ПЛАНИРОВАНИЕ ТРАЕКТОРИИ НА ПЛОСКОСТИ С УЧЕТОМ РАЗМЕРА АГЕНТА (МОБИЛЬНОГО РОБОТА, БЕСПИЛОТНОГО ТРАНСПОРТНОГО СРЕДСТВА)

А.А. Андрейчук (*andreychuk@mail.com*)
РУДН, Москва

К.С. Яковлев (*yakovlev@isa.ru*)
ФИЦ ИУ РАН, ВШЭ, Москва

Аннотация. В статье задача планирования траектории на плоскости рассматривается как задача поиска пути на графе особой структуры. Предполагается, что граф является моделью окружающей среды для интеллектуального агента (мобильного робота, беспилотного транспортного средства), который моделируется диском определенного радиуса и может перемещаться на плоскости в произвольном направлении. Предлагаются методы, применение которых позволяет известным алгоритмам эвристического поиска пути на графе учитывать размер агента, проводятся их экспериментальное исследование.[1]

Ключевые слова: планирование траектории, эвристический поиск, A*, Theta*, учет размера агента.

Введение

Планирование траектории – одна из ключевых задач неизбежно возникающих при разработке систем управления беспилотными транспортными средствами, мобильными роботами и т.д. [Макаров и др., 2015]. Известно достаточно много постановок этой задачи и подходов к её решению. Один из наиболее распространенных – представление задачи планирования как задачи поиска пути на графе, вершинам которого соответствуют положения, которые объект управления (в терминологии искусственного интеллекта – интеллектуальный агент) может занимать в пространстве, а ребрам – элементарные траектории между положениями (отрезки прямых или сегменты кривых определенной формы). Обзор графовых моделей, применимых для решения задач планирования приведен в [Яковлев и др., 2013], методов поиска на графах в [Казаков и др.], способов учета различных критериев оптимальности решения в

[1] Работа выполнена при финансовой поддержке РФФИ (проект № 17-07-00281).

[Лавренов и др., 2016], результатов применения известных методов для модельных и реальных задач управления беспилотным транспортом в [Афанасьев и др., 2015; Жулев и др., 2013].

В данной работе рассматривается статическая задача планирования траектории на плоскости, а в качестве модели окружающего агента пространства поиска используется граф регулярной декомпозиции, метрический-топологический граф (МТ-граф), известный в англ. терминологии как grid [Yap, 2002]. Зачастую при решении задачи планирования траектории с использованием МТ-графа пренебрегают размером и формой агента – см., например, [Андрейчук и др., 2016]. При этом, если граф является только моделью среды и не учитывает свойств агента, то необходима его дополнительная обработка для учета ограничений, накладываемых размером и формой агента.

В робототехнике для учета этих ограничений применяется метод расширения непроходимых областей на величину, соотносящуюся с размером агента [Choset et al., 2005] (см. рис. 1), т.н. преобразование рабочего пространства (workspace) в пространство конфигураций (configuration space или C-Space). Таким образом, при планировании агент может считаться материальной точкой, и найденные пути при этом будут осуществимыми. Однако такой подход не применим для ряда задач, например, для задачи многоагентного планирования, когда имеется множество агентов, и необходимо найти совокупность неконфликтных траекторий для этих агентов. Очевидно, что для определения конфликтов между агентами их нельзя считать лишь материальными точками.

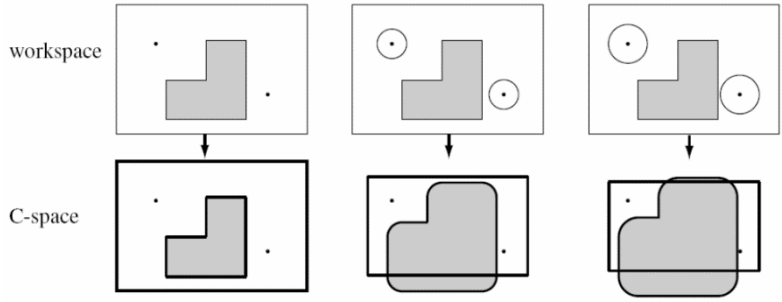

Рис. 1. Учет размера агента за счет расширения непроходимых областей

В данной работе предлагается учитывать ограничения, накладываемые размером агента, в процессе решения задачи планирования. Для этого предлагаются методы, которые необходимо реализовать как составные части известных алгоритмов эвристического поиска, таких как A* [Hart et al., 1968], Theta* [Daniel et al., 2010] и др. Предполагается, что агент

представлен открытым диском определенного радиуса, т.к. во многих практических задачах, связанных с автоматизацией управления беспилотных транспортных средств, это предположение уместно (например, когда речь идёт и мультироторных летательных аппаратах или малых колесных роботах).

