой точки в финальную не существует, вводится ограничение на максимальное число итераций.

Кроме того, в проекте реализована возможность настройки количества гексагонов $distance$, которые можно пройти за единицу времени (например, за час).

1.2. Данные для тестирования

Как уже говорилось, в качестве тестовых статических ограничений используются ограничения с рис 1.5.

Реальные погодные данные можно посмотреть в репозитории на github: https://github.com/4eckah78/weather_routing в папке my_weather1. Прогнозы отражают зоны с большой скоростью ветра и заданы в виде многоугольников способом, аналогичным описанному в разделе «Постановка задачи». Прогноз охватывает область Атлантического океана примерно от 25° до 60° с.ш. и от 8° до 81° з.д.

Пример данных о погоде в квадратном растре можно увидеть на рис. 2.1.

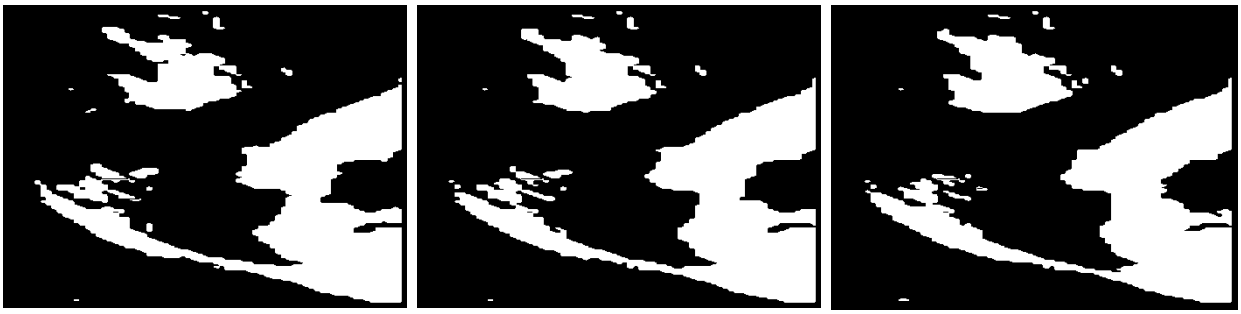

Рис. 2.1. Пример реальных погодных данных. Белым выделены зоны с опасной скоростью ветра. Данные обновляются каждый час

1.3. Тестирование алгоритма

Сначала протестируем алгоритм, учитывая только статические ограничения. Результаты представлены на рис. 2.2.

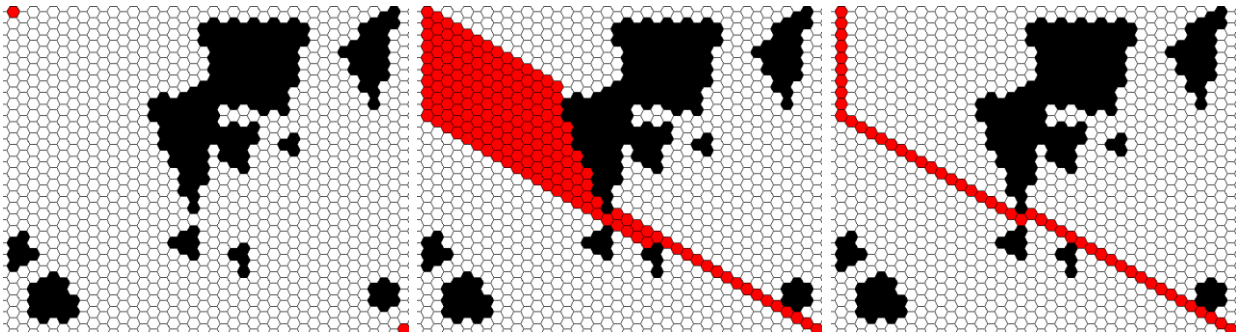

Рис. 2.2. Построение оптимальных маршрутов в условиях статических ограничений. Черным выделены ограничения. Красным на картинке слева выделены стартовая и финальная точки. На картинке посередине – все кратчайшие маршруты, на картинке справа – один из этих маршрутов

Визуально алгоритм работает корректно. Маршруты не проходят по областям ограничений. Однако так как под расстоянием понимается **Манхэттенское расстояние**, кратчайшие пути могут образовывать «параллелограммы» (рис. 2.2, картинка посередине). Это влечёт за собой некую «неоптимальность» построенных маршрутов в том смысле, что можно построить более короткие. Подробный анализ данной проблемы и методы её решения будут описаны в следующем параграфе.

Теперь добавим два динамических ограничения: одно будет двигаться по диагонали в левом верхнем углу, второе будет расширяться в правом нижнем углу.

Пусть $distance$ равняется 10. Этапы построения оптимальных маршрутов представлены на рис. 2.3.

Здесь и далее на первой картинке показан начальный прогноз, на следующей строится соответствующая этому прогнозу часть оптимальных маршрутов, на следующей - новый прогноз и так далее. Бордовые гексы означают те участки маршрутов, на которых при обновлении прогноза появились ограничения.

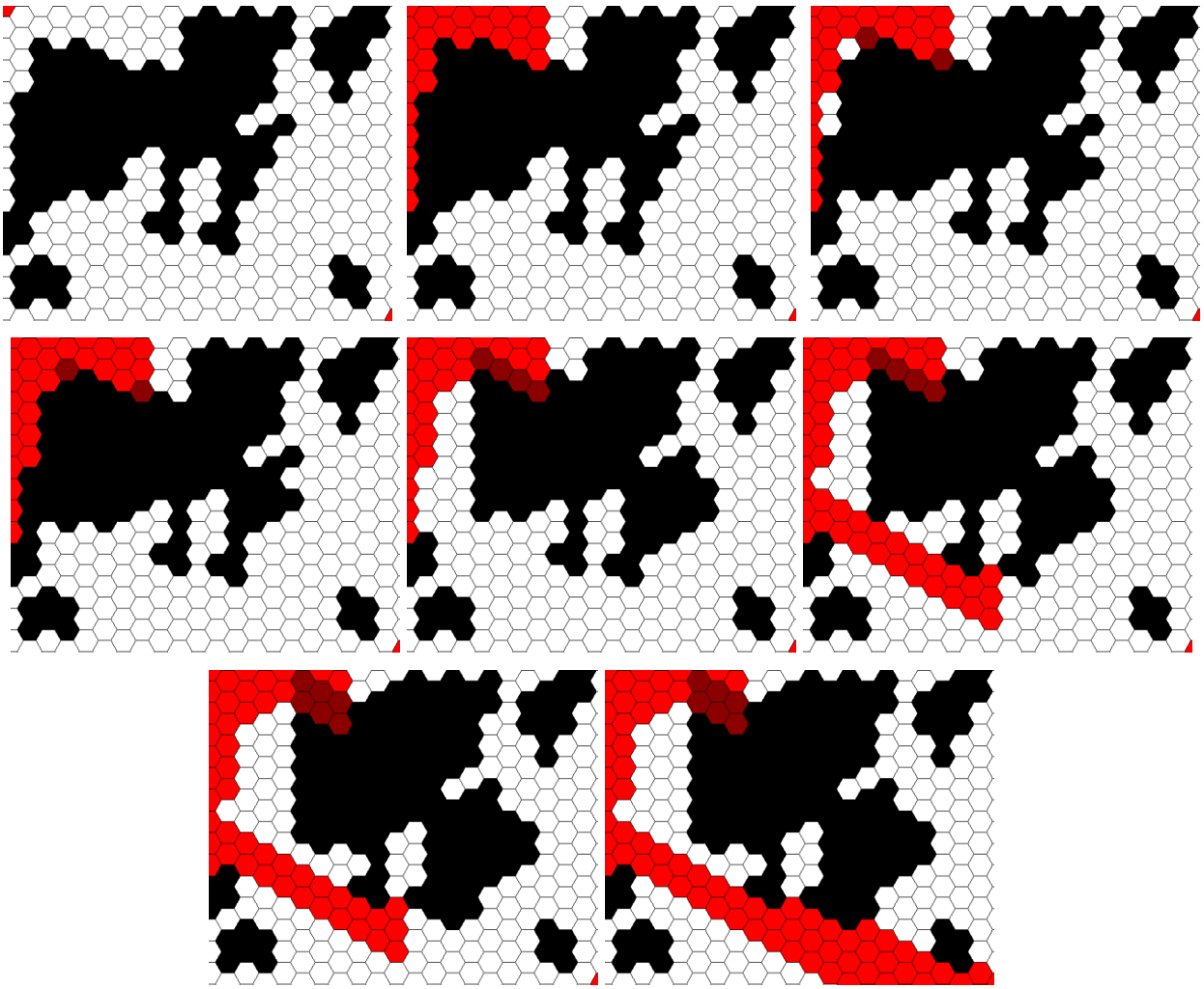

Рис. 2.3. Этапы построение оптимальных маршрутов в условиях динамических ограничений

