УДК 007.52-047.44

АНАЛИЗ СУЩЕСТВУЮЩИХ АЛГОРИТМОВ РАБОТЫ БЕСПИЛОТНОГО ТРАНСПОРТА

КУКУШКИН А. В.

Белорусский государственный университет информатики и радиоэлектроники
(г. Минск, Республика Беларусь)

E-mail: 3274533@mail.ru

Аннотация. Представлена разрабатываемая система управления для реального мобильного робота.

Abstract. The developed control system for a real mobile robot is presented.

Мобильных роботов можно классифицировать по признаку использования в рабочей среде:

- наземные или домашние роботы, которые обычно принято называть беспилотные транспортные средства (БТС). Наиболее часто встречающиеся это колесные или гусеничные, а также шагающие (человекоподобные или насекомоподобные);
- транспортные роботы перемещаются только в рабочей области;
- воздушные роботы, или как принята называть – беспилотные летательные аппараты (БЛА);
- подводные работы, или автономные подводные аппараты (АПА);
- полярные роботы, предназначены для перемещения в снегах.

Перемещение в каждой из этих сред имеет свои отличительные характеристики, которые непосредственно зависят от физических свойств среды. Так как в данной работе рассматривается наземные мобильной робот, то рассмотрим способы перемещения наземных мобильных роботов:

- конечности или ноги;
- колесные шасси;
- гусеничные шасси.

Основные свойства мобильного робота, которые следует учитывать при проектировании движений мобильного робота:

- скоростные свойства;
- влияние ускорения;
- надежность;
- уровень потребления энергии [1].

Разрабатываемую систему управления для реального мобильного робота можно представить, как взаимосвязь ряда подсистем с внешней средой и системой управления (рис. 1.).

Система управления движением предназначена для планирования таких программных траекторий движения робота, которые бы приводили робота в заданное целевое состояние в среде с препятствиями, учитывая динамические характеристики робота. Целевое состояние для этой системы формирует система планирования траектории. На выходе данная система формирует требуемое значения скоростей линейного движения.

Система управления исполнительными механизмами предназначена для управления исполнительными механизмами робота. Эта система реализует интерфейс с аппаратной частью робота. На входе имеет сформированный сигнал значений скоростей, полученный от системы управления движением. На выходе получаем изменение положения робота.

Информационно-измерительная система предназначена для сбора, обработки и преобразования сенсорной информации в сигналы, удобные для использования в системе управления робота. Входными данными являются условия внешней среды.

