Декомпозиция при поиске суб-оптимальных решений

Мария Малышева, Вадим Салаватов

1 июня 2021 г.

1 Постановка задачи

1.1 Постановка задачи

Дана сетчатая карта, с размеченными проходимыми и непроходимыми клетками; на ней используется евклидова метрика, связность – 8. Необходимо найти суб-оптимальный путь между двумя клетками, т.е. такой, что $\text{cost}(\text{a} \to \text{b}) \leq w \cdot \text{cost}^*(a \to b)$;

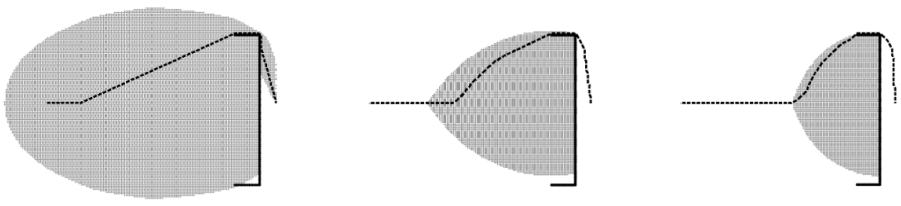
1.2 Входные и выходные данные

На вход программа получает карту (.map) и задачу (запись из .scen) в формате[1], а также параметры алгоритма и путь до файла, в который нужно записать результаты. В результате генерируется log-файл с найденным путём, а также некоторые метрики (длина пути, затраченное время, память, число раскрытых вершин).

2 R* search

2.1 Мотивация

- Проблема WA* заключается в том, что он обходит все вершины, для которых f − value меньше f − value целевой вершины;
- Площадь областей, где такое выполняется, может быть довольно большой из-за разного рода тупиков на пути между заданными вершинами;



(a) A* (b) wA*, $w = 2$ (c) wA*, $w = 5$

2.2 Предложенное решение [2]

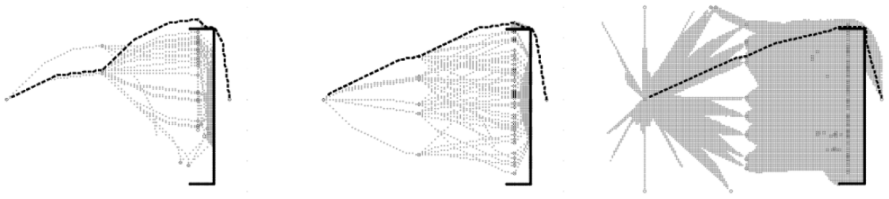
Возьмем за основу WA*. При раскрытии, вместо непосредственных соседей очередной вершины, будем добавлять в OPEN K случайных вершин на расстоянии Δ от текущей. Выбранные вершины связываются ребрами с раскрытой вершиной в некотором отдельном графе Γ, ответ будет является путём в этом графе. Ребра Γ на самом деле нужно будет уточнить до конкретных путей в исходном графе, откладываем эти подзадачи поиска пока можем.

А именно, при попытке раскрытия вершины, если до нее ещё не известен путь от родительской вершины в Γ, мы сначала пытаемся найти этот путь, запустив WA* с ограничением на «сложность пути» (число раскрытий, timeout, etc.). Если не смогли найти путь, пометим вершину AVOID меткой — такие вершины имеют наименьший приоритет в OPEN, — а также обновим нижнюю оценку на длину рассматриваемого ребра Γ. Иначе назначим стоимостью ребра стоимость найденного пути. Остальное $\pm$ повторяет WA*.

2.3 Свойства алгоритма

- Меньшая зависимость от качества эвристической функции за счёт рандомизации;
- Меньшее потребление памяти — нам необязательно хранить состояния запусков WA*, так как их относительно дешево можно посчитать заново;
- Более высокая скорость нахождения решения за счет быстрого выхода из областей с малым f − value;
- Теоретически доказуемая субоптимальность с контролируемым коэффициентом этой самой субоптимальности (почти как в WA*, но с пометкой «почти наверное» и ω^2 вместо ω);