Данный пример иллюстрирует возможность судна стоять на месте и пережидать, пока динамические ограничения не пройдут.

Маршруты не проходят по областям ограничений, алгоритм работает корректно.

Протестируем работу алгоритма на реальных погодных данных (рис. 2.4).

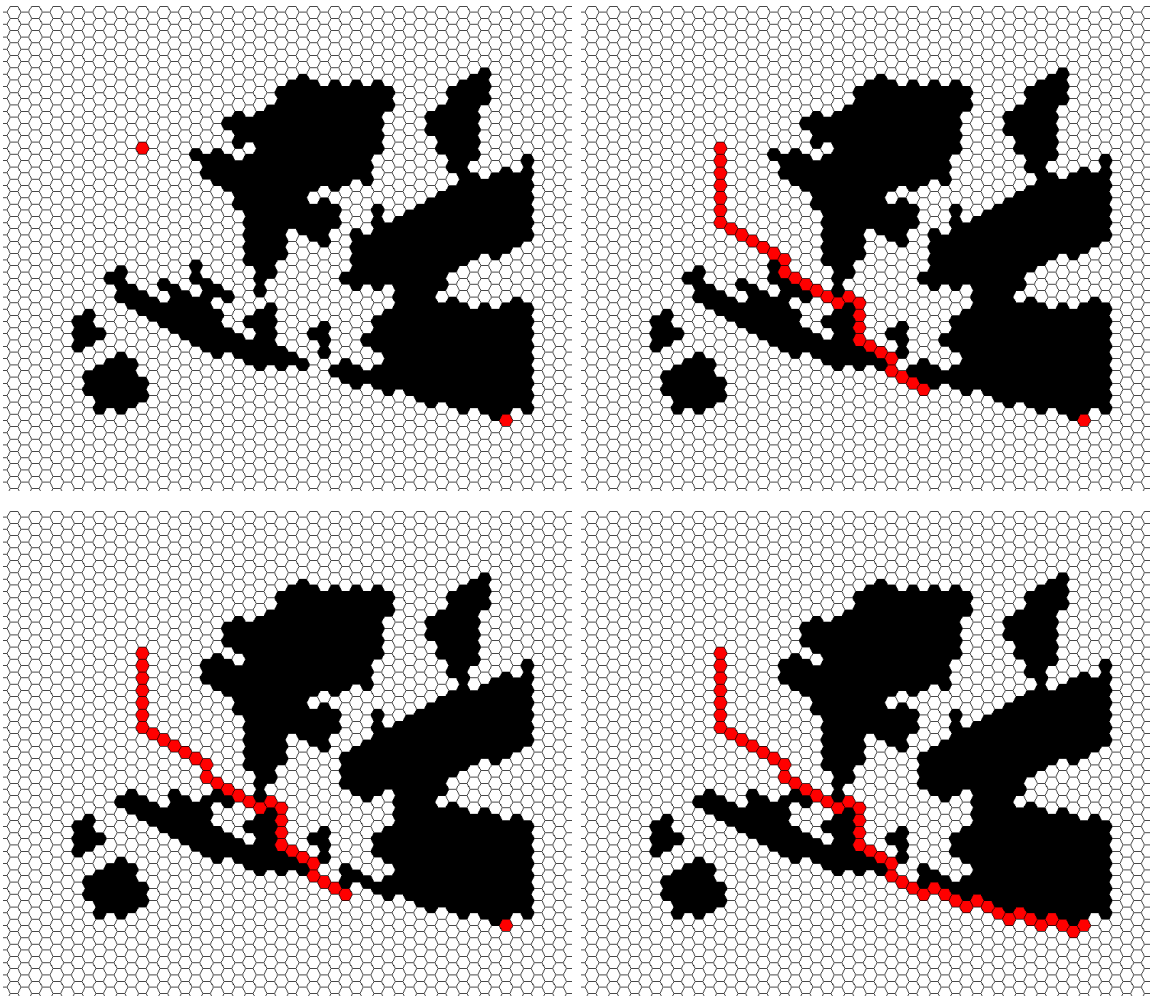

Рис. 2.4. Этапы построение оптимального маршрута в условиях динамически меняющихся ограничений. Реальные погодные данные, $distance = 30$

В этот раз отобразим этапы построения одного из оптимальных путей. Полученные результаты отражают корректную работу алгоритма.

Сравним различные разрешения гексагональной сетки. Результаты получились ожидаемо похожими (рис. 2.5).

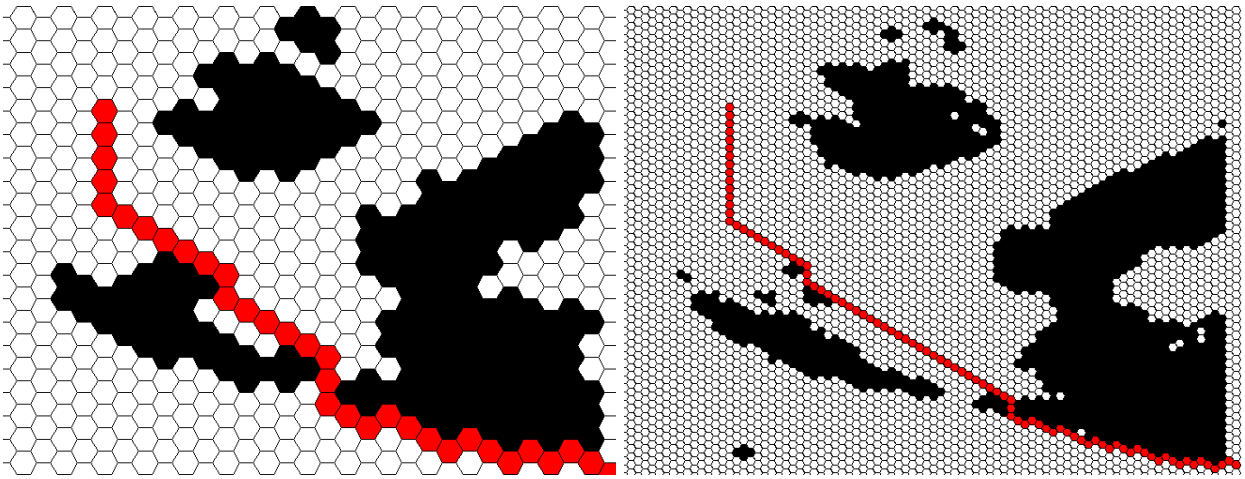

Рис. 2.5. Оптимальный маршрут в условиях динамических ограничений. Реальные погодные данные, $a = 15$ (слева) и $a = 5$ (справа)

Глава 3. Анализ алгоритма

Сначала проведём анализ упомянутой выше проблемы возникновения «параллелограммов» (рис. 2.2).