1. Постановка задачи

Агент моделируется открытом кругом радиуса $r > 0$ и функционирует внутри прямоугольной области на плоскости, которая разбита на квадратные проходимые и непроходимые ячейки, образуя тем самым MT-граф [Yap, 2002] (см. рис. 2). Размер каждой клетки равен l. Без ограничения общности будем считать, что $l = 1$, а X/Y координаты центров вершин имеют целые значения: 1, 2, 3 и т.д. Начало осей координат находится в нижнем левом углу (см. рис. 2). Положения, в которых может находиться агент, привязаны к центрам проходимых вершин. Более того, агент может находиться в вершине $U(x, y)$, только если открытый диск радиуса r с центром в (x, y) не задевают какую-либо непроходимую область. Здесь и далее будем называть такие положения (вершины) допустимыми.

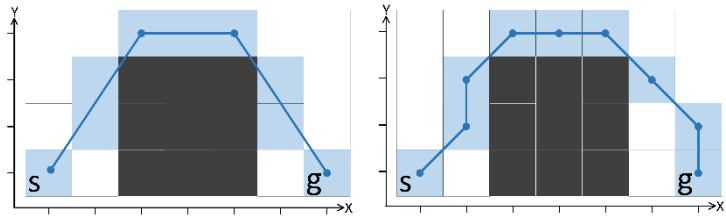

Рис. 2. Пути, построенные алгоритмами Theta* и A* для точечного агента. Оба пути являются неосуществимыми для агентов с радиусом r > 0. Подсвеченные вершины – это вершины, которые идентифицировал алгоритм Брезенхема

Агент может перемещаться вдоль отрезков, которые соединяют центры двух проходимых вершин. При этом движение является допустимым, если открытый диск радиуса r, двигаясь вдоль секции, не задевает какие-либо непроходимые клетки. Путь между начальным положением (*start*) и целевым (*goal*) – это последовательность смежных допустимых секций. Задача планировщика состоит в отыскании пути из начального положения в целевое. Более короткие пути являются предпочтительными, хотя длина пути не подвержена строгим ограничениям (задача поиска минимального по длине пути не ставится). В качестве базового алгоритма поиска используется Theta* (модификация известного алгоритма A*) – см. рис. 2.

2. Планирование с учетом размера агента

2.1 Theta*

Theta* – это эвристический алгоритм поиска, применимый для планирования на МТ-графах и учитывающий возможность перемещения агента в произвольном направлении. Принцип его работы аналогичен алгоритму A*. Он использует ту же стратегию поиска в пространстве состояний и оперирует списком вершин-кандидатов на дальнейшее рассмотрение, *OPEN*. На каждом шаге алгоритм выбирает наиболее перспективную вершину из списка и раскрывает её. Под раскрытием подразумевается процесс генерации потомков и расчет их g-значений (оценок длины пути от начальной вершины). Разница между Theta* и A* заключается в том, что при обновлении g-значения потомка s', при раскрытии вершины s, Theta* также проверяет достижимость s' из родителя s, т.е. *parent*(s). Если перемещение из *parent*(s) в s' является допустимым, то g-значение s' рассчитывается на основе g-значения *parent*(s), иначе оно рассчитывается на основании g-значения состояния s. Более подробно принцип работы алгоритма описан в [Daniel et al., 2010].

Стоит отметить, что мы различаем вершины МТ-графа $\{C(x, y)\}$ и состояния $\{s\}$, хотя между ними есть взаимно-однозначное соответствие. Причина этого различия заключается в том, что в используемой реализации алгоритма Theta* используются отдельные структуры данных для хранения вершин. Каждое состояние, таким образом, представляет собой кортеж s = [*cell*(s), g(s), *parent*(s)]. В качестве эвристической функции используется Евклидова метрика.

