|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт Кибернетики

Кафедра проблем управления

**ЛАБОРАТОРНАЯ РАБОТА**

по дисциплине: **УИР и РТС**

**Тема:**

**«Сравнение алгоритмов планирования пути A-Star и Дейкстры**»

|  |  |
| --- | --- |
| **Студент группы:** КРБО-01-16 | Богословский А. М. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Преподаватель:** | Голубов В. В. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

|  |  |
| --- | --- |
| Работа представлена к защите: | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г. |
| Допущен к защите | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

**Москва 2020**

**Цель работы:** изучить особенности и различия алгоритмов планирования пути робота Дейкстры и A-Star

**Задание:** Провести эксперименты с имеющимися программами планирования, изменяя

* Размер робота
* Размер сетки карты
* Карту

Записать выводы о своих наблюдениях.

**Теоретическое введение**

Крайне важной задачей в мобильной робототехнике является планирование пути по заранее известной карте. Задача состоит создании оптимального пути на с наименьшим расстоянием. Популярным решениями являются алгоритм Дейкстры и алгоритм A-Star, которые работают с картами через графовое представление.

1. **Алгоритм Дейкстры**

Алгоритм Дейкстры работает на том основании, что любой подпуть B -> D кратчайшего пути A -> D между вершинами A и D также является кратчайшим путем между вершинами B и D. Дейкстра использовал это свойство в противоположном направлении, т.е. мы переоцениваем расстояние каждой вершины от начальной вершины. Затем мы посещаем каждый узел и его соседей, чтобы найти кратчайший подпуть к этим соседям.

Алгоритм использует «жадный» подход в том смысле, что мы находим следующее лучшее решение, надеясь, что конечный результат является лучшим решением для всей задачи.

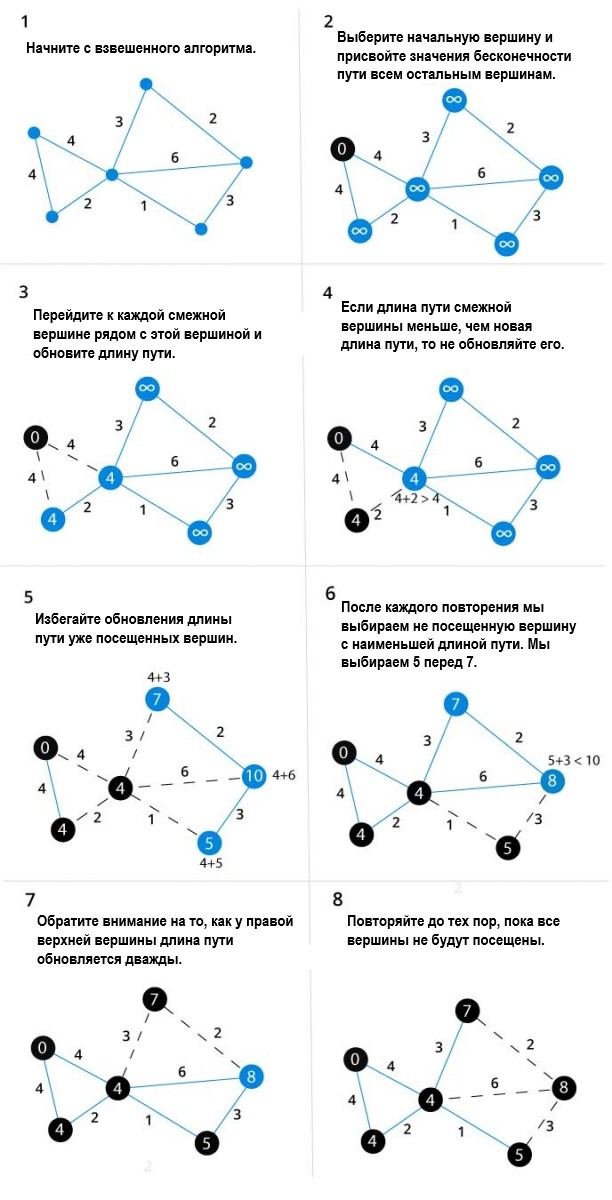


Рис.1. Пример работы алгоритма Дейкстры.