3.1. Повороты карты

Проведём два эксперимента, позволяющие выяснить причины такого поведения.

1. **Первый эксперимент** заключается в построении оптимальных маршрутов без ограничений при разных значениях a от центра левой стороны карты до 1) правого нижнего угла 2) центра правой стороны и 3) правого верхнего угла (рис 3.1-3.2).

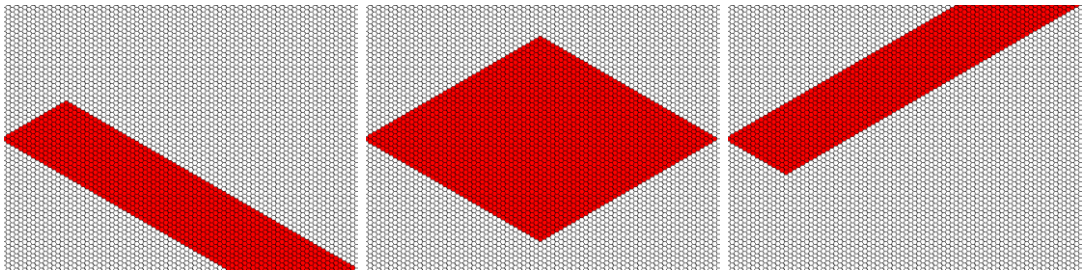

Рис. 3.1. Эксперимент 1. Различные точки финиша. a = 5

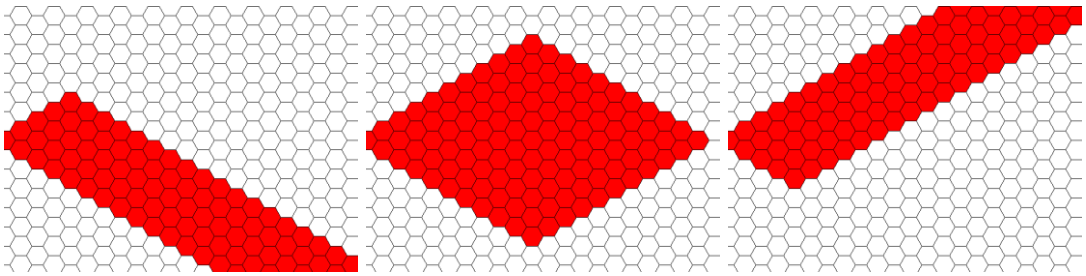

Рис. 3.2. Эксперимент 1. Различные точки финиша. a = 15

Первый вывод, который можно сделать – данная проблема не зависит от размера стороны гекса a. Второй вывод – «наихудший» вариант получается на средней картинке, при нахождении старта и финиша на одной горизонтальной прямой.

2. **Второй эксперимент** заключается в повороте гексагональной карты вокруг центра на угол $alpha$ в целях выяснить наилучший угол между стартом и финишем (рис. 3.3).

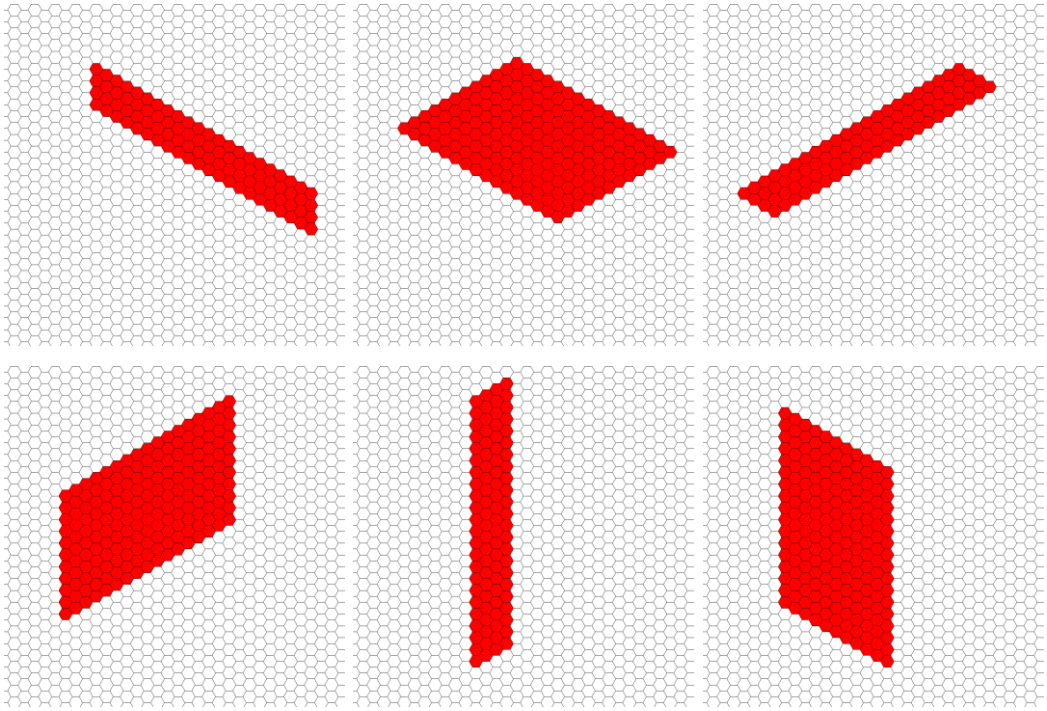

Рис. 3.3. Эксперимент 2. Повороты карты. На первой картинке начальное положение, на следующих – повернутое на 30° против часовой стрелки. a равняется 20

Наилучшее положение получается в двух случаях: начальная и конечная точка расположены 1) по диагонали 2) на одной вертикали. Поворот карты происходит с использованием формул ортогонального преобразования и параллельного переноса центра координат. В целях проверки корректности выполняемых преобразований, на рис. 3.4 показаны повороты карты с статическими ограничениями.

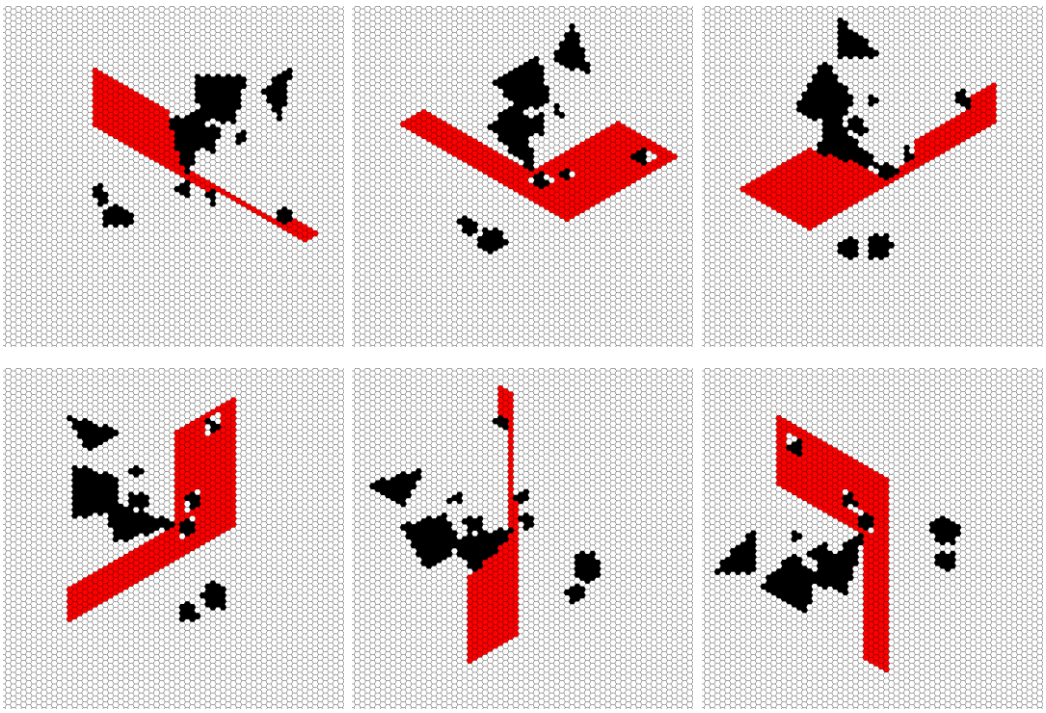

Рис. 3.4. Эксперимент 2. Повороты карты с статическими ограничениями. На первой картинке начальное положение, на следующих – повернутое на 30° против часовой стрелки. $a = 20$

А на рис. 3.5 представлены результаты поворота многоугольников, взятых из реальных погодных данных в квадратном растре.

