Проект по курсу «Эвристические методы планирования»

Алгоритмы поиска с бикритериальной оптимизацией

https://github.com/ugadiarov-la-phystech-edu/hs-project

Угадяров Л.А.

МФТИ, группа М05-006а

11 мая 2021 г.

1 Задачи многокритериальной оптимизации

Многокритериальная оптимизация заключается в поиске множества оптимальных решений с учётом нескольких неоднородных критериев, которые невозможно свести друг к другу.

Практические приложения:

- Прокладка телекомуникационных сетей: минимизация стоимость и вероятность отказа
- Планирование в робототехнике: минимизация длины пути и потребления энергии
- Езда на велосипеде: минимизация длины пути и максимизация безопасности велосипедиста
- Грузоперевозки: минимизация стоимости транспортировки, минимизация времени в пути, учёт экологических факторов
- Перевозки опасных грузов: минимизация длины пути, минимизация риска человеческих жертв при возможной аварии
- Пассажирские перевозки: минимизация стоимости проезда, времени в пути, количества пересадок
- Планирование спутниковой фотосъёмки: удовлетворение максимального количества запросов пользователей с учётом приоритета запросов, минимизация износа оборудования

2 Математическая постановка бикритериальной задачи

Определение. Пусть $\mathbb{p} = (p_1, p_2)$ и $\mathbb{q} = (q_1, q_2)$ — пары вещественных чисел, тогда:

- $\mathbb{p} \prec \mathbb{q}$ ($\mathbb{p}$ доминирует $\mathbb{q}$), если $(p_1 < q_1) \wedge (p_2 \leq q_2)$ или $(p_1 = q_1) \wedge (p_2 < \_2)$
- $\mathbb{p} \leq \mathbb{q}$ ($\mathbb{p}$ слабо доминирует $\mathbb{q}$), если $(p_1 \leq q_1) \wedge (p_2 \leq q_2)$

Определение. Бикритериальная задача поиска маршрутов с наименьшей стоимостью $\mathcal{U} = (S, E, \mathbb{h}, \mathbb{c}, s_{start}, s_{goal})$, где:

- S — конечное множество состояний
- $E \subseteq S \times S$ — множество рёбер
- $\mathbb{c} : E \to \mathbb{R}^+ \times \mathbb{R}^+$ — функция стоимости, $\mathbb{c}(e) = (c_1(e), c_2(e))$
- $\mathbb{h} : S \to \mathbb{R}^+ \times \mathbb{R}^+$ — эвристическая функция, $\mathbb{h}(s) = (h_1(s), h_2(s))$
- s_{start} — начальное состояние
- s_{goal} — целевое состояние

Определение. $\pi(s_1, s_n) = s_1, \ldots, s_n$ — маршрут из s_1 в s_n, где $\{s_i\} \subseteq S$ и $\{(s_i, s_{i+1})\} \subseteq E$.
Определение. $\mathbb{c}(\pi) = \sum_{i=1}^{n-1} \mathbb{c}(s_i, s_{i+1})$ — стоимость маршрута π.
Определение. Маршрут $\pi(s_1, s_n)$ доминирует маршрут $\pi^{'}(s_1, s_n) : \pi \prec \pi^{'} \Leftrightarrow \mathbb{c}(\pi) \prec \mathbb{c}(\pi^{'})$.
Определение. Маршрут $\pi(s_{start}, s_{goal})$ называется парето-оптимальным решением задачи $\mathcal{U} \Leftrightarrow \nexists \pi^{'}(s_{start}, s_{goal}) : \pi^{'} \prec \pi$.
Определение. Решением задачи $\mathcal{U}$ называется множества всех парето-оптимальных решений с уникальной стоимостью.
Замечание. Рассматриваются монотонные эвристические функции: $\mathbb{h}(s_{goal}) = (0, 0)$ и $\forall (s, t) \in E \;\; \mathbb{h}(s) \leq \mathbb{c}(s, t) + \mathbb{h}(t)$.