1. **Алгоритм A-Star (A\*)**

A\* пошагово просматривает все пути, ведущие от начальной вершины в конечную, пока не найдёт минимальный. Как и все [информированные алгоритмы поиска](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA), он просматривает сначала те маршруты, которые «кажутся» ведущими к цели. От [жадного алгоритма](https://ru.wikipedia.org/wiki/%D0%96%D0%B0%D0%B4%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), который тоже является алгоритмом поиска по первому лучшему совпадению, его отличает то, что при выборе вершины он учитывает, помимо прочего, *весь* пройденный до неё путь. Составляющая *g(x)* — это стоимость пути от начальной вершины, а не от предыдущей, как в жадном алгоритме. Составляющая *h(x)* - функции эвристической оценки расстояния от рассматриваемой вершины к конечной. Функция *h(x)* не должна переоценивать расстояния к целевой вершине. Например, для задачи маршрутизации *h(x)* может представлять собой расстояние до цели по прямой линии, так как это физически наименьшее возможное расстояние между двумя точками. Если *h(x)* удовлетворяет условию ***h*(*x*) ≤ *d*(*x*, *y*) + *h*(*y*)** для каждого ребра *(x, y)* графа (где *d(x, y)* - длина ребра), тогда *h(x)* называется монотонной. С такой эвристикой, алгоритм гарантированно найдет оптимальный путь с единоразовой обработкой каждого узла.

**Примеры эвристик:**

Если мы можем перемещаться в четырех направлениях, то в качестве эвристики стоит выбрать манхэттенское расстояние

Расстояние Чебышева применяется, когда к четырем направлениям добавляются диагонали:

Если передвижение не ограничено сеткой, то можно использовать евклидово расстояние по прямой:

В начале работы просматриваются узлы, смежные с начальным; выбирается тот из них, который имеет минимальное значение *f(x)*, после чего этот узел раскрывается. На каждом этапе алгоритм оперирует с множеством путей из начальной точки до всех ещё не раскрытых (листовых) вершин графа — множеством частных решений, — которое размещается в [очереди с приоритетом](https://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_%D1%81_%D0%BF%D1%80%D0%B8%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D1%82%D0%BE%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Приоритет пути определяется по значению *f(x) = g(x) + h(x)*. Алгоритм продолжает свою работу до тех пор, пока значение *f(x)* целевой вершины не окажется меньшим, чем любое значение в очереди, либо пока всё дерево не будет просмотрено. Из множества решений выбирается решение с наименьшей стоимостью.

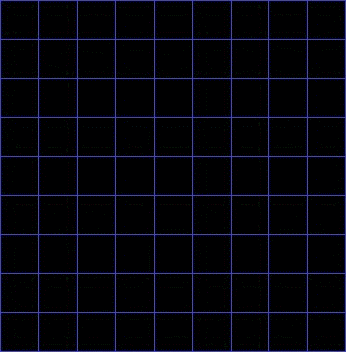
****

Рис. 2. Пример работы алгоритма A-Star

**Главным отличием** **алгоритмов** является направление поиска. Алгоритм A-Star осуществляет поиск, направленный к цели, а алгоритм Дейкстры осуществляет поиск во все стороны.