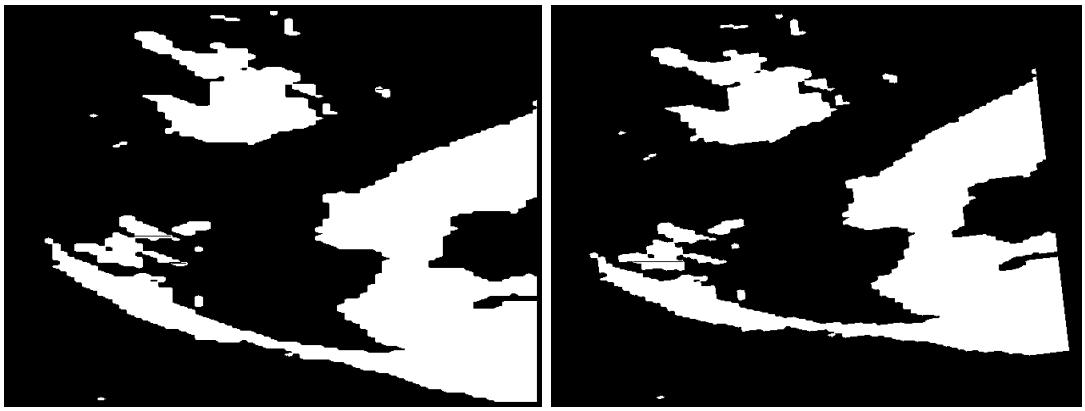

Рис. 3.5. Повороты карты с реальными погодными данными. Угол поворота равен 15°

3.2. Поиск оптимального угла

Уберём все статические и динамические ограничения и расположим стартовую и конечную точки в левый верхний и правый нижний углы карты соответственно.

Для начала проверим, есть ли зависимость длины кратчайшего пути в пикселях от длины стороны гекса a (тоже в пикселях). Значение a изменяется в пределах от 2 до 50 (рис. 3.6). Расстояние между двумя соседними гексами считается как расстояние между их центрами. Как уже упоминалось, расстояние между центрами двух соседних гексов равняется $\sqrt{3}a$.

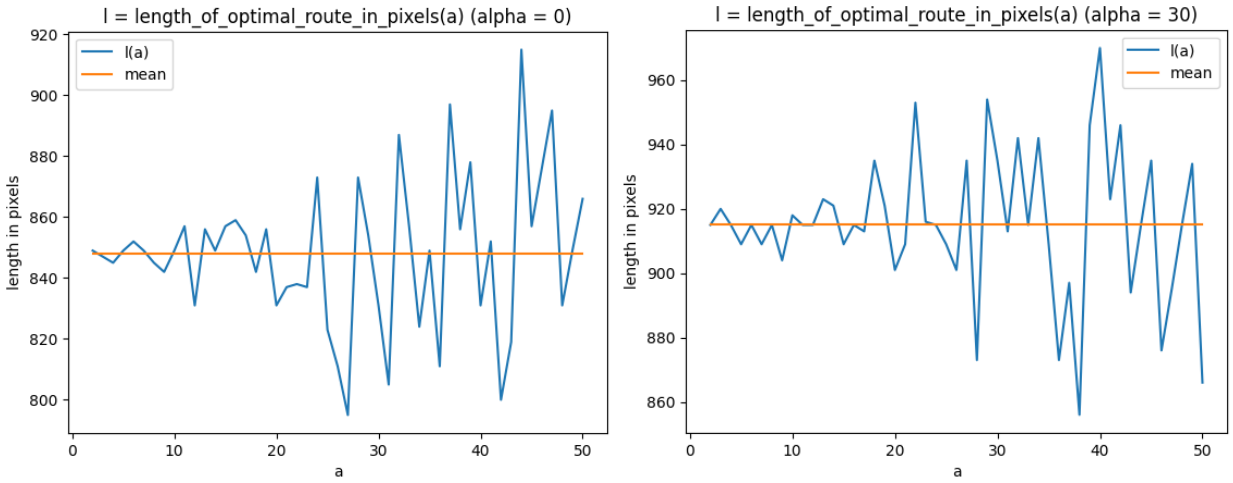

Рис. 3.6. Зависимость длины оптимального маршрута в пикселях от длины стороны гекса a. На первой картинке поворота не происходит, на второй происходит поворот карты на 30° против часовой стрелки

С ростом a увеличивается «разброс», который составляет до 120 пикселей. Это примерно равно суммарной длине двух соседних гексов с стороной a равной 40. Конкретную зависимость от a отследить трудно, она похожа на случайную.

Рассмотрим состояние карты при наилучшем и наихудшем значениях a (рис. 3.7).

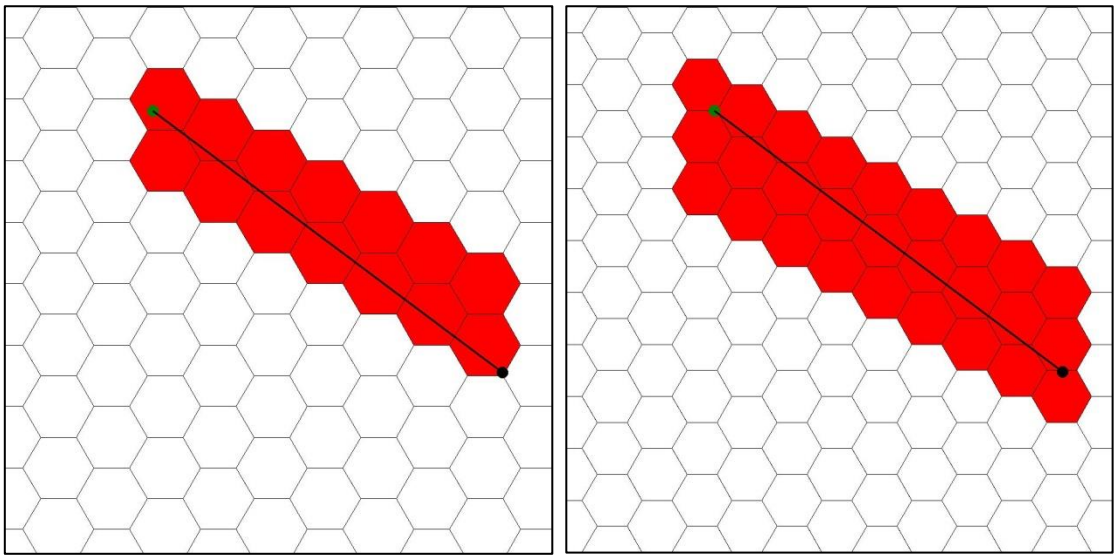

Рис. 3.7. Наилучшее a (слева) и наихудшее a (справа). Зеленая точка – стартовая (в пикселях), чёрная – конечная (в пикселях)

Начальная точка находится не в центре, вследствие чего возникает погрешность (равная $2a$). Поэтому, для чистоты эксперимента, было решено помещать начальную точку в центр гексагона. Данное преобразование происходит с помощью параллельного переноса всей карты. Результаты представлены на рис. 3.8.