Отношение доминирования задаёт частичный порядок на множестве пар, т.е. не все пары сравнимы друг с другом. Таким образом множество парето-оптимальных решений можно рассматривать как множество всех решений, которые являются лучшими среди тех, с которыми их можно сравнить. Пример парето-оптимального множества представлен на рисунке 1.

3 Общий подход к решению бикритериальной задачи

Адаптация алгоритма A^* для решения бикритериальной задачи.
Изменения в списке $OPEN$ узлов – кандидатов на раскрытие:

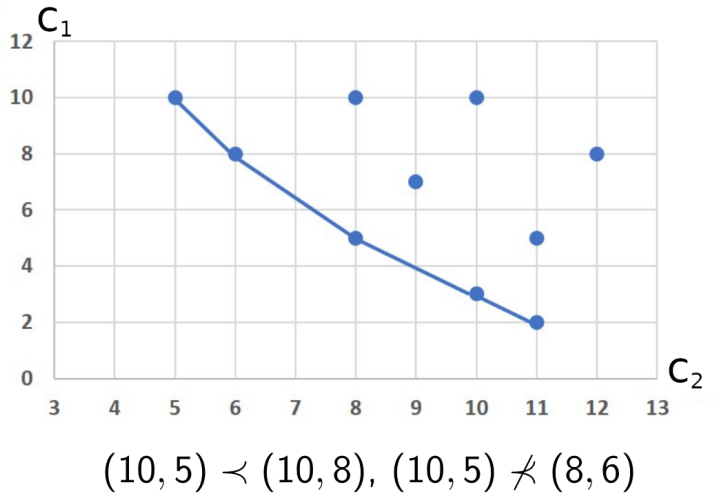

Рис. 1: Пример парето-оптимального множества. Оптимальное решение $(10, 5)$ доминирует неоптимальное решение $(10, 8)$. Оптимальные решения $(10, 5)$ и $(8, 6)$ не доминируют друг друга.

- $OPEN$ содержит нераскрытые узлы — кортежи $x = (s, g, f)$, где g и f – векторы
- Для состояния s в $OPEN$ одновременно могут находится несколько узлов (s, g_1, f_1) и (s, g_2, f_2)

Выбор узла из $OPEN$:

- Для раскрытия выбирается такой узел $(s, g, f) \in OPEN$, что $\nexists(s^{'}, g^{'}, f^{'}) \in OPEN : f^{'} \prec f$
  Для бикритериальной задачи это условие эквивалентно извлечению узла с лексикографически минимальным значением f

Обработка узлов вида (s_{goal}, g, f):

- Поддерживается множество найденный решений: $SOL = \{\pi_i(s_{start}, s_{goal})\}$
- Если для найденного маршрута $\pi(s_{start}, s_{goal}) \Rightarrow \nexists\pi^{'} \in SOL : \pi^{'} \prec \pi$, то удаляем из SOL все маршруты $\tilde{\pi} : \pi \prec \tilde{\pi}$ и добавляем π в SOL

Раскрытие узла (s, g, f):

- Для каждого дочернего состояния $s^{'} \in Succ(s)$ строится узел $(s^{'}, g^{'}, f^{'})$
- Дочерний узел $(s^{'}, g^{'}, f^{'})$ добавляется в $OPEN$ при одновременном выполнении двух условий:

- $\nexists (s^{'}, \tilde{g}, \tilde{f}) \in OPEN : \tilde{f} \prec f^{'}$
- $\nexists \pi \in SOL : \mathbb{c}(\pi) \prec f^{'}$

- Если $(s^{'}, g^{'}, f^{'})$ добавляется в $OPEN$, то из $OPEN$ удаляются все узлы $(s^{'}, \tilde{g}, \tilde{f}) : f^{'} \prec \tilde{f}$

Если $OPEN$ пустой, то алгоритм завершает работу и возвращает SOL.