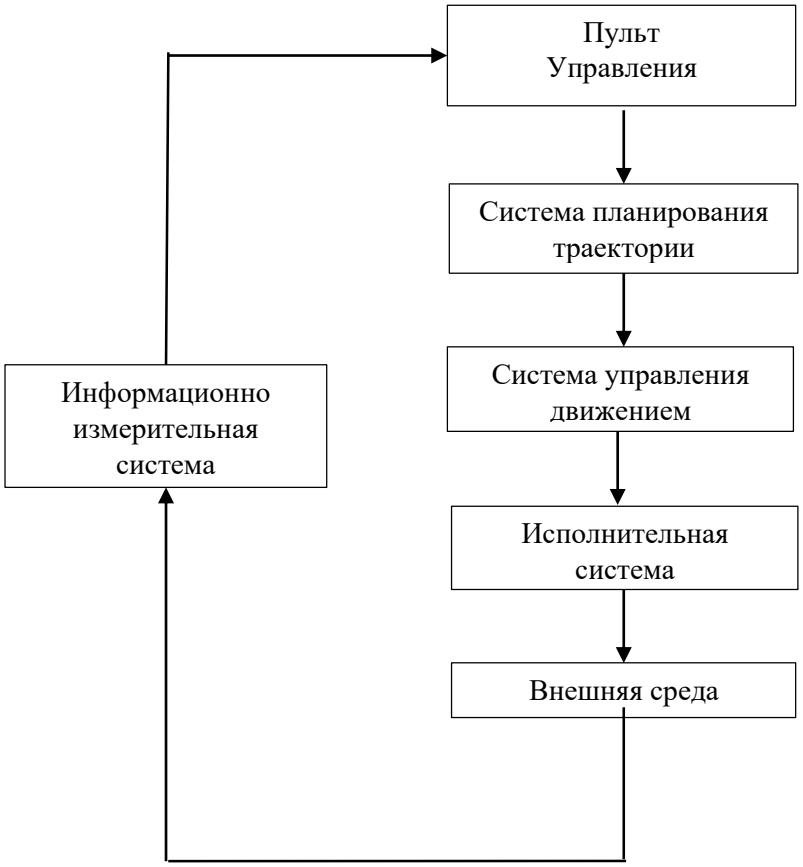

Рис. 1. Структура системы управления мобильного робота

Алгоритмы поиска кратчайшего пути

В рамках одной из основных задач планирования движения выделяют задачу кратчайшего пути – поиск наиболее короткого пути, расположенного между двумя точками (вершинами) на графе, в которой уменьшается сумма весов ребер, составляющих путь.

Кратчайшую цепь (простую) обычно называют геодезическая. Теория графов относит задачу поиска кратчайшего пути к важнейшим классическим задачам. Именно поэтому в настоящее время существует довольно широкий список алгоритмов, направленных на решение данной задачи. Кроме того, данную задачу принято называть поиском минимального пути, а также задачей дилижанса.

Высокий уровень значимости этой задачи достигается благодаря точу, что она имеет широкое практическое применение. Для примера, GPS-навигаторы (рис. 3.), осуществляют поиск пути, расположенного между заданными позициями. Вершинами, в данном случае, являются пересечения дорог, а сами дороги между пересечениями, в свою очередь, являются ребрами данного графа. Для нахождения самого короткого пути необходимо посчитать расстояние от перекрестка до перекрестка, во всех возможных вариантах. Кроме данного примера, задача поиска кратчайшего пути активно применяется в мобильной робототехнике для поиска оптимального маршрута.

Для планирования пути необходимо учитывать текущие положения и занятость пути. Это необходимо для прокладки наиболее оптимального (учитывая необходимые критерии: кратчайший путь или наименьшее время) маршрута к заданной позиции. Вследствие того, что выделяют огромное число вариаций данной задачи, разработан целый ряд специальных алгоритмов для решения задачи поиска кратчайшего пути на графе, которые представлены ниже [2].

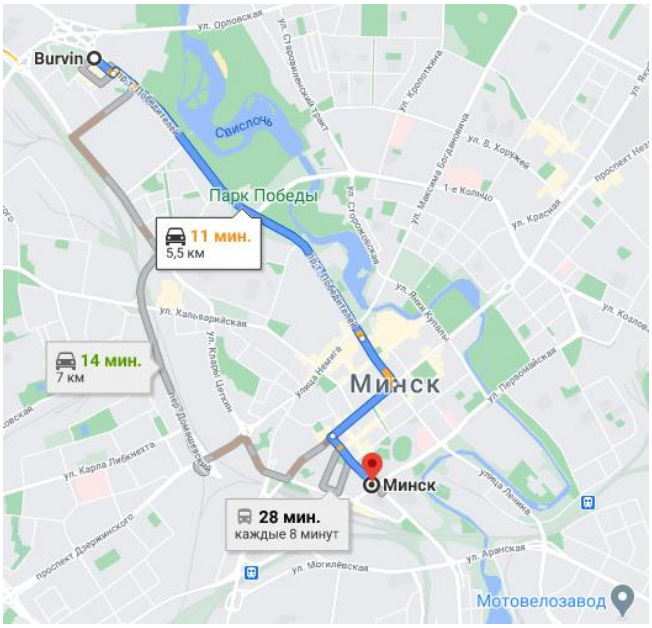

Рис. 2. Поиск кратчайшего пути на примере автомобильного навигатора

Алгоритм Дейкстры (Dijkstra's algorithm)

С использованием данного алгоритма поиска кратчайшего пути, можно найти расстояние из одной вершины графа до существующих других. Необходимо учитывать, что данный алгоритм возможно применить только для тех графов, у которых веса ребер не отрицательные.