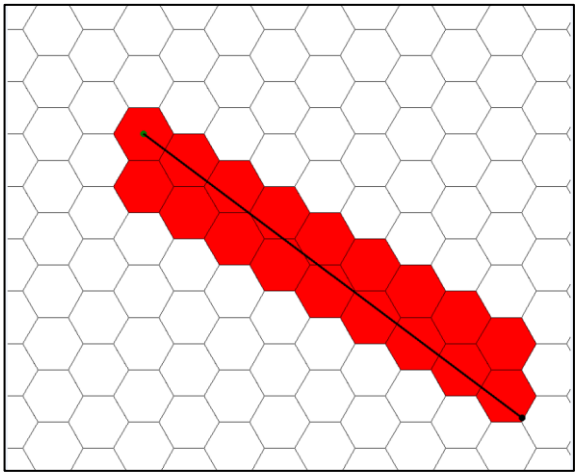

Рис. 3.8. Результат помещение стартовой точки (выделена зелёным цветом) в центр гексагона

Погрешность всё равно остаётся, но она уже меньше и составляет a пикселей (расстояние от центра гекса до его углов).

На рис. 3.6 наблюдается некая зависимость длины маршрута в пикселях от угла поворота. В связи с этим, рассмотрим более подробно зависимость длины кратчайшего маршрута в пикселях от угла поворота $alpha$ (от $0°$ до $150°$) при фиксированной длине стороны гекса a (рис. 3.9).

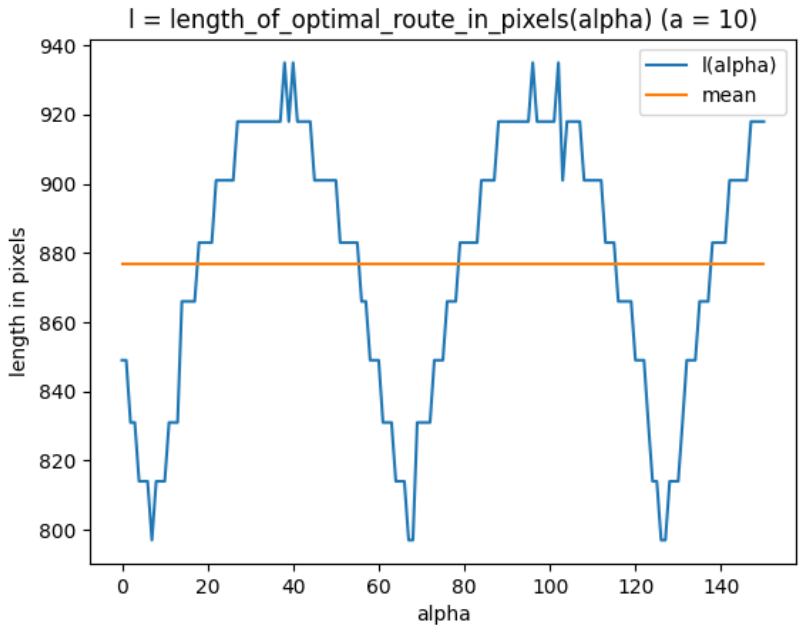

Рис. 3.9. Зависимость длины оптимального маршрута в пикселях от угла поворота $alpha$ при $a = 10$

Наблюдается периодическая зависимость с периодом примерно $60°$. Данный факт гармонирует со свойствами правильного шестиугольника.

Ровные, параллельные оси Ox участки возникают из-за того, что небольшие повороты карты в 1-2 градуса не изменяют длину кратчайшего маршрута в силу относительно большой величины гекса.

Минимальные значения расстояния были получены при следующих положениях карты (рис. 3.10):

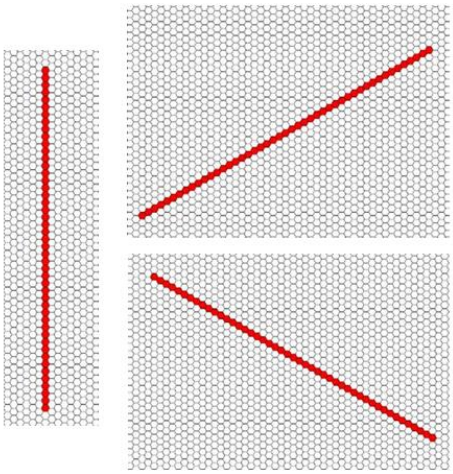

Рис. 3.10. Оптимальные положения карты

То есть, чтобы уменьшить влияние проблемы возникновения «параллелограммов», необходимо поворачивать карту так, чтобы стартовая и конечная точка были в одном из положений, показанных на рис. 3.10. В качестве такового было выбрано положение, когда стартовая точка находится в левом нижнем углу, а конечная – в правом нижнем (правая нижняя картинка на рис. 3.10). При таком положении, угол между прямой, соединяющей стартовую и финальную точки, и осью Ox составляет -30°. Это следует из свойств правильного шестиугольника (см. рис. 3.11).

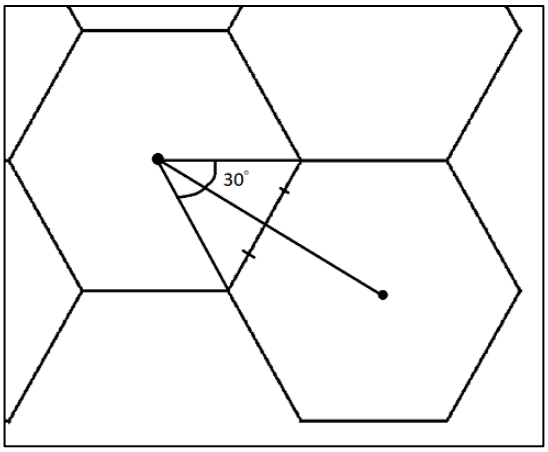

Рис. 3.11. Свойство правильного шестиугольника

Таким образом, перед началом вычислений необходимо поворачивать карту на угол $alpha_{opt}$ по формуле:

$$alpha_{opt} = -\frac{\pi}{6} + betta,$$

где

$$betta = \tan^{-1}\frac{y_f - y_s}{x_f - x_s}.$$

Теперь рассмотрим те же зависимости, но с добавлением динамических ограничений (рис. 3.12)

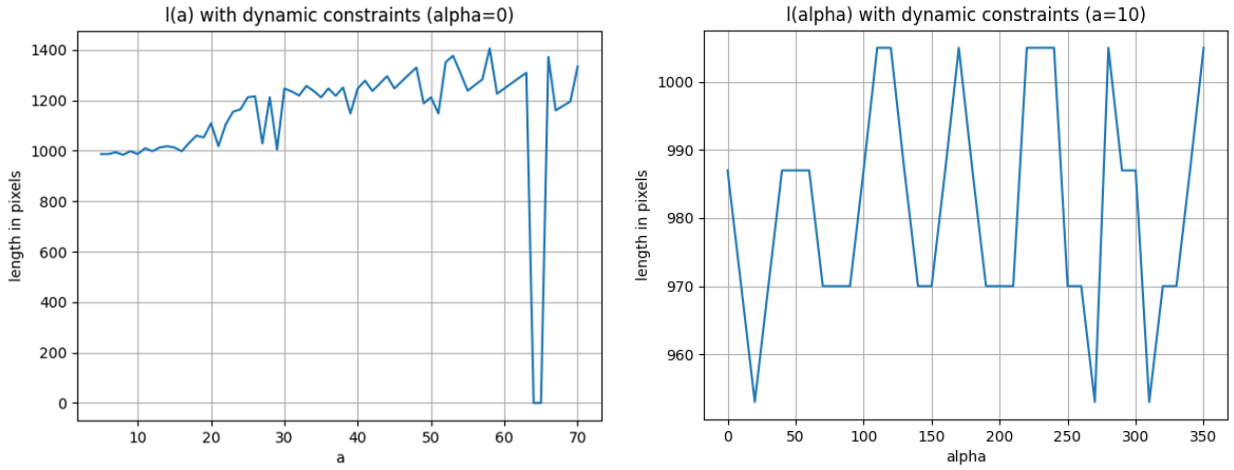

Рис. 3.12. Зависимость длины оптимального маршрута в пикселях от a (слева) и от угла поворота $alpha$ с шагом 15° (справа)

Резкий спад до нуля у картинки слева означает, что при таких значениях a не было найдено маршрута до конечной точки (то есть либо конечная, либо стартовая точки попали в область ограничений).