Следует отметить, что общий подход совершает большое количества проверок доминирования, что сказывается на производительности.

4 Алгоритмы бикритериальной оптимизацией

Проект основан на работе [1], в которой описаны алгоритмы NAMOA*, NAMOA*dr, BOA*, выполняющие бикритериальный поиск. Псевдокод алгоритма NAMOA* представлен на рисунке 2. Псевдокод алгоритма BOA* представлен на рисунке 3. Ниже описаны основные особенности алгоритмов.

Алгоритм NAMOA*:

- Реализация общего подхода с незначительными оптимизациями за счёт поддержки множества всех раскрытых узлов $G_{cl}(s)$

Алгоритм NAMOA*dr

- Оптимизация операции добавления дочернего узла в $OPEN$ для случая монотонной эвристической функции и извлечения узлов из $OPEN$ в лексикографическом порядке по f:
  - Проверка $\nexists \pi \in SOL : \mathbb{c}(\pi) \prec (f_1^{'}, f_2^{'})$ заменяется на $\min_{\pi \in SOL} c_2(\pi) \geq f_2^{'}$
  - В некоторых случаях проверку $\nexists (s^{'}, \tilde{g}, \tilde{f}) \in OPEN : \tilde{f} \prec f^{'}$ можно можно заменить на $\min_{\tilde{f} \in OPEN} \tilde{f}_2 \geq f_2^{'}$

Алгоритм BOA*:

- Для случая монотонной эвристической функции и извлечения узлов из $OPEN$ в лексикографическом порядке по f авторами работы доказаны ещё более сильные утверждения, которые позволяют все проверки доминирования совершать за константное время за счёт поддержки $g_2^{min}(s)$ — минимального значения g_2 для раскрытых узлов с состоянием s

- В узлах дополнительно хранятся ссылки на родительские узлы: $x = (s, g, f, parent(x))$

Algorithm 1: NAMOA*

Input : A search problem $(S, E, \mathbf{c}, s_{start}, s_{goal})$ and a consistent heuristic function $\mathbf{h}$
Output: The Pareto-optimal solution set

$sols \leftarrow \emptyset$
for each $s \in S$ **do**
  $\mathbf{G}_{op}(s) \leftarrow \emptyset$; $\mathbf{G}_{cl}(s) \leftarrow \emptyset$
$\mathbf{G}_{op}(s) \leftarrow \{(0,0)\}$
$parent((0,0)) \leftarrow \emptyset$
Initialize $Open$ and add $(s_{start}, (0,0), \mathbf{h}(s_{start}))$ to it
while $Open \neq \emptyset$ **do**
  Remove a node $(s, \mathbf{g}_s, \mathbf{f}_s)$ from $Open$ with the lexicographically smallest f-value of all nodes in $Open$
  Remove $\mathbf{g}_s$ from $\mathbf{G}_{op}(s)$ and add it to $\mathbf{G}_{cl}(s)$
  if $s = s_{goal}$ **then**
    Add $\mathbf{g}_s$ to $sols$
    Remove all nodes $(u, \mathbf{g}_u, \mathbf{f}_u)$ with $\mathbf{f}_s \prec \mathbf{f}_u$ from $Open$
    continue
  for each $t \in \mathtt{Succ}(s)$ **do**
    $\mathbf{g}_t \leftarrow \mathbf{g}_s + \mathbf{c}(s,t)$
    if $\mathbf{g}_t \in \mathbf{G}_{op}(t) \cup \mathbf{G}_{cl}(t)$ **then**
      Add $\mathbf{g}_s$ to $parent(\mathbf{g}_t)$
      continue
    if $\mathbf{G}_{op}(t) \cup \mathbf{G}_{cl}(t) \prec \mathbf{g}_t$ **then**
      continue
    $\mathbf{f}_t \leftarrow \mathbf{g}_t + \mathbf{h}(t)$
    if $sols \prec \mathbf{f}_t$ **then**
      continue
    Remove all g-values $\mathbf{g}'_t$ from $\mathbf{G}_{op}(t)$ that are dominated by $\mathbf{g}_t$ and remove their corresponding nodes $(t, \mathbf{g}'_t, \mathbf{f}'_t)$ from $Open$
    Remove all g-values from $\mathbf{G}_{cl}(t)$ that are dominated by $\mathbf{g}_t$
    $parent(\mathbf{g}_t) \leftarrow \{\mathbf{g}_s\}$
    Add $\mathbf{g}_t$ to $\mathbf{G}_{op}(t)$
    Add $(t, \mathbf{g}_t, \mathbf{f}_t)$ to $Open$
return $sols$