2.2 Учет размера агента

2.2.1 Оценка допустимости состояния

Рассмотрим некоторую вершину МТ-графа $C(x, y)$, соответствующую преемнику s', который находится на рассмотрении на текущем этапе алгоритма. Задача состоит в том, чтобы оценить, может ли агент радиусом r занимать положение (x, y). В случае $0 < r \leq l/2$ (=1/2) достаточно проверить проходимость только вершины C. Если же радиус агента превышает величину $l/2$, требуется более сложная процедура проверки. Сначала строится квадрат, обрамляющий открытый диск радиусом r, то есть определяются координаты вершин, образующих квадрат. Длина стороны квадрата не превышает $2\cdot\lfloor r + l/2 \rfloor + 1$. Затем перебираются ячейки, образующие квадрат, и вычисляется расстояние между центром квадрата (x, y) и ближайшей к (x, y) точке, лежащей на границе этой клетки. Такой точкой является либо угол, либо середина стороны, ближайшей к (x, y) (см. рис. 3). Если расстояние меньше, чем r, то вершину необходимо проверить

на проходимость. Если все проверенные вершины являются проходимыми, то положение (x, y) является допустимым.

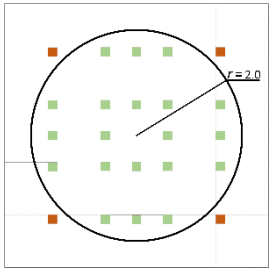

Рис. 3. Оценка допустимости положения агента радиусом r = 2,0. Вершины, отмеченные зелеными метками, должны быть проверены на проходимость. Положение меток указывает на ближайшие к центру круга точки на границах клеток МТ-графа

2.2.2 Оценка допустимости элементарного перемещения

Элементарное перемещение – это переход из проходимой вершины в одну из смежных проходимых вершин. В случае, когда агент переходит в ортогонально смежную вершину, ход является допустимым, если обе вершины являются допустимыми. Если же происходит переход по диагонали, необходимо убедиться, что агент не задевает угол препятствия. Если $0 < r \leq l/2$, то необходимо проверить проходимость двух смежных вершин по направлению движения. В случае, когда $r > l/2$, агент также может задевать дополнительные клетки и проверка должна осуществляться так, как изложено ниже.

2.2.3 Оценка осуществимости неэлементарного перемещения

Неэлементарным перемещением является переход из одной вершины в другую, несмежную с ней. Процедура проверки осуществимости неэлементарного перемещения в литературе обычно называется проверкой линии видимости (в англ. *line-of-sight*) и как правило основывается на использовании алгоритма Брезенхема [Bresenham, 1965], наиболее известного в компьютерной графике алгоритма растеризации прямых линий. На вход алгоритму подаются две пары целых чисел, соответствующие координатам концов отрезка. На выходе получается набор точек с целочисленными координатами (вершин МТ-графа), которые являются ближайшими к этому отрезку. Следует заметить, что использование оригинального алгоритма Брезенхема, неприменимо к рассматриваемой задаче даже в случае, если агент имеет нулевой радиус, т.к. алгоритм не идентифицирует *все* вершины, которые пересекает отрезок, а лишь часть из них (см. рис. 2 слева). В данной работе предлагается

алгоритм проверки линии видимости, применимый для агентов любого неотрицательного радиуса.

Без ограничения общности будем считать, что агент движется слева направо, снизу вверх из клетки $A(x_A, y_A)$ в клетку $B(x_B, y_B)$. Аналогично алгоритму Брезенхема, мы движемся слева направо и обрабатываем каждый столбец МТ-графа. В отличие от алгоритма Брезенхема, который отмечает в каждом столбце только одну вершину (ближайшую к $<A, B>$), предлагаемый алгоритм определяет все клетки, которых задевает агент в этом столбце, двигаясь вдоль секции $<A, B>$. Далее подробно опишем процесс определения задеваемых клеток, которые находятся над $<A, B>$. Определение задеваемых клеток, которые находятся под секцией, осуществляется аналогично.