В целом, длина оптимального маршрута от a увеличилась на 40%, а от $alpha$ – на 10% по сравнению с экспериментами при отсутствующих ограничениях.

Построим 3D графики, иллюстрирующие зависимость длины оптимального маршрута в пикселях от a и от угла поворота $alpha$ без ограничений (рис. 3.13) и с ограничениями (рис. 3.14). Все упомянутые выше тенденции сохранились.

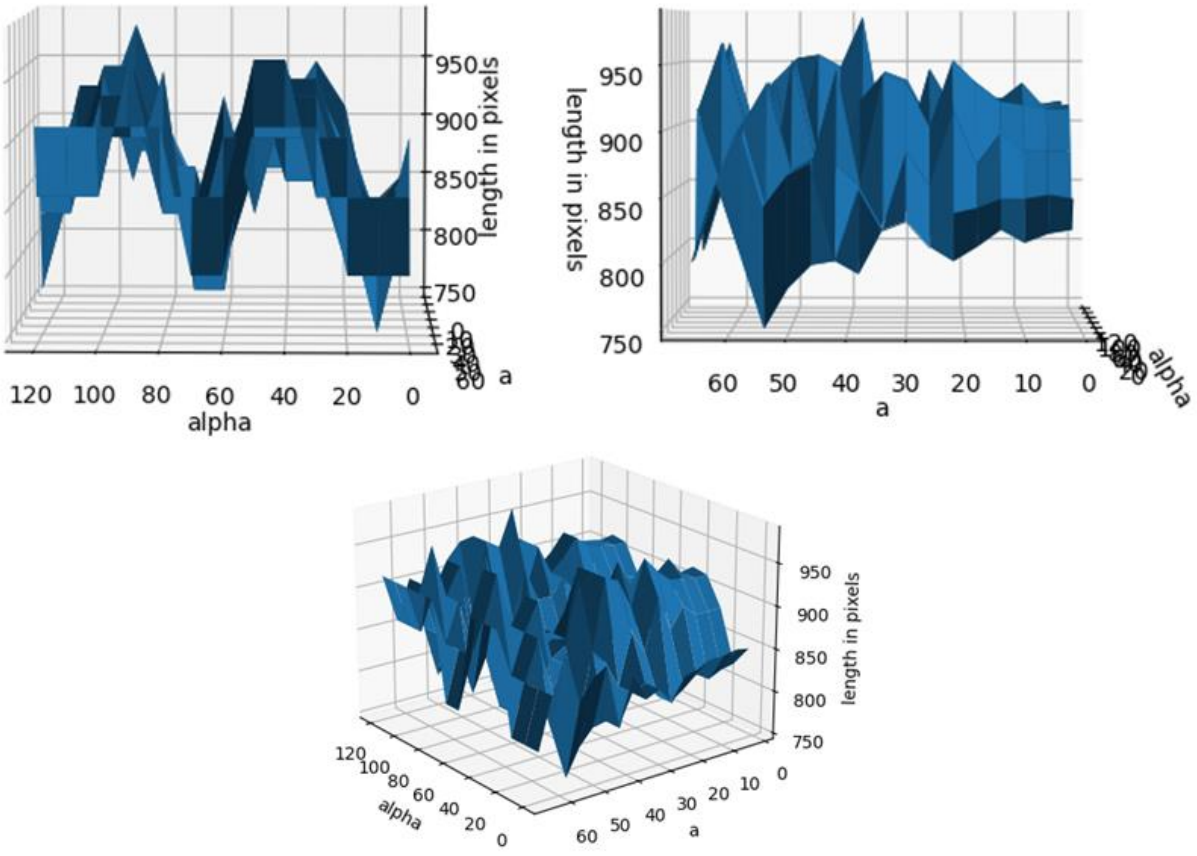

Рис. 3.13. 3D график зависимости длины оптимального маршрута в пикселях от a и от $alpha$ без ограничений

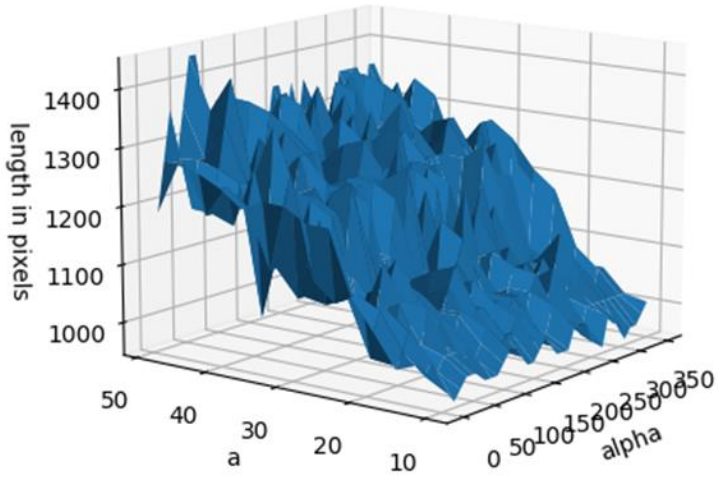

Рис. 3.14. 3D график зависимости длины оптимального маршрута в пикселях от a и от $alpha$ с ограничениями

Наконец, проведём сравнение работы алгоритма с поворотом на угол $alpha_{opt}$ и с $alpha$ равным 0°, то есть без поворота (рис. 3.15). При этом учитываются динамические ограничения.

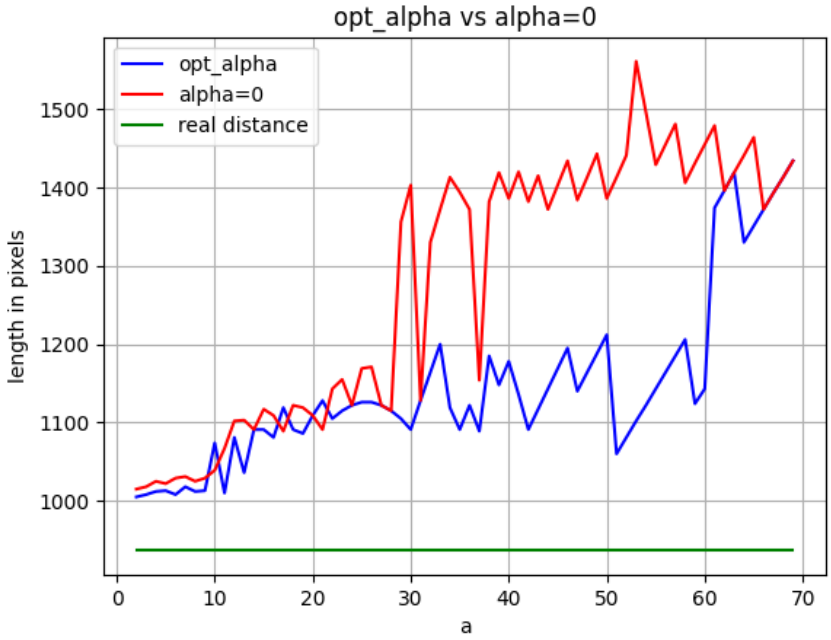

Рис. 3.15. Сравнение зависимости длины оптимального маршрута в пикселях от a при $alpha_{opt}$ (синим) и с $alpha$ = 0° (красным). Зелёным показано евклидово расстояние между стартовой и конечной точками

При маленьких a от 2 до 28 разница невелика, а при $a > 30$ составляет до 400 пикселей.