Рис. 2: Псевдокод алгоритма NAMOA*

Авторы работы [1] сравнили производительность BOA* с другими алгоритмами на дорожных картах городов США соревнования 9th DIMACS Implementation Challenge - Shortest Paths (`http://www.diag.uniroma1.it/challenge9/download.shtml`). В качестве критериев используется длина марш-

Algorithm : Bi-Objective A* (BOA*)

Input : A search problem $(S, E, \mathbf{c}, s_{start}, s_{goal})$ and a consistent heuristic function $\mathbf{h}$

Output: A cost-unique Pareto-optimal solution set

```
1  sols ← ∅
2  for each s ∈ S do
3      g_2^min(s) ← ∞
4  x ← new node with s(x) = s_start
5  g(x) ← (0, 0)
6  parent(x) ← null
7  f(x) ← (h_1(s_start), h_2(s_start))
8  Initialize Open and add x to it
9  while Open ≠ ∅ do
10     Remove a node x from Open with the
         lexicographically smallest f-value of all nodes in
         Open
11     if g_2(x) ≥ g_2^min(s(x)) ∨ f_2(x) ≥ g_2^min(s_goal) then
12         continue
13     g_2^min(s(x)) ← g_2(x)
14     if s(x) = s_goal then
15         Add x to sols
16         continue
17     for each t ∈ Succ(s(x)) do
18         y ← new node with s(y) = t
19         g(y) ← g(x) + c(s(x), t)
20         parent(y) ← x
21         f(y) ← g(y) + h(t)
22         if g_2(y) ≥ g_2^min(t) ∨ f_2(y) ≥ g_2^min(s_goal) then
23             continue
24         Add y to Open
25 return sols
```

Рис. 3: Псевдокод алгоритма BOA*

рута и время движения по маршруту:

- NAMOA*, NAMOA*dr, BOA* — авторские реализации (язык Си)
- sBOA* — модификация BOA* без применения оптимизаций, приводящих к константному времени проверки условий доминирования, авторская реализация (язык Си)
- Оригинальные реализации на Си для Bi-Objective Dijkstra (BDijkstra) и Bidirectional Bi-Objective Dijkstra (BBDijkstra) предоставлены авторами этих алгоритмов

Некоторые результаты работы алгоритмов представлены на рисунке 4.

New York City (NY)				
264,346 states, 730,100 edges, $\|sols\|$ = 199 on average				
	Solved	Average	Max	Min
NAMOA*	50/50	157.17	1,936.36	**0.02**
sBOA*	50/50	9.75	148.65	0.10
NAMOA*dr	50/50	0.65	4.99	0.11
BOA*	50/50	**0.32**	**1.95**	0.11
BBDijkstra	50/50	1.94	23.43	0.26
BDijkstra	50/50	2.55	21.16	0.17

San Francisco Bay (BAY)				
321,270 states, 794,830 edges, $\|sols\|$ = 119 on average				
	Solved	Average	Max	Min
NAMOA*	50/50	58.87	1,474.76	**0.02**
sBOA*	50/50	3.38	120.57	0.12
NAMOA*dr	50/50	0.38	6.08	0.12
BOA*	50/50	**0.29**	**4.17**	0.12
BBDijkstra	50/50	0.87	9.61	0.28
BDijkstra	50/50	1.83	33.39	0.22