Рассмотрим ситуацию, когда агент проходит через столбец i, а ближайшей вершиной, определенной алгоритмом Брезенхема, является $U(i, j)$. На каждом шаге алгоритм вычисляет значение ошибки $error(i)$, которая фактически является разницей между j и реальным значением секции $<A, B>$ вдоль оси Y, в точке с той же самой X-координатой, то есть i. Далее рассмотрим величину $offset(i) = error(i) + \Delta_y/2\Delta_x + |AB'|$ (см. рис. 4), где $\Delta_x=|x_B - x_A|$, $\Delta_y = |y_B - y_A|$. Очевидно, что точка с координатами $(i+l/2, j+offset)$ – это самая верхняя точка в столбце i, которую задевает агент. Величина AB' эквивалентна значению $r\cdot\mathrm{sqrt}(\Delta_x^2 + \Delta_y^2)/\Delta_x$, т.к. треугольники ABC и $AB'C'$ являются подобными. Зная $offset(i)$, можно вычислить число клеток над $U(i, j)$, которые необходимо проверить на проходимость: $N_{u\text{-}hit}(i) = \lfloor offset(i) + l/2 - \varepsilon \rfloor$, где $\lfloor\cdot\rfloor$ - округление до целого вниз, а добавление $\varepsilon << l/2$ необходимо для корректного округления в случаях, когда тело агента находится строго на границе между клетками.

Начало отрезка A является особым случаем. В общем случае, агент может задевать дополнительные вершины даже в тех столбцах, которые предшествуют ряду x_A. Число таких столбцов, k, зависит от величины $C'D$, которая равна $r\cdot\Delta_y/\mathrm{sqrt}(\Delta_x^2 + \Delta_y^2)$, т.к. треугольники ABC и ADC' являются подобными. Таким образом, $k=\lfloor |C'D| + l/2 - \varepsilon \rfloor - 1$, где "-1" необходимо для того, чтобы не учитывать столбец x_A.

Конечное положение B также является особым случаем. Как можно заметить на рис. 4 число вершин, которые задевает агент, когда приближается к конечному положению, меньше чем величина $N_{u\text{-}hit}(i)$. В то же время, очевидно, что вершины в этих столбцах уже были проверены, когда проверялась осуществимость положения (x_B, y_B). Поэтому, их проверять не нужно.

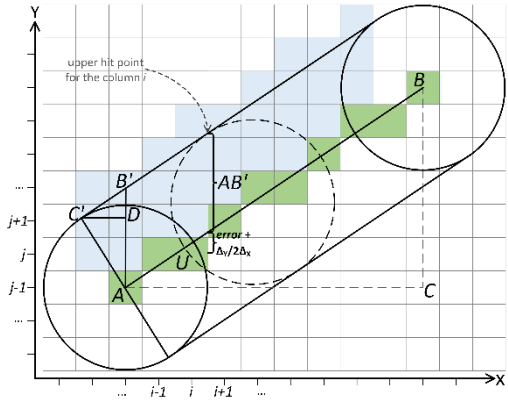

Рис. 4. Определение клеток (вершин МТ-графа), задеваемых агентом во время совершения неэлементарного перемещения. Зеленым отмечены вершины, которые нашел алгоритм Брезенхема. Голубым отмечены вершины, найденные предлагаемым методом

Псевдокод предложенного алгоритма проверки линии видимости представлен на рис. 5. Часть кода, отвечающая за случай, когда $\Delta_y \geq \Delta_x$, опущена, т.к. все операции и проверки являются аналогичными. Для оптимизации кода все операции по возможности были сведены к целочисленным вычислениям.

Algorithm 1: Line Of Sight

Input: s_x, s_y, g_x, g_y, r

$\Delta_x := |s_x - g_x|,\ \Delta_y := |s_y - g_y|,\ error := 0$
$gap := \lfloor r \cdot \sqrt{{\Delta_x}^2 + {\Delta_y}^2} + (\Delta_x + \Delta_y)/2 - \epsilon \rfloor$
if $\Delta_x > \Delta_y$ **then**
  if $s_x > g_x$ **then**
    $swap(s_x, g_x)$, $swap(s_y, g_y)$
  $step_y := sign(g_y - s_y)$
  if $step_y = 0$ **then**
    $step_y := 1$
  $extra\_check := \lfloor \frac{r \cdot \Delta_y}{\sqrt{({\Delta_x}^2 + {\Delta_y}^2)}} + 1/2 - \epsilon \rfloor$
  for k from 1 to $extra\_check$ **do**
    $error := error + \Delta_y$
    for n from 1 to $\lfloor (gap - error)/\Delta_x \rfloor$ **do**
      if $cell\_is\_obstacle(s_x - k, s_y + n \cdot step_y)$ **then**
        return false
      if $cell\_is\_obstacle(g_x + k, g_y - n \cdot step_y)$ **then**
        return false
  $error := 0,\ y := s_y$
  for x from s_x to g_x **do**
    if $cell\_is\_obstacle(x, y)$ **then**
      return false
    if $x < g_x - extra\_check$ **then**
      for k from 1 to $\lfloor (gap - error)/\Delta_x \rfloor$ **do**
        if $cell\_is\_obstacle(x, y + k \cdot step_y)$ **then**
          return false
    if $x > s_x + extra\_check$ **then**
      for k from 1 to $\lfloor (gap + error)/\Delta_x \rfloor$ **do**
        if $cell\_is\_obstacle(x, y - k \cdot step_y)$ **then**
          return false
    $error := error + \Delta_y$
    if $2 \cdot error > \Delta_x$ **then**
      $y := y + step_y$
      $error := error - \Delta_x$
else
  the same, but all checks and operations
  should be swapped between X- and Y-axis
return true