2.4 Пример работы алгоритма из статьи

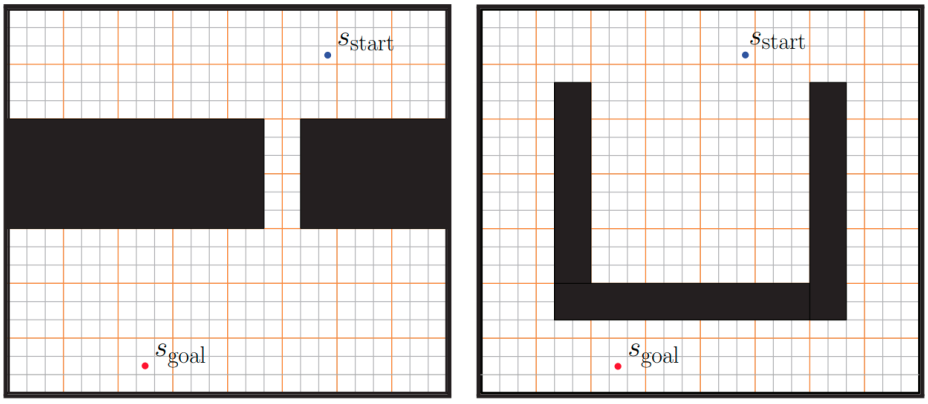


(a) R*, $w = 2$ (b) R*, $w = 1.5$ (c) R*, $w = 1$

3 MRA* search

3.1 Мотивация

- При поиске пути в непрерывном пространстве часто прибегают к дискретизации, а потом применяют A* или WA*;
- Чем выше разрешение, тем лучше качество решения, но его нахождение требует больших затрат по времени и по памяти;
- На картах с низким разрешением качество решения хуже, может пропасть связность;



(a) Narrow passage (b) Cul-de-sac

3.2 Предложенное решение

Авторы статьи [3] предлагают алгоритм MRA*, объединяющий плюсы поиска в пространствах с высоким и низким разрешением.

Идея алгоритма: пусть на каждом уровне дискретизации будет своя очередь OPEN$_i$, а состояния будут общими, то есть все состояния, у которых центр клетки совпадает, объединяются в одно. Каждый шаг алгоритма это выбор очереди и затем один шаг алгоритма WA* на соответствующем уровне дискретизации. Новые состояния после раскрытия добавляются/обновляются во всех очередях (на тех уровнях, где данное состояние есть).

ω_1 - параметр каждого WA*, ω_2 - коэффициент субоптимальности всего алгоритма.

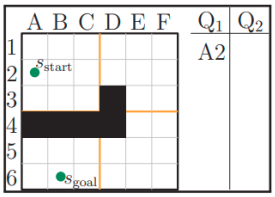
Для того, чтобы гарантировать выполнение ограничения на субоптимальность вводится нулевой уровень (т. н. якорь), на котором выполняется обычный A*. Раскрытие вершин на этом уровне происходит тогда и только тогда, когда в очереди OPEN$_i$, выбранной для раскрытия, значение минимального элемента превышает значение минимального элемента в очереди OPEN$_0$ в ω_2 раз. В таком случае раскрытие вершины из OPEN$_i$ откладывается, а раскрывается вершина из OPEN$_0$.

3.3 Мета-алгоритмы

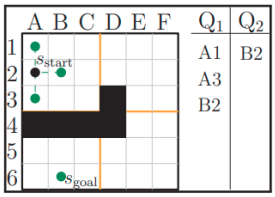
Качество работы алгоритма (время работы, количество раскрытых вершин) в значительной степени зависит от того, каким способом выбирается очередь для раскрытия. В статье [4] предложены следующие варианты:

- Round-Robin - перебирает очереди по циклу
- Meta-A* - рассматривает поиск на каждом уровне как граф, где:
  $G_m(i)$ - количество раскрытий на уровне i
  $H_m(i) = \min\limits_{s \in \text{OPEN}_i} h(s) / \max \Delta h_i$
- Dynamic Thompson Sampling - решает задачу о многоруком бандите с помощью обучения с подкреплением (дает награду, если при раскрытии удалось получить вершину с меньшим значением эвристической функции)

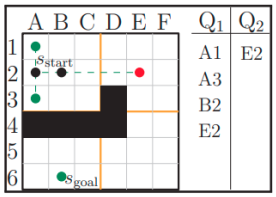
3.4 Пример работы



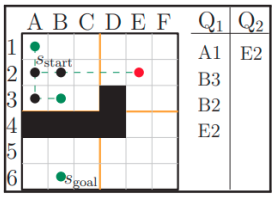
(a) MRA* is initialized. State s_{start} (A2) is inserted into Q_1.



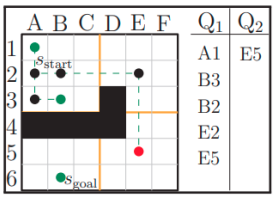
(b) State A2 is expanded by high resolution search. Since state B2 lies at the center of a coarse cell, it is also inserted into Q_2.



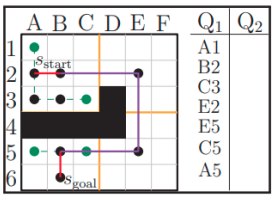
(c) State B2 is expanded by low resolution search. The successor E2 is inserted into both queues.