Каждая вершина из графа V должна быть сопоставлена определенной метке – минимальным известным расстоянием из определенной вершины до вершины a. Данный алгоритм является пошаговым, то есть при наступлении каждого шага, алгоритм «посещает» определенную из вершин и старается уменьшить метку. Завершается алгоритм в тот момент, когда каждая из вершин посещена.
Определение. Метка начальной вершины a определяется как 0, остальные же вершины приравниваются бесконечности. Таким образом, это отражает тот факт, что расстояние от вершины, а до любой другой неизвестно. Каждая из вершин графа в этот момент определяется как не посещённая.

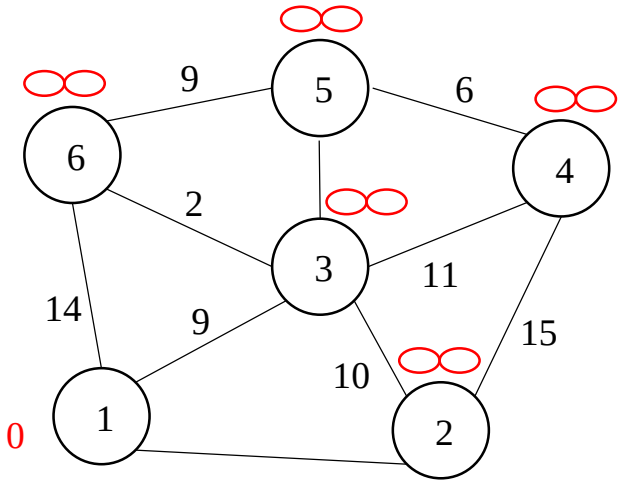

Рис. 3. Этап определения графа алгоритма Дейкстры

Этап алгоритма. Завершение алгоритма происходит в том случае, при котором посещена каждая из вершин. Если же все вершины еще не посещены, то необходимо выбрать такую вершину u, у которой имеется минимальная метка. Необходимо в рамках данного алгоритма рассматривать возможные пути, где вершина u будет предпоследней точкой маршрута. Вершина, в которую приходит ребро из u необходимо назвать соседом данной вершины. У каждой соседской вершины вокруг u,

помимо посещенных, рассматривается новая длина пути, равная сложению текущего значения метки и расстоянию до ребра, которое соединяет u с соседской вершиной. В случае, когда длина меньше величины соседской метки, данная метка изменяет свое значение на полученную длину. Изучив все соседские вершины, вершина u помечается как посещенная и алгоритм повторяется [3].

Алгоритм Беллмана — Форда (Bellman–Ford algorithm)

Данный алгоритм поиска кратчайшего пути позволяет определить путь до вершин взвешенного графа. Ребра могут иметь отрицательный вес, если сравнивать с алгоритмом Дейкстры.

В рамках данного алгоритма используются ориентированные или неориентированные графы (для примера G) с взвешенным значением ребер. Длина пути, в данном случае, является суммой весов каждого ребра, включённого в данный путь. Необходимо отыскать кратчайший путь до каждой вершины графа из определенной точки s.