Рис. 4: Сравнение алгоритмов бикритериальной оптимизации на дорожных картах городов США из 9th DIMACS Implementation Challenge для 50 случайно выбранных пар начальная вершина - целевая вершина. Average, Max, Min – время выполнения алгоритма в секундах

5 Собственная реализация алгоритмов NAMOA* и BOA*

В рамках проекта были реализованы алгоритмы BOA*, NAMOA*.
Репозиторий проекта с тетрадкой Jupyter Notebook и всеми необходимыми данными: `https://github.com/ugadiarov-la-phystech-edu/hs-project`.
Используемый язык программирования: `Python 3.6`.
Задействованные библиотеки:

- NetworkX, версия 2.5
- Numpy, версия 1.19.2
- Pandas, версия 1.0.5
- Matplotlib, версия 3.2.2
- tqdm, версия 4.47.0

Используемые данные:

- Граф расстояний и граф времени в пути для New York City из 9th DIMACS Implementation Challenge:
  `http://www.diag.uniroma1.it/challenge9/download.shtml`
- Карта duskwood из Moving AI Lab:
  `https://movingai.com/benchmarks/wc3maps512/index.html`

Особенности реализации NAMOA*:

- Класс `OpenNAMOA` представляет список узлов - кандидатов на раскрытие - для алгоритма NAMOA*. Список реализован с помощью кучи и предоставляет интерфейс для эффективного извлечения лексикографически минимального по f-значению узла. Удаление других элементов из списка происходит ленивым способом: удаляемый элемент помечается как удалённый, а фактическое удаление происходит только при извлечении элемента с вершины кучи
- Функция `namoa` реализует алгоритм NAMOA*. Принимает на вход граф `graph` — экземпляр класса `networkx.classes.graph.Graph`, `start` — стартовая вершина, `goal` — целевая вершина, `weights` — пара названий атрибутов ребра, которые используются в качестве весов. Функция возвращает словарь с элементами: `solutions` — парето-оптимальное множество весов кротчайших путей из `start` в `goal`, `parent` — множество родительских вершин, `n_expansions` — количество совершённых раскрытий вершин при выполнении алгоритма, `runtime` — время выполнения в секундах, `max_size_invalid` максимальный размер списка `OpenNAMOA` с учётом помеченных «удалённых» вершин, `max_size` максимальный размер списка `OpenNAMOA` без учёта помеченных «удалённых» вершин.

Особенности реализации BOA*:

- Класс `OpenBOA` представляет список узлов - кандидатов на раскрытие - для алгоритма BOA*. Список реализован с помощью кучи и предоставляет интерфейс для эффективного извлечения лексикографически минимального по f-значению узла. Другие операции удаления из списка не реализованы, т.к. алгоритм BOA* не требует таких операций
- Функция `boa` реализует алгоритм NAMOA*. Принимает на вход граф `graph` — экземпляр класса `networkx.classes.graph.Graph`, `start` — стартовая вершина, `goal` — целевая вершина, `weights` — пара названий атрибутов ребра, которые используются в качестве весов. Функция возвращает словарь с элементами: `solutions` — множество узлов - экземпляров класса `Node` для целевой вершины `goal`, g-значения которых образуют парето-оптимальное множество весов кротчайших путей из `start` в `goal` (оптимальные пути восстанавливаются с помощью атрибута `parent` экземпляров `Node`), `n_expansions` — количество совершённых раскрытий вершин при выполнении алгоритма, `runtime` —

время выполнения в секундах, `max_size` максимальный размер списка `OpenBOA`.

6 Проверка корректности реализаций NAMOA* и BOA*

Корректность реализаций NAMOA* и BOA* были проверены дважды:

1. **Проверка на случайном графе из 1000 вершин**. Проверка корректности на случайном графе, который был сгенерирован с помощью библиотеки NetworkX. Параметры графа:
   - Количество вершин — 1000
   - Вероятность создания ребра между вершинами — 0.1
   - Веса рёбер – кортежи из двух элементов — случайные целые числа от 1 до 1000
2. **Проверка на подграфе дорожной сети New York City (DIMACS) из 10000 вершин**.