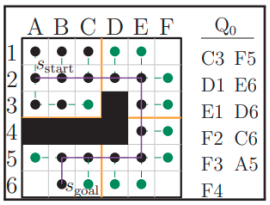
(d) State A3 is expanded by high resolution search.



(e) State E2 is expanded by low resolution search and the successor E5 is inserted into both queues.



(f) The last step when s_{goal} is expanded by high resolution search. A solution (solid line segments) is found with 8 expansions in total.



(g) The final status of a high resolution search. The same solution (purple) is found with 17 expansions.

3.5 Свойства алгоритма

- Гарантия субоптимальности - $\forall i \in \{0, ..., n\}\ g_i(s_{goal}) \leq \omega_2 \cdot g_0^*(s_{goal})$
- Более высокая скорость нахождения решения по сравнению с WA* с максимальным разрешением и таким же весом

4 Результаты работы

4.1 Что сделано

- Реализовали WA*, R*, MRA* и бенчмарки на C++, написали ноутбук для аналитики на Python;
- Провели эксперименты на картах и заданиях из датасета MovingAI [1];

4.2 R*

Стоит отметить, что в статье явным образом не говорится о следующих вещах:

1. как генерировать succ;
2. какое именно использовать отсечение WA*;
3. нужно ли обрабатывать AVOID состояния как-то иначе, или же единственное их отличие – низкий приоритет.

В связи с этим, мы реализовали данные составляющие алгоритма наиболее подходящими, на наш взгляд, образами. А именно:

1. выбираем случайно $r \in [0, \Delta]$, $\phi \in [0, 2\pi]$, succ соотвественно будет состоянием, которое расположено на расстоянии r под углом ϕ от оси Ox. Выполняется несколько попыток сгенерировать последующее состояние, если не смогли набрать нужное число последователей — запускается полный перебор всех доступных состояний, из них случайно добираются оставшиеся succ;
2. ограничением является значение $\mathrm{f} - \mathrm{value}$ при достижении которого мы считаем, что найти путь сложно. Изначально это некоторая константа, умноженная на эвристическую оценку длины пути, далее — при очередной неудачной попытке найти путь — лимит увеличивается мультипликативно;
3. AVOID состояния никак не отличаются от обычных;

Отмечу, что за рамками опубликованной работы проведен эксперимент, в котором AVOID состояния не имеют ограничения при запуске WA*. Лучших результатов данная модификация не показала.

4.3 MRA*

Реализован алгоритм из статьи с возможностью задавать все гиперпараметры:

1. список уровней дискретизации (уровень = размер клетки);
2. ω_1 - параметр W* на всех уровнях дискретизации
3. ω_2 - коэффициент субоптимальности

4. алгоритм выбора очереди (Round-Robin, Meta-A* или DTS)

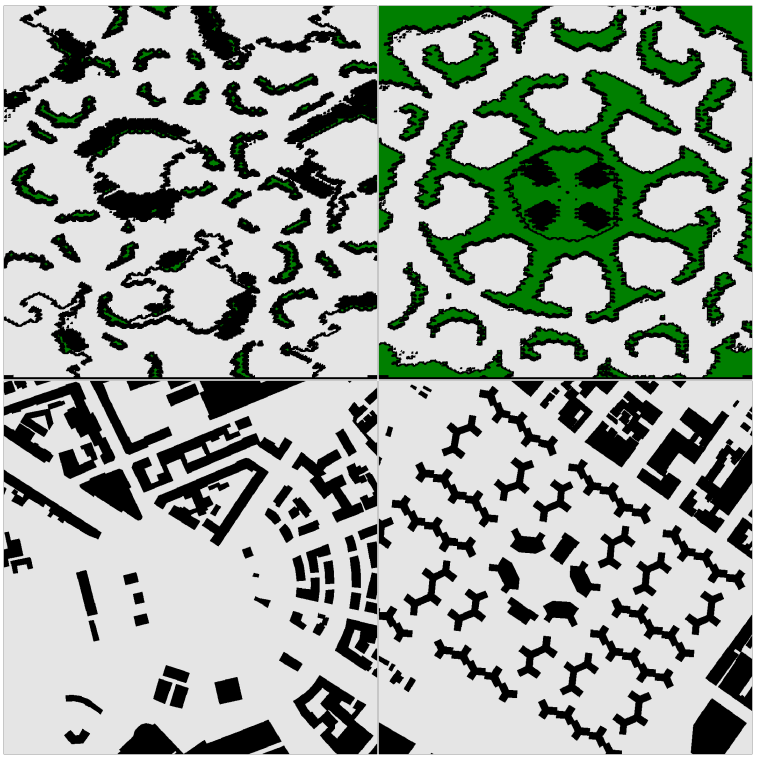
5. для DTS - коэффициент, показывающий, сколько последних раскрытий очереди i влияют на вероятность выбора этой очереди