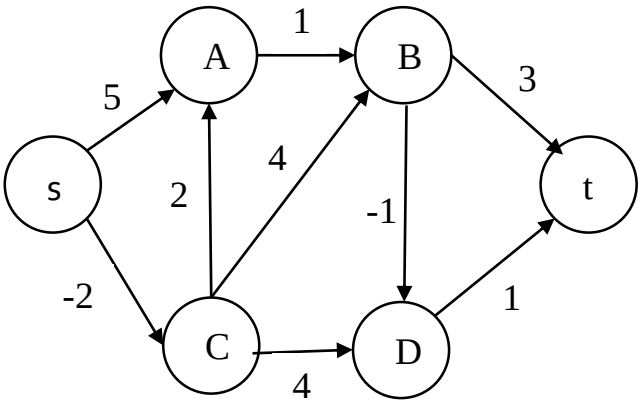

Рис. 4. Граф G алгоритма Алгоритм Беллмана — Форда

Стоит указать, что кратчайшей путь может отсутствовать. Граф, имеющий отрицательный суммарный вес цикла, содержит бесконечно малый путь между вершинами цикла (при каждом обходе циклом уменьшается длина пути). Цикл, сумма весов рёбер которого отрицательна, называется отрицательным циклом [4].

Алгоритм поиска A* (Algorithm A star)

Данный алгоритма поиска позволяет определить стоимость пути от начальной точки до целевой, используя первое наилучшее совпадение в графе.

Обход вершин определяет эвристическая функция «длина пути + цена» (принято обозначать f(x)). Данная функция получается путем сложения двух функций:

1) Стоимость (цена) достижения вершины и начального положения (принято обозначать g(x) – может являться и эвристической, и нет).

2) Эвристическая оценка расстояния от заданной точки до конечной (принято обозначать h(x))

Функция h(x) должна быть допустимой эвристической оценкой, то есть 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.

С использованием данного алгоритма все пути рассматриваются пошагово, и идут от заданной точки к целевой, до тех пор, пока не будет достигнута минимальная стоимость пути. В начале поиска, что характерно для всех информационных алгоритмов поиска, рассматриваются «кажущиеся» маршруты к целевой точки. Отличительной особенностью этого алгоритма от «жадного», является то, что данный алгоритм производит учет всего пройденного пути. Функция g(x), в свою очередь, является показателем стоимости маршрута от начальной позиции, а в «жадном» алгоритме данная функция отвечает только за стоимость пути от предыдущей вершины.

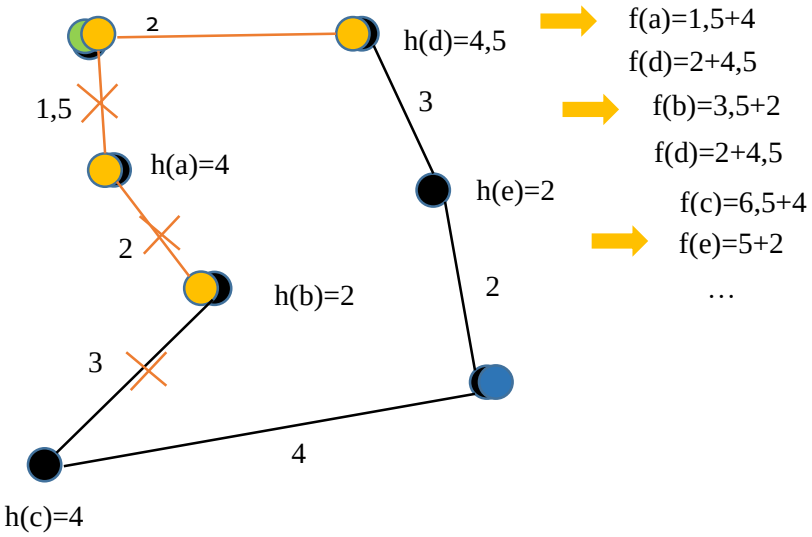

Рис. 5. Использование алгоритма поиска A*

При начале поиска рассматриваются вершины, граничащие с начальной точкой; выбирается вершина с минимальным значением f(x), и алгоритм переходит на этот узел. Во время каждой итерации алгоритмом рассматривается множество путей из начала графа до не посещённых вершин, которые размещаются в специальной очереди с разными приоритетами. Данный приоритет можно определить по выражению f(x) = g(x) + h(x). Итерации продолжаются до тех пор, пока функция f(x) конечной вершины графа не будет наименьшей из всех значений в очереди, либо, когда закончится просмотр графа. Итоговое решение выбирается из всех решений, у которых наименьшая стоимость

Алгоритм A* всегда может найти решение, при условии его существования, и относится к полным алгоритмам [5].