Результаты подтвердили корректность реализаций алгоритмов BOA* и NAMOA*.

7 Сравнение производительности реализаций NAMOA* и BOA*

Производительность алгоритмов сравнивалась на 100 случайно выбранных парах вершин из графа дорожной сети New York City (DIMACS). Результаты представлены в таблицах 1 и 2.

	Количество раскрытий	Время выполнения, с	Максимальный размер списка OPEN
Min	$4.9 * 10^1$	1.8	$2.8 * 10^1$
Mean	$4.9 * 10^5$	27.9	$2.2 * 10^4$
Max	$8.5 * 10^6$	521.7	$2.4 * 10^5$
Std	$1.3 * 10^6$	78.4	$3.6 * 10^4$

Таблица 1: Производительность реализации BOA*

	Количество раскрытий	Время выполнения, с	Максимальный размер списка OPEN (без учёта удалённых вершин)	Максимальный размер списка OPEN
Min	$4.9 * 10^1$	1.6	$2.2 * 10^1$	$2.7 * 10^1$
Mean	$4.9 * 10^5$	189.9	$1.3 * 10^4$	$1.9 * 10^4$
Max	$8.5 * 10^6$	4236.6	$1.5 * 10^5$	$2.1 * 10^5$
Std	$1.3 * 10^6$	643.7	$2.3 * 10^4$	$3.2 * 10^4$

Таблица 2: Производительность реализации NAMOA*

Сравнение производительности реализаций BOA* и NAMOA*:

- NAMOA* и BOA* делают одинаковое количество раскрытий узлов
- NAMOA* работает в среднем в 6.8 раз медленне
- Размер списка OPEN для алгоритма NAMOA* в среднем в 1.15 раз меньше

8 Применение BOA* для поиска множества оптимальных путей

Рассматривается применение реализации BOA* к задаче поиска оптимальных путей на примере 8-связной карты duskwood (`https://movingai.com/benchmarks/wc3maps512/index.html`) с двумя критериями: минимизация длины пути и максимизация безопасности пути. Безопасность проходимой клетки определяется как расстояние до ближайшей непроходимой клетки. Безопасность маршрута принимается равной сумме значений метрики безопосности для каждой клетки маршрута.
Особенности реализации:

- Класс `PassableMap` предоставляет интерфейс для поиска парето-оптимального множества путей между парой проходимых клеток по двум критериям: длинна пути и безопасность. Параметры конструктора: `symbol_map` — карта символов, по которой будет построен 8-связный граф проходимых клеток (см. формат карты: `https://movingai.com/benchmarks/formats.html`); `passable_symbol` - символ проходимой клетки.
- Метод `passable_coordinates` возвращает список координат всех проходимых клеток
- Метод `danger_score` возвращает словарь, ключом которого является координата проходимой клетки, а значением - величина «опасности» клетки:

$$\forall v \in V \Rightarrow danger(v) = -safety(v) + \max_{v^* \in V} safety(v^*) + 1,$$

где максимум берётся по всем проходимым клеткам графа. С учётом определения и аддитивности метрик $danger$ и $safety$ по длине пути, максимизация безопасности эквивалентна минимизации опасности. Параметр `normalize` позволяет нормировать метрику опасности на 1.

- Метод `optimal_routes` принимает на вход начальную вершину `start` и целевую `goal` и возвращает множество путей, оптимальное по длине и опасности, а также веса этих путей в виде словаря с ключами `routes` и `costs`.

На рисунке 5 показана оригинальная карта duskwood и визуализация метрики «опасность».