Судя по всему, авторы статьи предполагали возможность использования только нечетных размеров больших клеток (о чем говорят их иллюстрации и эксперименты), потому что если размер четный, то центр клетки не совпадает с какой-дибо вершиной в графе с максимальным разрешением. Мы отказались от этого ограничения, подвинув решетку так, чтобы центры больших клеток всегда совпадали с центрами маленьких, а границы — необязательно. Решили сделать так, чтобы во всех разрешениях центр левой верхней клетки совпадал с левой верхней клеткой в максимальном разрешении.

5 Эксперименты

5.1 Данные

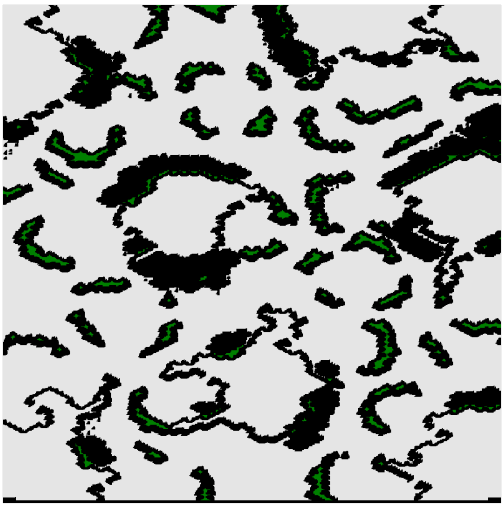
Эксперименты проводились на нескольких картах из датасета MovingAI [1]. Задания взяты из того же датасета, но были прорежены, чтобы на каждую карту было по 200 штук, разного уровня сложности (длина оптимального пути для задания равномерно возрастает от 0 до примерно 1.5 тысяч единиц).