**Ход работы:**

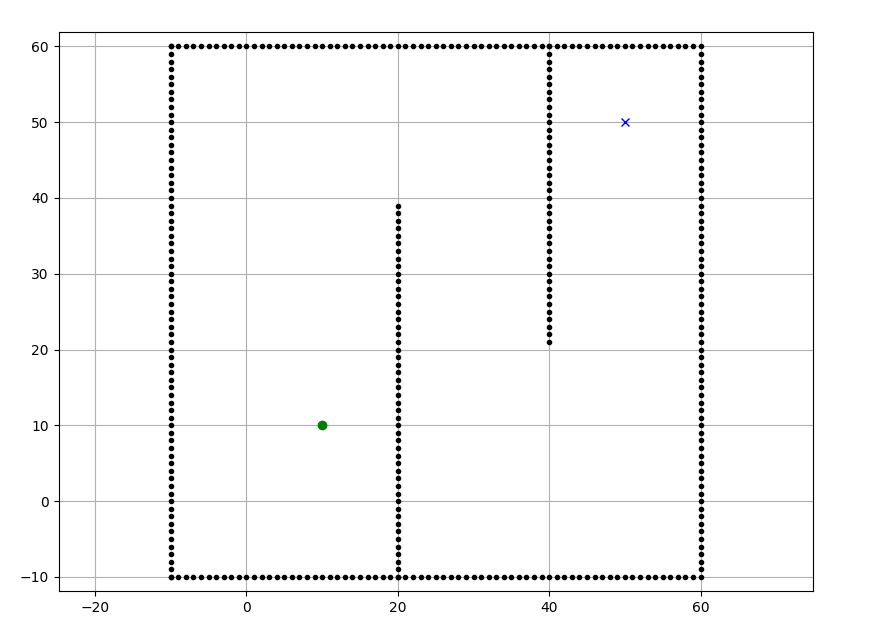


Рис 3. Стандартная карта. Зеленая точка - старт. Синий крест - цель.

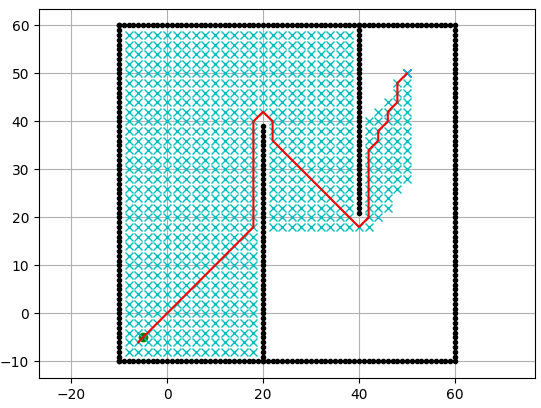


Рис 4. A-Star без изменений. Время: 20,7 с.

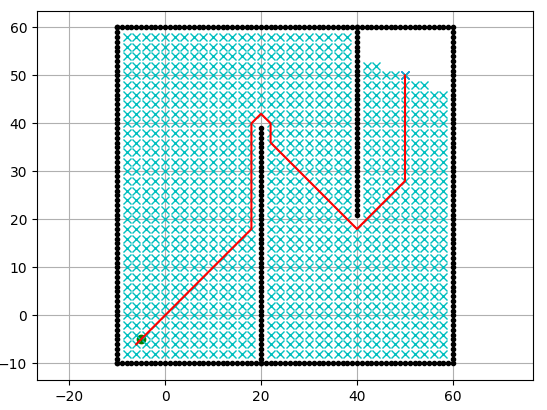


Рис 5. Dijkstra без изменений. Время: 30,7 с.

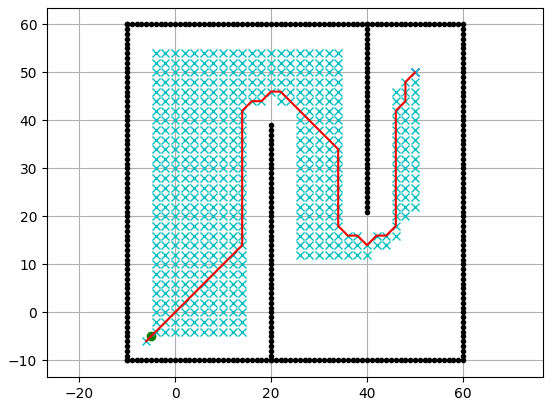


Рис 6. A-Star с радиусом робота = 5. Время: 12,3 с.

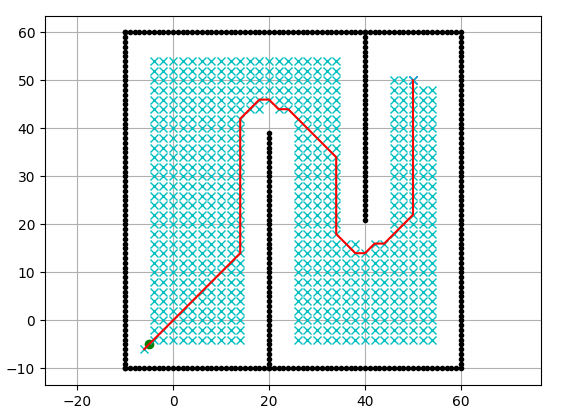


Рис 7. Dijkstra с радиусом робота = 5 м. Время: 16,6 с.