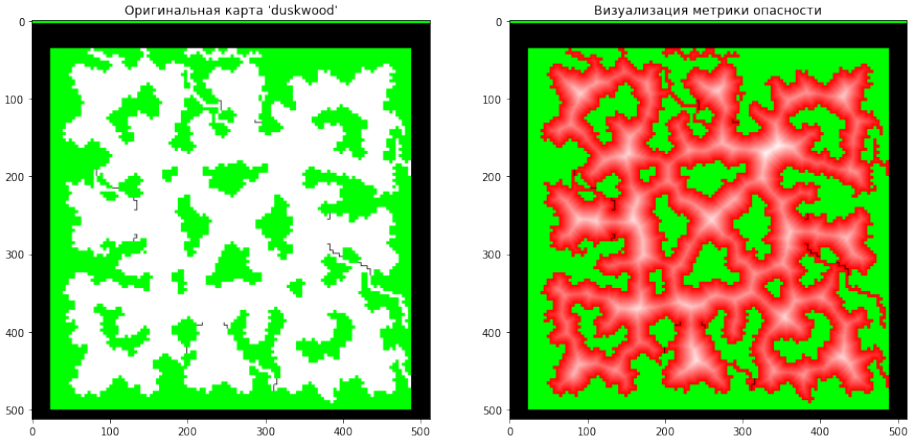

Рис. 5: Визуализация карты duskwood (слева) и метрики «опасность» (справа). Чёрные и зелёные клетки не являются проходимыми. Безопасность проходимой клетки принимается равной расстоянию до ближайшей непроходимой клетки. Интенсивностью красного цвета показана опасность клетки.

В качестве примера с помощью алгоритма BOA* было построено множества парето-оптимальных маршрутов из клетки (100, 110) в клетку (430, 400). Полученное множество содержит 258 маршрутов и показано на рисунке 6. График стоимости парето-оптимальных маршрутов показан на рисунке 7.