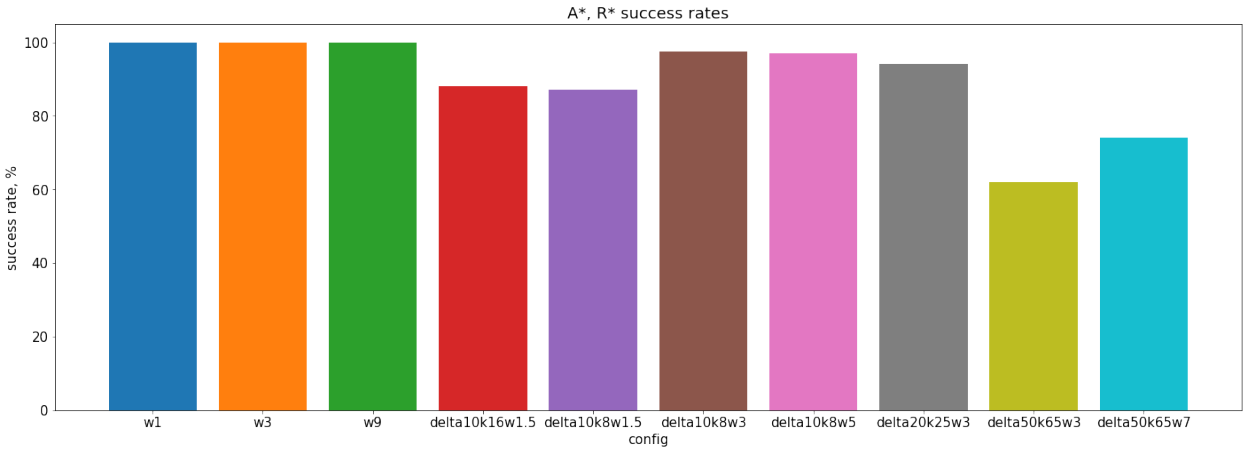


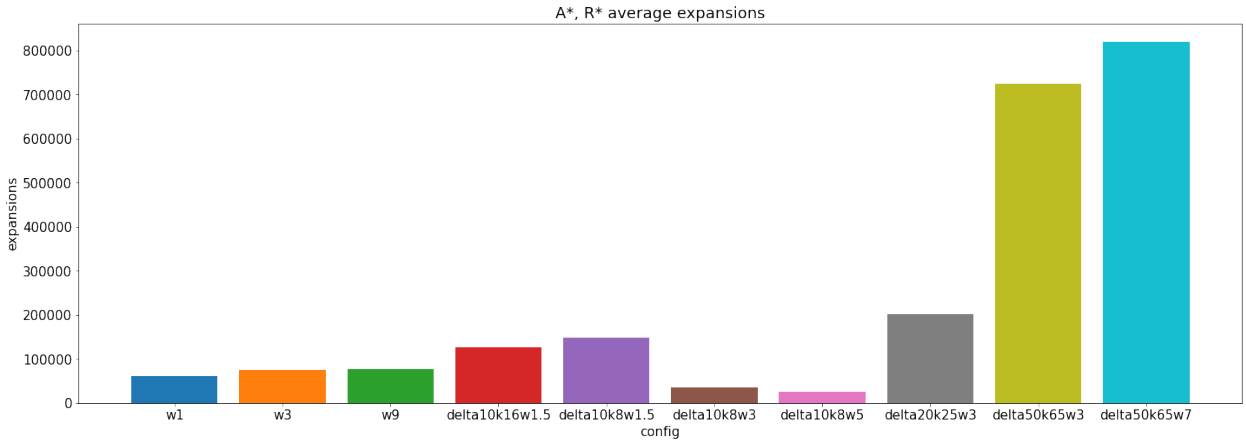
При проведении запусков алгоритмов отсеживались такие параметры как: время работы, потребление памяти, число раскрытых вершин, итоговая длина пути. Время работы алгоритмов было ограничено пятью секундами, поскольку, во-первых, большая тестовая выборка, а во-вторых, WA* укладывался на всех тестах в 3 секунды.

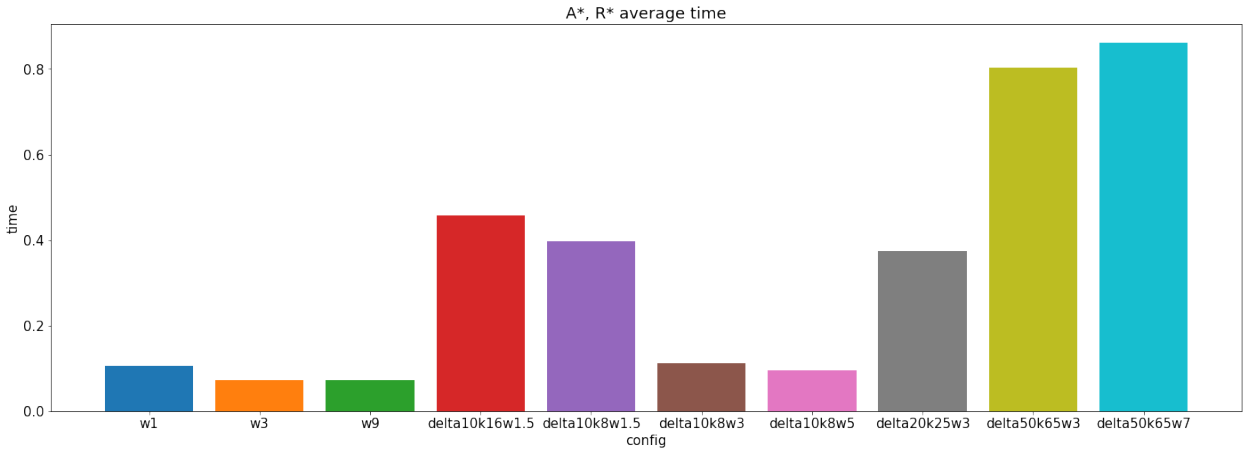
5.2 R*

Рассмотрим подробно результаты работы алгоритма на карте Broken Steppes из набора Starcraft.