Рис. 5. Псевдокод предлагаемого алгоритма проверки линии видимости

3. Экспериментальные исследования

Для проведения экспериментальных исследований был реализован алгоритм Theta* (на языке C++), содержащий описанные выше проверки. Эксперименты проводились на персональном компьютере (AMD FX-8350 (4.0 GHz), 16 Gb RAM) под управлением ОС Windows-8.1.

В качестве входных данных использовались 2 карты из открытой коллекции MovingAI [Sturtevant, 2012] размером 512 x 512 клеток. Первая карта, *AR0700SR*, представляет собой открытую местность, разделенную на несколько областей, соединенных проходами различной ширины. Вторая карта, *Rooms*, состоит из комнат размером 32 x 32, которые изначально изолированы. Затем добавлены 255 дверей шириной в 5 клеток таким образом, чтобы они соединяли все 256 комнат между собой. После этого дополнительно добавлены 32 двери шириной в 5, 32 двери шириной в 3 и 32 двери шириной в 1 клетку соответственно.

Радиус агента составлял: r = 0,5; 0,7; 1,0; 2,0. Также для сравнения был протестирован точечный агент без тела (r = 0).

Задания генерировались следующим образом. Начальное положение выбиралось случайным образом (недопустимые положения отбрасывались). После этого выбиралось целевое положение, таким образом, чтобы, во-первых, оно было допустимым, во-вторых, путь, находимый A*, имел длину в диапазоне [220; 240] (это необходимо для обоснованного усреднения результатов). Всего было сгенерировано 1500 заданий для карты из *Baldur's Gate* и 3000 заданий для карты *Rooms*.

Результаты исследований представлены в таблицах 1 и 2. При усреднении учитывались только те задания, решения для которых были найдены для агентов всех размеров. Значение числа итераций алгоритма можно также интерпретировать как показатель расхода памяти, т.к. на каждом шаге алгоритм раскрывает одну вершину и сохраняет её в списке до окончания работы. Первый ряд в таблицах, r = 0(Br), относится к алгоритму Theta* с проверкой линии видимости, использующей оригинальный алгоритм Брезенхема, который может находить пути неосуществимые даже для точечного агента.

Таблица 1

Результаты тестирования на карте *AR0700SR*

	Time (ms)	*Path length*	*Steps*	*SR*
r = 0(Br)	1.324	228.95	5 660	100.0%
r = 0	1.817	229.18	5 688	100.0%
r = 0.5	2.009	229.71	5 787	100.0%
r = 0.7	2.109	231.61	5 483	96.1%

r = 1.0	2.249	232.86	5 433	94.8%
r = 2.0	3.794	262.45	7 206	84.3%

Результаты тестирования на карте *AR0700SR* показывают, что для агентов, чей размер превышает одну клетку MT-графа (r > 0,5), решение некоторых заданий может не существовать. Причина этого заключается в том, что начальные и целевые положения в этих заданиях расположены в разных областях, соединенных между собой лишь узкими проходами, через которые не могут пройти агенты большого размера. По той же причине с увеличением размера агента увеличивается и средняя длина пути. Также можно заметить, что число итераций у агентов с радиусом 0,7 и 1,0 меньше, чем итераций у агентов меньших размеров. Объяснить это можно тем, что проходимые вершины смежные с препятствиями являются недопустимыми для агентов большого размера, благодаря чему пространство поиска для них сокращается. С увеличением размера агента время работы закономерно растет, т.к. чем больше агент, тем больше времени требуется для проверки линии видимости.