Алгоритм поиска D* (Algorithm D star)

Алгоритм поиска кратчайшего пути во взвешенном ориентированном графе, где структура графа неизвестна заранее или постоянно подвергается изменению.

Постановка задачи. Дан взвешенный ориентированный граф G(V, E). Даны вершины f и t. Требуется в процессе движения по кратчайшему пути в графе G обновлять значения функции g(s) при поступлении новой информации о графе G.

На основе алгоритма A* описывается алгоритм D*, который способен определять расстояние между текущей вершиной f, в которой, допустим, находится способный к сканированию местности "робот", и конечной вершиной t при каждом изменении графа в то время, как "робот" движется вдоль найденного пути [6].

Алгоритм Флойда — Уоршелл (Floyd–Warshall algorithm)

Алгоритм находит кратчайшие пути между всеми вершинами взвешенного ориентированного графа.

Пусть вершины графа $G = (V, E), |V| = n$ пронумерованы от 1 до n и введено обозначение $d_{ij}^{k} (i < j < k)$ для длины кратчайшего пути от i до I, который кроме самих вершин i до I, проходит через вершины 1……к. Очевидно, что d_{ij}^{k}- длина (вес) ребра (i, j), если такое существует (в противном случае его длина может быть обозначена как ∞).

Существует два варианта значения $d_{ij}^{k},\ k \in (1, \dots, n)$

1 кратчайший путь между i, j не проходит через вершину k, тогда $d_{ij}^{k} == d_{ij}^{k-1}$

2 существует более короткий путь между i, j проходящий через k, тогда он сначала идет от i до k, а потом от k до j. В этом случае, очевидно, $d_{ij}^{k} == d_{ik}^{k-1} + d_{kj}^{k-1}$

Таким образом, для нахождения значения функции достаточно выбрать минимум из двух обозначенных значений.

Тогда рекуррентная формула для d_{ij}^{k} имеет вид:

$$d_{ij}^{0} \text{ – длина ребра } (i,j); \quad (1)$$

$$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}). \quad (2)$$

Алгоритм Флойда-Уоршелла последовательно вычисляет все значения $d_{ij}^{k}, \forall i$, j для k от 1 до n. Полученные значения d_{ij}^{k} является длинами кратчайших путей между вершинами i, j [7].

Алгоритм находит кратчайшие пути между всеми парами вершин взвешенного ориентированного графа.

Дан граф $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ с весовой функцией $\omega\colon \mathrm{E} \longrightarrow \mathrm{R}$. Если веса всех рёбер в графе неотрицательные, можно найти кратчайшие пути между всеми парами вершин, запустив алгоритм Дейкстры один раз для каждой вершины. Если в графе содержатся рёбра с отрицательным весом, но отсутствуют циклы с отрицательным весом, можно вычислить новое множество рёбер с неотрицательными весами, позволяющее воспользоваться предыдущим методом. Новое множество, состоящее из весов рёбер ώ, должно удовлетворять следующим свойствам:

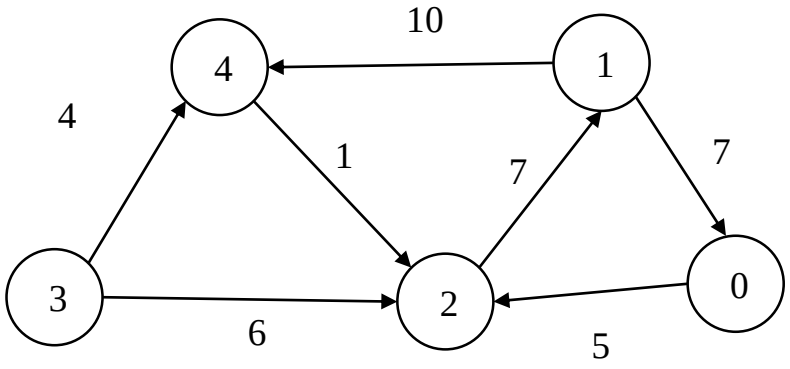

Рис. 6. Использование Алгоритма Флойда — Уоршелла

Алгоритм Джонсона (Johnson's algorithm)