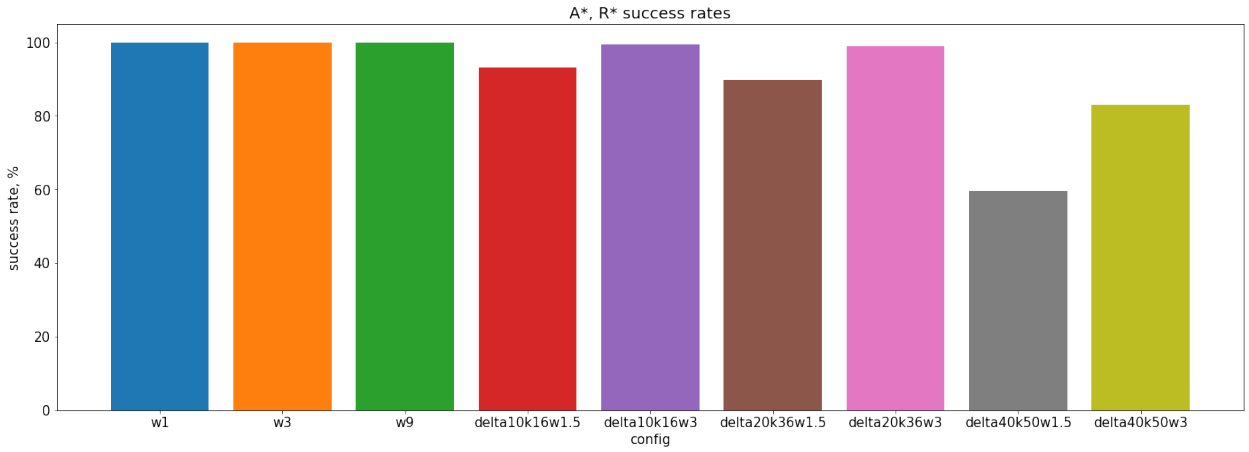


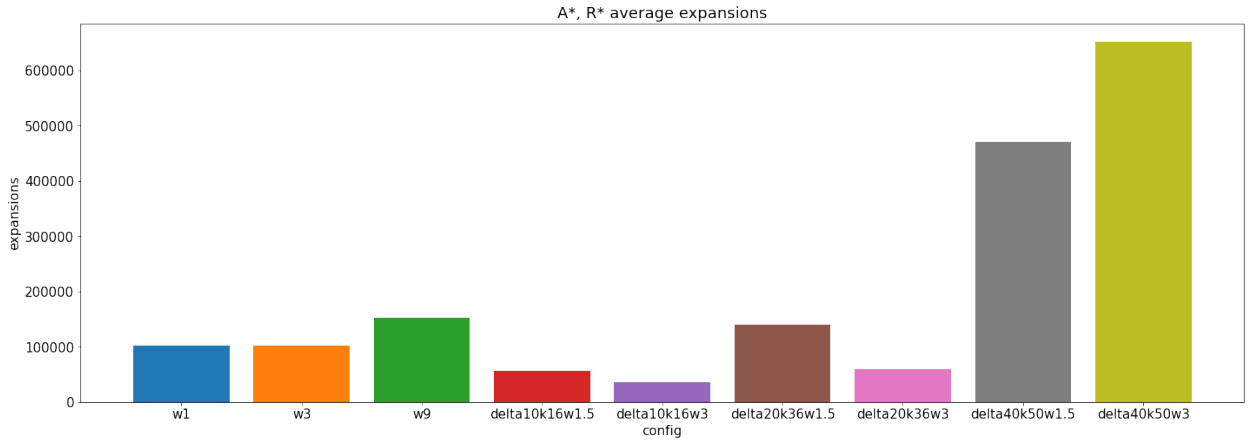
Три первые колонки на графиках — метрики по запускам WA*, остальные — метрики R* на подписанной под колонками параметризации.

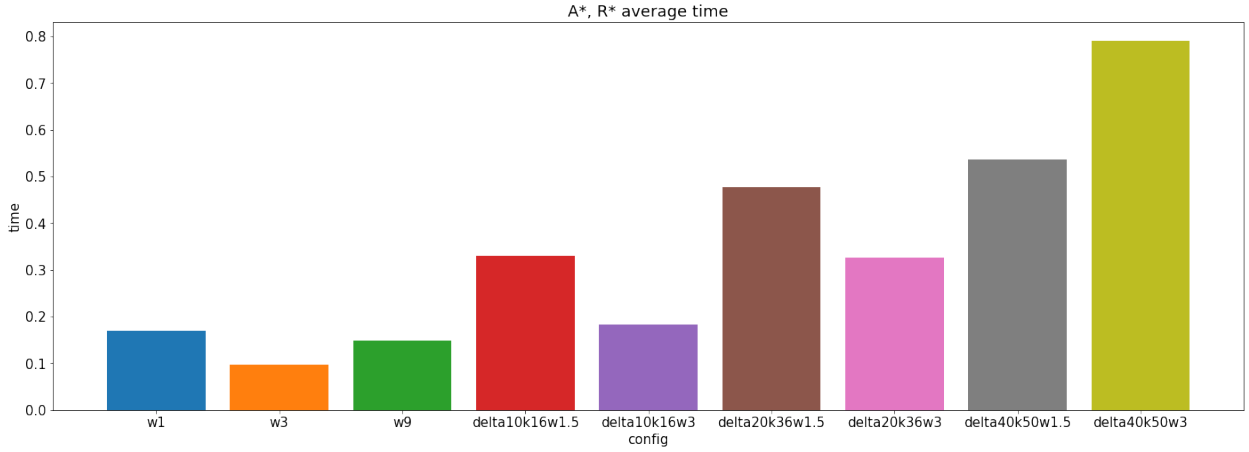
Как можно видеть, результаты работы сильно зависят от выбранной параметризации. На некоторых алгоритм работает хорошо, у WA* выигрывая значительно по числу раскрытых вершин, на некоторых ведет себя намного хуже, иногда даже не успевая уложиться в ограничение по времени. Также видно, что увеличение веса для внутреннего WA* сильно улучшает производительность алгоритма. При этом по времени работы алгоритм всё же показывает результат хуже, чем WA*.