Таблица 2

Результаты тестирования на карте *Rooms*

	Time (ms)	*Path length*	*Steps*	*SR*
r = 0(Br)	1.408	218.22	6 339	100%
r = 0	1.829	218.73	6 421	100%
r = 0.5	1.939	219.29	6 534	100%
r = 0.7	5.296	285.26	15 616	100%
r = 1.0	5.705	287.25	15 745	100%
r = 2.0	6.519	320.84	16 073	100%

В таблице 2 представлены результаты экспериментов на карте *Rooms*. Аналогично результатам, полученным на карте *AR0700SR*, увеличение размера агента приводит к увеличению времени работы алгоритма, средней длины пути и числу раскрытых вершин. При этом разница на этой карте более существенная. Например, время планирования для агента с r = 2,0 в среднем в 3,4 раза больше, чем для агента с радиусом 0,5. Также у агента самого большого размера длина пути в среднем в 1,5 раза выше, а число требуемых итераций увеличивается на 46%.

Выводы

Результаты экспериментов подтверждают гипотезу о том, что учет размера агента существенно изменяет поведение алгоритма планирования

(Theta* в рассматриваемом случае). Этот факт определенно должен быть принят во внимание при применении хорошо зарекомендовавших себя эвристических методов для решения задач планирования для реальных физических объектов, имеющих определенную форму и размер.

Список литературы

[Андрейчук и др., 2016] Андрейчук А.А., Яковлев К.С. Метод разрешения конфликтов при планировании пространственных траекторий для группы беспилотных летательных аппаратов // Третий Всероссийский научно-практический семинар «Беспилотные транспортные средства с элементами искусственного интеллекта». 2016. С. 31-40.

[Афанасьев и др., 2015] Афанасьев И.М., Сагитов А.Г., Данилов И.Ю., Магид Е.А. Навигация гетерогенной группы роботов (БПЛА и БНР) через лабиринт в 3D симуляторе Gazebo методом вероятностной дорожной карты // Второй Всероссийский научно-практический семинар «Беспилотные транспортные средства с элементами искусственного интеллекта». 2015. С. 18-25.

[Жулев и др., 2013] Жулев В.И., Левушкин В.С., Нгуен Т.Н. Планирование локальной траектории автомобиля-робота в реальном времени // Вестник РГРТУ, 4(46), 2013. С. 18–23.

[Казаков и др.] Казаков К.А., Семенов В.А. Обзор современных методов планирования движения // Труды ИСП РАН, 28(4), 2016. С. 241-294

[Лавренов и др., 2016] Лавренов Р.О., Афанасьев И.М., Магид Е.А. Планирование маршрута для беспилотного наземного робота с учетом множества критериев оптимизации // Третий Всероссийский научно-практический семинар «Беспилотные транспортные средства с элементами искусственного интеллекта». 2016. С. 10-20.

[Макаров и др., 2015] Макаров Д.А., Панов А.И., Яковлев К.С. Архитектура многоуровневой интеллектуальной системы управления беспилотными летательными аппаратами // Искусственный интеллект и принятие решений, 3, 2015. С.18-32.

[Яковлев и др., 2013] Яковлев К.С., Баскин Е.С. Графовые модели в задаче планирования траектории на плоскости // Искусственный интеллект и принятие решений, 1, 2013. С.5-12.

[Bresenham. 1965] Bresenham, J. E. Algorithm for computer control of a digital plotter. *IBM Systems journal*. 1965. № 4(1), P.25-30.

[Choset et al., 2005] Choset, H., Burgard, W., Hutchinson, S., Kantor, G., Kavraki, L. E., Lynch, K., and Thrun, S. Principles of Robot Motion: Theory, Algorithms, and Implementation, MIT Press, April 2005.

[Daniel et al., 2010] Daniel, K., Nash, A., Koenig, S., and Felner, A. Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*. 2010. № 39, P.533-579.

[Hart et al., 1968] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*. 1968. № 4(2), P.100-107.

[Sturtevant, 2012] Sturtevant, N. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*. 2012. №4(2), P.144–148.

[Yap, 2002] Yap, P. Grid-based path-finding // In Proceedings of 15th Conference of the Canadian Society for Computational Studies of Intelligence, Springer: Berlin, Heidelberg. 2002. P.44-5