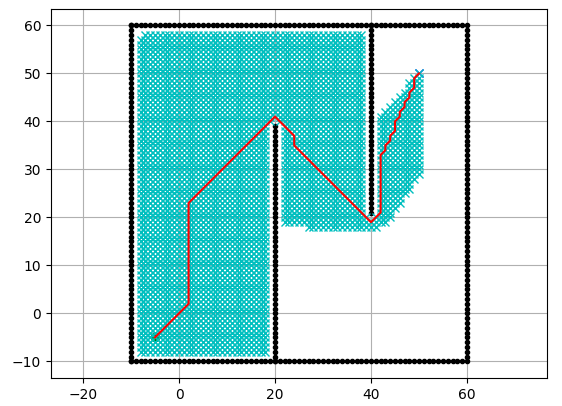


Рис 8. A-Star с размером сетки = 1 м. Время: 109,5 с.

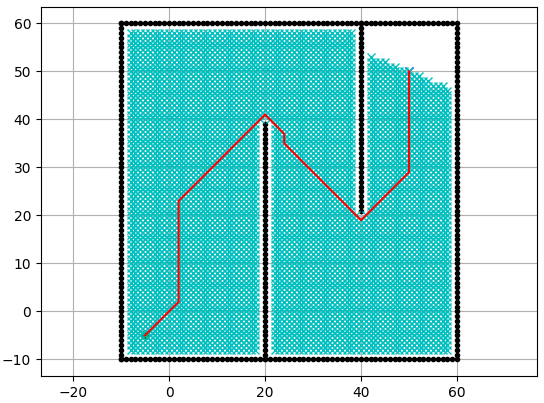


Рис 9. Dijkstra с размером сетки = 1 м. Время: 180,3 с.

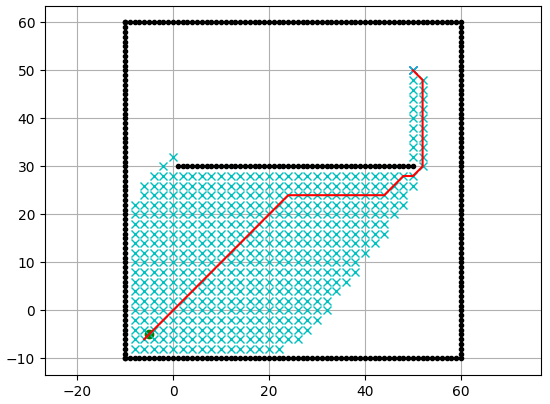


Рис 10. A-Star с другой картой. Время: 11,6 с.

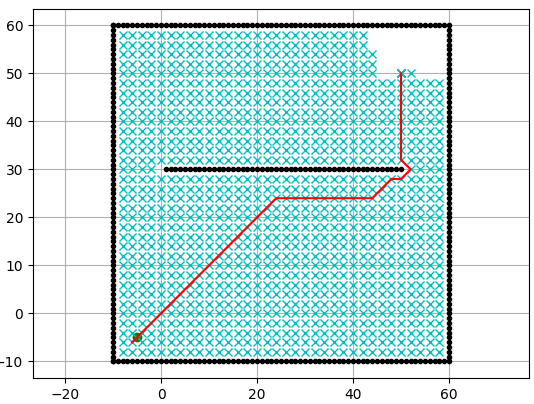


Рис 11. Dijkstra с другой картой. Время: 33 с.

Таблица 1 - Результаты проведения симуляций.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Изменения** | **Время для A-Star, с** | **Время для Dijkstra, с** |
| 1 | Без изменений | *20.7* | *30.7* |
| 2 | Размер робота = 5 м | *12.3* | *16.6* |
| 3 | Размер сетки = 1 м | *109.5* | *180.3* |
| 4 | Вид карты | *11,6* | *33* |

**Вывод:** алгоритм A-Star показал себя намного лучше, чем алгоритм Дейкстры. Это и не удивительно, тк A-Star является логическим развитием алгоритма Дейкстры, дополнительно учитывая направление к цели. Стоит указать, что чем меньше вариабельность проверки соседних узлов (уже коридор), тем меньше разница во времени работы между двумя алгоритмами. Если же карта очень большая, то предпочтение стоит отдавать A-Star. Исключение - очень сложные карты-лабиринты, где движение по направлению к цели не гарантирует, что до нее можно будет добраться напрямую.