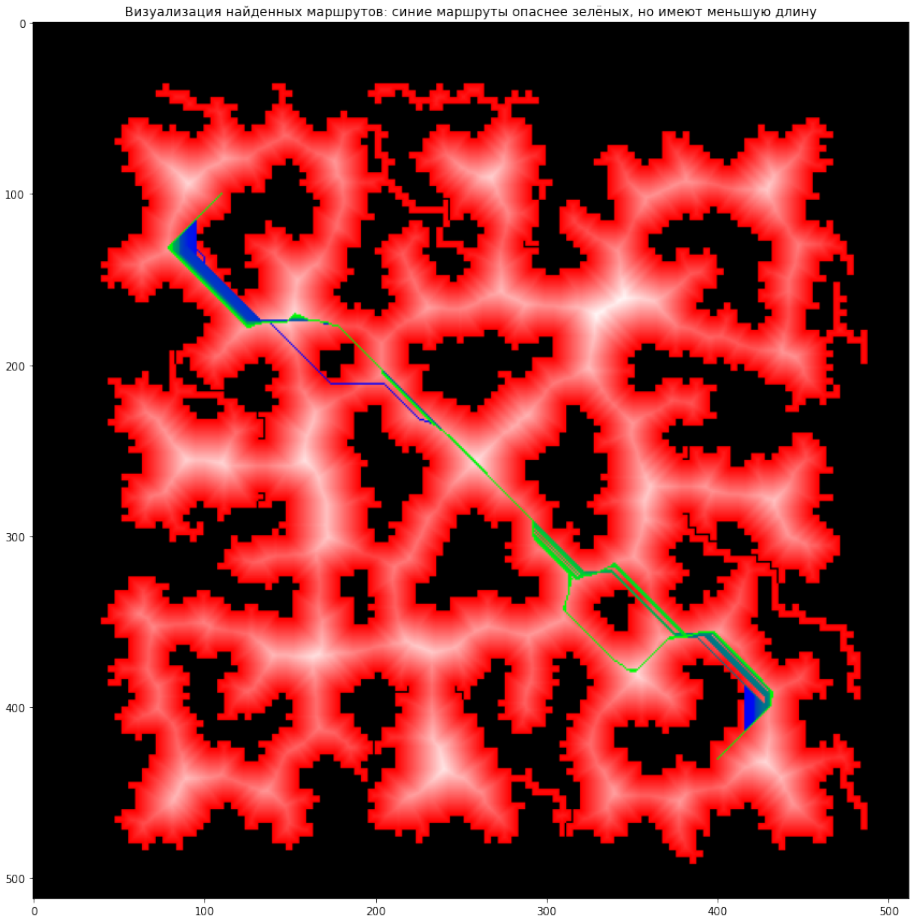

Рис. 6: Визуализация парето-оптимального множества маршрутов из клетки (100, 110) в клетку (430, 400) для карты duskwood. Градацией цвета от зелёного к синему показано увеличение опасности маршрута.

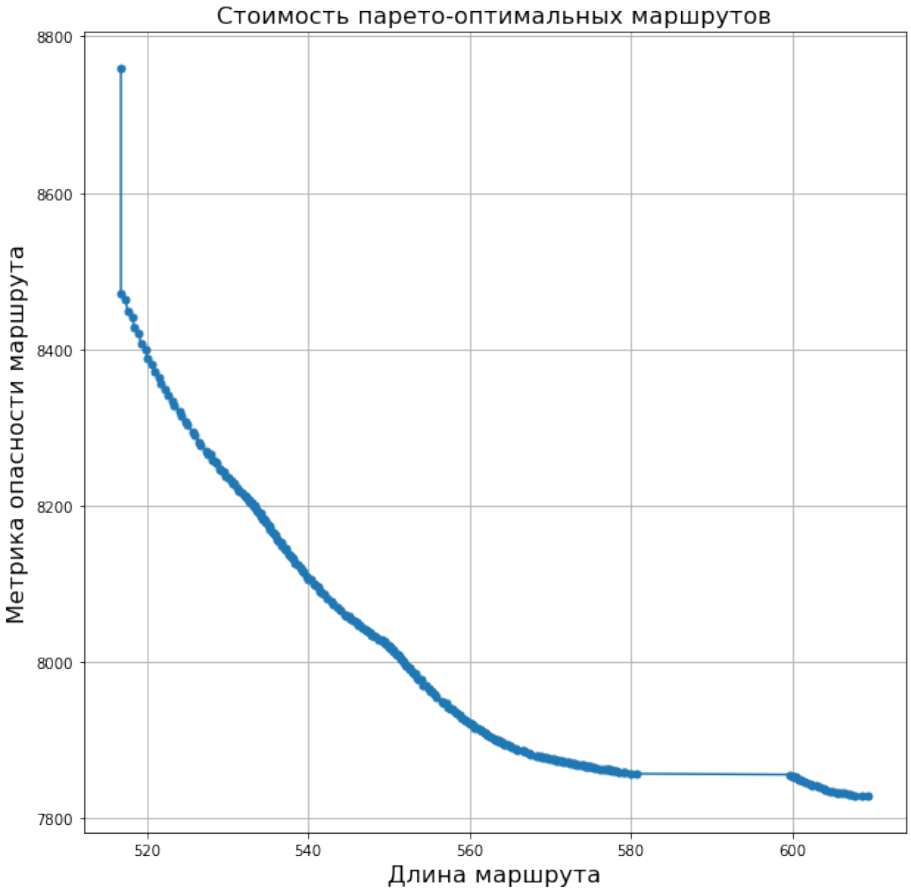

Рис. 7: График стоимости парето-оптимальных маршрутов из клетки (100, 110) в клетку (430, 400) для карты duskwood.

Список литературы

[1] Carlos Hernández Ulloa и др. «A Simple and Fast Bi-Objective Search Algorithm». в: *Proceedings of the International Conference on Automated Planning and Scheduling* 30.1 (июнь 2020), с. 143—151. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/6655.