3.3. Временная сложность алгоритма

Оценим временную сложность алгоритма эмпирическим путём. На рис. 3.16 представлена зависимость времени работы алгоритма от длины стороны гекса a при $alpha$ равным 0°. a изменяется от 2 до 70 с шагом 1.

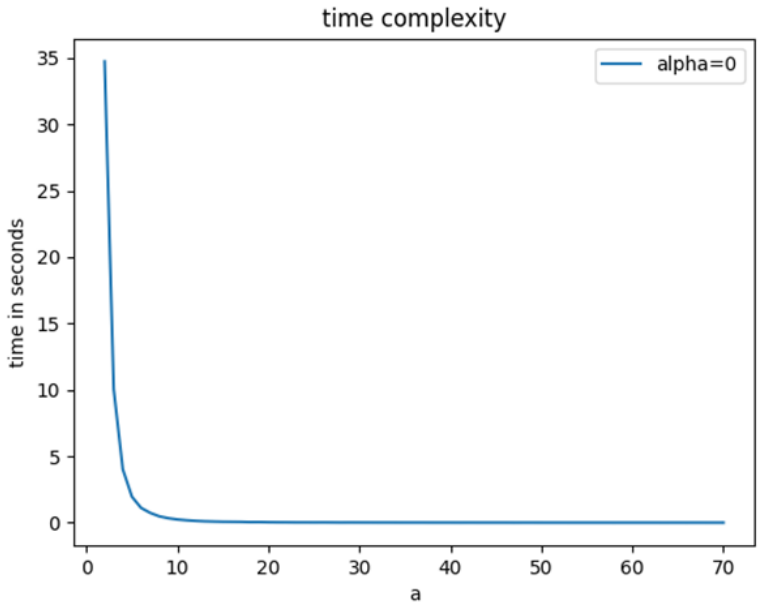

Рис. 3.16. Временная сложность алгоритма. $alpha = 0°$

Чтобы показать, что время не зависит от угла поворота, переберём несколько $alpha$ (рис. 3.17).

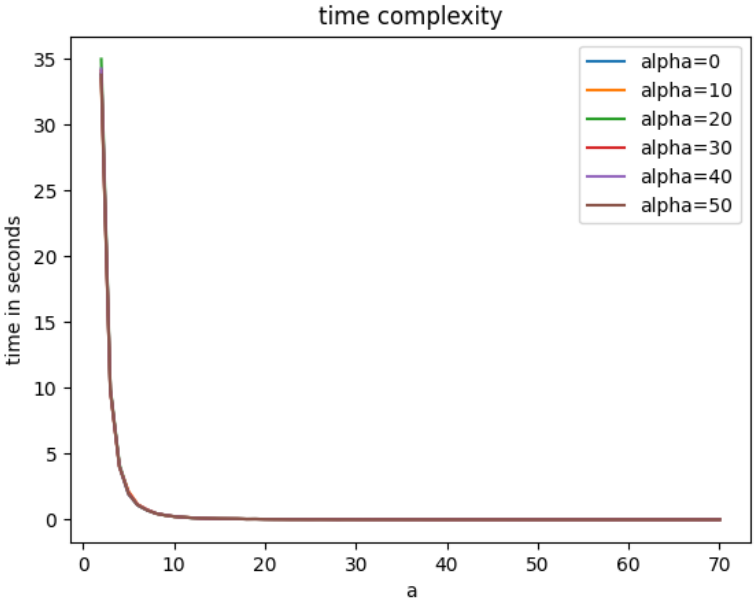

Рис. 3.17. Временная сложность алгоритма. $alpha$ изменяется от 0° до 50° с шагом 10°

Графики почти полностью совпадают, что говорит о независимости времени выполнения алгоритма от угла поворота $alpha$.

Из рис. 3.16-3.17 видно, что зависимость экспоненциальная, то есть

$$y = ke^{-ax}.$$

При помощи функции curve_fit библиотеки *SciPy* [34] были подобраны коэффициенты k и a. На рис. 3.18 сопоставляются теоретическая и эмпирическая зависимости.

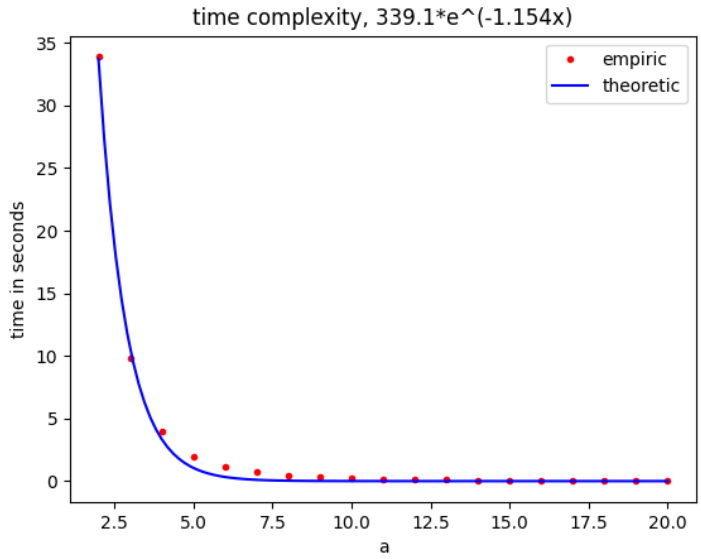

Рис. 3.18. Теоретическая и эмпирическая временная сложность алгоритма. Синей линией отражена теоретическая зависимость $t = 339.1e^{-1.154a}$, красными точками – эмпирические измерения

Итак, реализованный в данной работе алгоритм, решающий задачу weather routing, имеет **экспоненциальную сложность** $O(e^{-a})$, где a – длина стороны гекса.

3.4. Характеристики вычислительной среды

Тестирование проводилось на ЭВМ со следующими характеристиками:

- **Операционная система** – Windows 10;
- **CPU** – 11th Gen Intel® Core™ i7-11800H 2.30 GHz;
- **IDE** – PyCharm 2022.1 (Community Edition);
- **Объем ОЗУ** – 16 ГБ.

Заключение

Вопрос навигации судов был актуален с самого зарождения человеческой цивилизации. Плавание в океанских просторах всегда считалось опасным, в какой-то степени непредсказуемым предприятием. В данной работе был рассмотрен не встречавшийся ранее метод нахождения оптимального по времени маршрута, учитывающий динамически меняющиеся ограничения (например, погодные), основывающийся на гексагональной растеризации, алгоритме поиска в ширину и математической морфологии.

Были разработаны вспомогательные алгоритмы генерации случайных ограничений-многоугольников и их растеризации в гексагональном растре, а также непосредственно алгоритм поиска оптимальных маршрутов в условиях динамически меняющихся ограничений. Был проведён анализ получившегося алгоритма, показана практическая применимость на реальных тестовых данных о состоянии погоды. Разработанный алгоритм может быть адаптирован для любых движущихся в условиях динамически меняющихся ограничений объектов. Написана программная реализация, код которой находится в репозитории на github (ссылка: https://github.com/4eckah78/weather_routing).

Подводя итоги, можно сказать, что маршрутизация по погодным условиям (англ. *weather routing*) – это хотя и относительно не новая концепция, но она довольно быстро получила широкое распространение и повышенное внимание ученых и практиков в последние годы. Возможно, это связано с возросшими вычислительными возможностями современных вычислительных систем.