У нас не получилось найти хорошую параметризацию с большим Δ, алгоритм хорошо работает только для небольших радиусов (порядка 10-15 единиц).

Ниже результаты работы алгоритма на карте Cities, Milan.





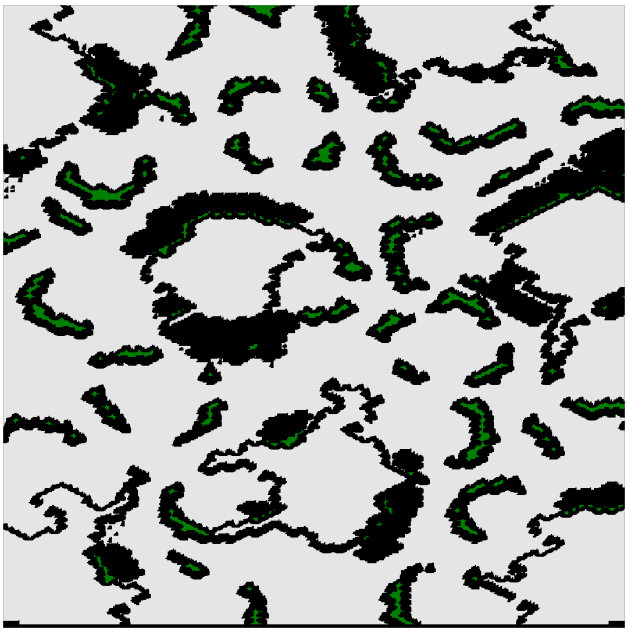


В этом случае выводы можно сделать примерно такие же, как и для прошлой карты.

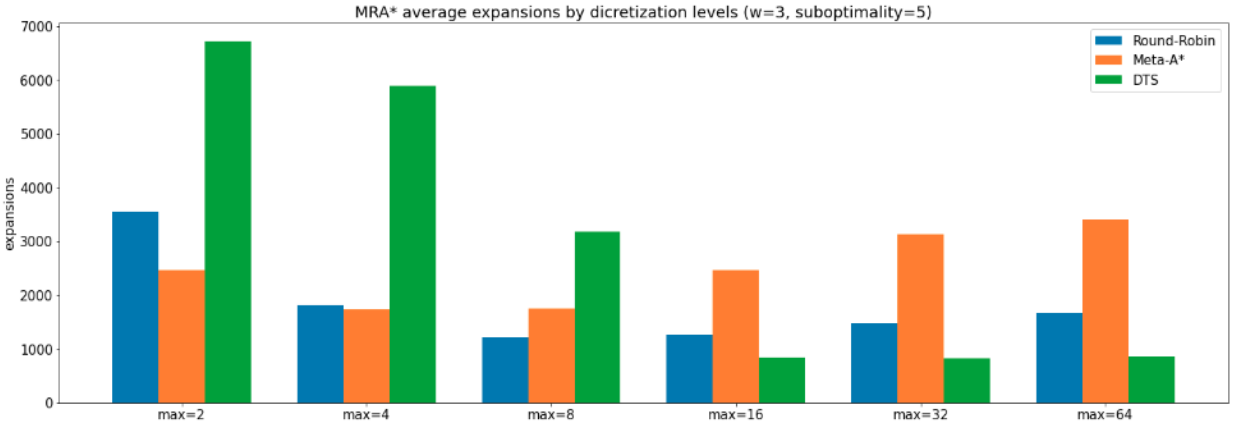
5.3 MRA*

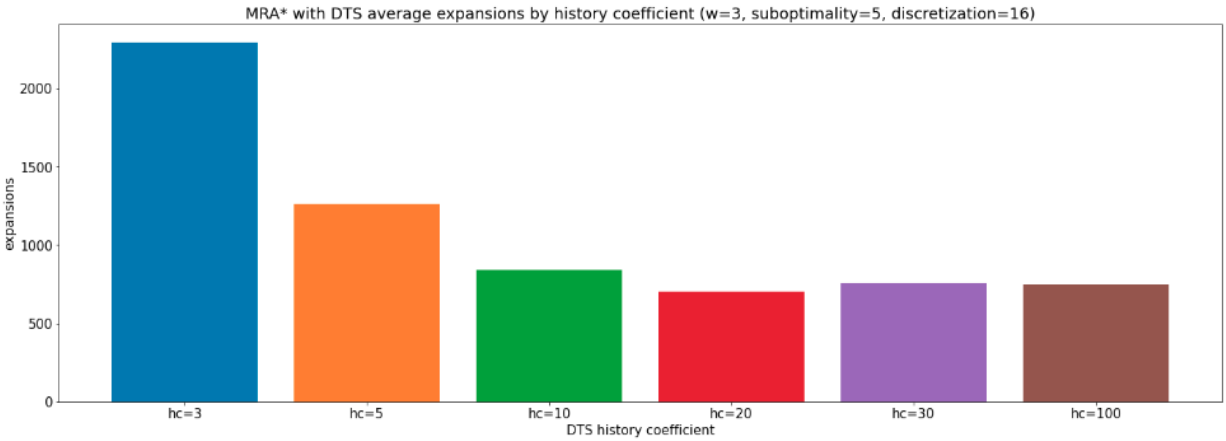
Мы проанализировали влияние выбора мета-алгоритма на эффективность алгоритма на нескольких картах. В качестве уровней дискретизации выбирали первые несколько степеней двойки.

Поскольку результаты на разных картах были похожими, рассмотрим в качестве примера только одну.



BrokenSteppes





Выводы:

- Round-Robin быстро перестает получать выгоду от увеличения количества уровней дискретизации

- Meta-A* слишком сильно полагается на эвристику, поэтому отдает предпочтение уровням с самой крупной сеткой даже если это невыгодно, поэтому с увеличением количества уровней дискретизации показывает худшие результаты
- DTS работает намного хуже остальных алгоритмов, если нет возможности совершать большие прыжки, но при увеличении количества уровней дискретизации начинает работать значительно лучше
- Во всех случаях после достижения максимума результаты ухудшаются

5.4 Сравнение алгоритмов

Ниже приведены таблицы по запускам алгоритмов на оптимальных конфигурациях (с весом $w = 3$).

Broken steppes

	Success rate (%)	Average expansions	Average time (s)	Average path length
WA*	100.00	72113.27	0.06	640.36
R*	97.50	35162.99	0.09	742.99
MRA*	100.00	1876.09	0.01	678.66

Wheel of war

	Success rate (%)	Average expansions	Average time (s)	Average path length
WA*	100.00	407225.80	0.32	608.73
R*	96.04	90111.96	0.38	674.29
MRA*	100.00	9324.33	0.05	629.62

Milan

	Success rate (%)	Average expansions	Average time (s)	Average path length
WA*	100.00	100822.33	0.09	766.89
R*	99.51	35196.92	0.15	835.19
MRA*	100.00	1860.91	0.01	796.26

New York

	Success rate (%)	Average expansions	Average time (s)	Average path length
WA*	100.00	80906.25	0.07	809.33
R*	100.00	30316.68	0.13	879.97
MRA*	100.00	1475.51	0.01	840.62

Из этих данных видно, что MRA* значительно превосходит WA* по всем метрикам, при этом по длине пути уступает совсем незначительно.

R* хоть и совершает меньше раскрытий вершин, но всё же работает медленнее реализации WA*. По длине пути этот алгоритм уступает уже достаточно значительно.

На последок стоит отметить, что, судя по статье, этот алгоритм разрабатывался во многом для continuous-space случая задачи поиска пути, так что, может быть, эти результаты и не должны сильно удивлять.

Список литературы

[1] N. Sturtevant, "Benchmarks for Grid-Based Pathfinding," 2, т. 4, 2012, с. 144—148. url: http://web.cs.du.edu/~sturtevant/papers/benchmarks.pdf.

[2] A. S. Maxim Likhachev, "R*Search," в *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008, с. 344—350. url: https://www.aaai.org/Papers/AAAI/2008/AAAI08-054.pdf.

[3] F. I. Wei Du и M. Likhachev, "Multi-Resolution A*," в *The Thirteenth International Symposium on Combinatorial Search*, 2020, с. 29—37. url: https://aaai.org/ocs/index.php/SOCS/SOCS20/paper/viewFile/18515/17554.

[4] M. Phillips и др., "Efficient Search with an Ensemble of Heuristics," в *International Joint Conferences on Artificial Intelligence*, 2015, с. 784—791. url: https://www.cs.cmu.edu/~maxim/files/searchwithmanyheuristics_ijcai15.pdf.