В целом ожидается, что область погодных маршрутов и оптимизации рейсов будет всё более привлекать исследовательский интерес в будущем. Она достаточно широка, в ней используется множество подходов, о чем свидетельствует множество статей с порой совершенно различными

разработанными методиками. Также присутствует явная экономическая выгода. Улучшение качества прогнозов погоды, доступность данных и всё увеличивающаяся вычислительная мощность открывают пути для дополнительных исследований по оптимизации погодных маршрутов. Можно с уверенностью сказать, что это перспективное поле для научных и коммерческих исследований.

Список использованных источников

[1] Фонтанка.ру [Электронный ресурс]. URL: https://www.fontanka.ru/2022/03/17/70515458/ (Дата обращения: 18.03.2022).

[2] Grifolla, M. Ship weather routing using pathfinding algorithms: the case of Barcelona – Palma de Mallorca / Manel Grifolla, Lluis Martorella, Marcella Castellsb, F. Xavier Martínez de Osés // Transportation Research Procedia. – 2018. – Vol. 33. – P. 299-306.

[3] Zis, T. Ship weather routing: A taxonomy and survey / Thalis P.V. Zis, Harilaos N. Psaraftis, Li Ding // Ocean Engineering. – 2020. – Vol. 213. – P. 1-18.

[4] Walther, L. Modeling and Optimization Algorithms in Ship Weather Routing / Laura Walther, Anisa Rizvanolli, Mareike Wendebourg, Carlos Jahn // International Journal of e-Navigation and Maritime Economy. – 2016. – Vol. 4. – P. 31-45.

[5] Takashima, K. On the Fuel Saving Operation for Coastal Merchant Ships using Weather Routing / K. Takashima, B. Mezaoui, R. Shoji // International Journal on Marine Navigation and Safety of Sea Transportation. – 2009. – Vol. 3, №4. – P. 401-406.

[6] Mannarini, G. A Prototype of Ship Routing Decision Support System for an Operational Oceanographic Service / G. Mannarini, G. Coppini, P. Oddo, N. Pinardi // International Journal on Marine Navigation and Safety of Sea Transportation. – 2013. – Vol. 7, №1. – P. 53-59.

[7] Böttner, C.-U. Weather routing for ships in degraded condition // International Symposium on Safety, Security and Environmental Protection. – 2007.

[8] Shin, Y. Near-Optimal Weather Routing by Using Improved A* Algorithm / Yongwoo Shin, Misganaw Abebe, Y. Noh, Sangbong Lee, Inwon Lee, Dong-Hyun Kim, Jungchul Bae, K. Kim // Applied Sciences. – 2020. – Vol. 10, Iss. 17, 6010.

[9] Szlapczynska, J. Multicriteria Evolutionary Weather Routing Algorithm in Practice // International Journal on Marine Navigation and Safety of Sea Transportation. – 2013. – Vol. 7, №2. – P. 61-65.

[10] Szlapczynska, J. Multicriteria Optimisation in Weather Routing / J. Szlapczynska, R. Smierzchalski // International Journal on Marine Navigation and Safety of Sea Transportation. – 2009. – Vol. 3, №4. – P. 393-400.

[11] Веремей, Е. И. Алгоритмы оптимизации маршрутов движения с учётом погодных условий / Е. И. Веремей, М. В. Сотникова // International Journal of Open Information Technologies. – 2016. – Т. 4, № 3. – С.55-61.

[12] Ван, Х. Построение маршрута с помощью улучшенного метода изохрон при минимизации времени плавания и с учетом прогноза погоды / Ван Х., Ли П., Сюэ Ю., Коровкин М. В. // Вестник Санкт-Петербургского университета. – 2017. – Т. 13, № 3. – С.286-299.

[13] Larsson, E., Simonsen, M. H. DIRECT Weather Routing, 2014. URL: https://hdl.handle.net/20.500.12380/205858.

[14] Marine Digital. Оптимизация маршрута вашего судна с учетом времени и расхода топлива [Электронный ресурс]. URL: https://marine-digital.com/ru/tpost/9d7glblfr5-optimizatsiya-marshruta-vashego-sudna-s (Дата обращения: 29.04.2022).

[15] ScienceDirect [Электронный ресурс]. URL: https://www.sciencedirect.com/ (Дата обращения: 13.03.2022).

[16] Connected Papers [Электронный ресурс]. URL: https://www.connectedpapers.com/ (Дата обращения: 13.03.2022).

[17] eLibrary.Ru [Электронный ресурс]. URL: https://www.elibrary.ru/ (Дата обращения: 13.03.2022).

[18] Hinnenthal, J. Robust Pareto-Optimum Routing of Ships utilizing Deterministic and Ensemble Weather Forecasts, 2008. URL: https://d-nb.info/988754096/34.

[19] Aneja, Y. P. Shortest Chain Subject to Side Constraints / Y. P. Aneja, V. Aggarwal, K. P. Nair // NETWORKS. – 1983. – Vol. 13, Iss. 2. – P.295-302.

[20] James, R. W. Application of wave forecast to marine navigation // Washington, D.C.: US Navy Hydrographic Office. – 1957. – 78 p.

[21] Hagiwara, H. Weather routing of (sail-assisted) motor vessels // PhD Thesis. Delft: Technical University of Delft. – 1989. – 337 p.

[22] Bijlsma, S. J. A Computational Method for the Solution of Optimal Control Problems in Ship Routing // Navigation. Journal of The Institute of Navigation. – 2001. – Vol. 48, Iss. 3. – P. 145-154.

[23] Сотникова, М. В. Алгоритмы формирования маршрутов движения судов с учётом прогноза погодных условий // Вестник Санкт-Петербургского университета. – 2009. – № 2. – С.181-196.

[24] Wei, D. Development of a 3D Dynamic Programming Method for Weather Routing / D. Wei, P. Zhou // International Journal on Marine Navigation and Safety of Sea Transportation. – 2012. – Vol. 6, №1. – P. 79-85.

[25] Chen H., A dynamic program for minimum cost ship routing under uncertainty, 1978. URL: http://hdl.handle.net/1721.1/16315

[26] Техническая документация библиотеки Pillow [Электронный ресурс]. URL: https://pillow.readthedocs.io/en/stable/ (Дата обращения: 28.09.2021).

[27] Техническая документация библиотеки Matplotlib [Электронный ресурс]. URL: https://matplotlib.org/ (Дата обращения: 08.04.2022).

[28] Жигалова А. В., Генерация случайных многоугольников, 2017. URL: http://hdl.handle.net/11701/11033.

[29] Нигмедзянова, А. М. Замощение плоскости многоугольниками в системе GeoGebra / А. М. Нигмедзянова, А. Д. Исмоиловна // Системы компьютерной математики и их приложения. – 2018. – № 19. – С.70-79.

[30] Hex Map [Электронный ресурс]. URL: https://catlikecoding.com/unity/tutorials/hex-map/part-1/ (Дата обращения: 26.04.2022).

[31] Гексагональные сетки, базовые понятия и определения [Электронный ресурс]. URL: https://www.redblobgames.com/grids/hexagons/ (Дата обращения: 25.09.2021).

[32] Алгоритм Брезенхема гексагональной растеризации отрезков [Электронный ресурс]. URL: http://zvold.blogspot.com/2010/01/bresenhams-line-drawing-algorithm-on_26.html (Дата обращения: 25.09.2021).

[33] Алгоритмы заливки многоугольников [Электронный ресурс]. URL: https://habr.com/ru/post/116398/ (Дата обращения: 05.05.2022).

[34] Техническая документация библиотеки SciPy [Электронный ресурс]. URL: https://scipy.org/ (Дата обращения: 13.05.2022).