- для всех ребер (u, v) новый вес $\acute{\omega}(\mathrm{u}, \mathrm{v}) > 0$
- для всех пар вершин $\mathrm{u}, \mathrm{v} \in V$ путь p является кратчайшим путем из вершины u в вершину v с использованием весовой функции ω тогда и только тогда, когда p – также кратчайший путь из вершины u в вершину v с весовой функцией ώ [8].

Алгоритм Ли – Алгоритм волновой трассировки (Lee algorithm)

Алгоритм основан на методе поиска в ширину. Находит путь между вершинами s и t графа (s не совпадает с t), содержащий минимальное количество промежуточных вершин (ребер). Основное применение — трассировки электрических соединений на кристаллах микросхем и на печатных платах. Так же используется для поиска кратчайшего расстояния на карте в стратегических играх.

Алгоритм работает на дискретном рабочем поле (ДРП), представляющем собой ограниченную замкнутой линией фигуру, не обязательно прямоугольную, разбитую на прямоугольные ячейки, в частном случае — квадратные. Множество всех ячеек ДРП разбивается на подмножества: «проходимые» (свободные), т. е при поиске пути их можно проходить, «непроходимые» (препятствия), путь через эту ячейку запрещён, стартовая ячейка (источник) и финишная (приемник). Назначение стартовой и финишной ячеек условно, достаточно — указание пары ячеек, между которыми нужно найти кратчайший путь.

Алгоритм предназначен для поиска кратчайшего пути от стартовой ячейки к конечной ячейке, если это возможно, либо, при отсутствии пути, выдать сообщение о непроходимости.

Работа алгоритма включает в себя три этапа: инициализацию, распространение волны и восстановление пути.

Во время инициализации строится образ множества ячеек обрабатываемого поля, каждой ячейке приписываются атрибуты проходимости/непроходимости, запоминаются стартовая и финишная ячейки.

Далее, от стартовой ячейки порождается шаг в соседнюю ячейку, при этом проверяется, проходима ли она, и не принадлежит ли ранее меченной в пути ячейке.

Соседние ячейки принято классифицировать двояко: в смысле окрестности Мура и окрестности фон Неймана, отличающийся тем, что в окрестности фон Неймана соседними ячейками считаются только 4 ячейки по вертикали и горизонтали, в окрестности Мура — все 8 ячеек, включая диагональные.

При выполнении условий проходимости и непринадлежности её к ранее помеченным в пути ячейкам, в атрибут ячейки записывается число, равное количеству шагов от стартовой ячейки, от стартовой ячейки на первом шаге это будет 1. Каждая ячейка, меченая числом шагов от стартовой ячейки становится стартовой и из неё порождаются очередные шаги в соседние ячейки. Очевидно, что при таком переборе будет найден путь от начальной ячейки к конечной, либо очередной шаг из любой порождённой в пути ячейки будет невозможен.

Восстановление кратчайшего пути происходит в обратном направлении: при выборе ячейки от финишной ячейки к стартовой на каждом шаге выбирается ячейка, имеющая атрибут расстояния от стартовой на единицу меньше текущей ячейки. Очевидно, что таким образом находится кратчайший путь между парой заданных ячеек. Трасс с минимальной числовой длиной пути, как при поиске пути в окрестностях Мура, так и фон Неймана может существовать несколько. Выбор окончательного пути в приложениях диктуется другими соображениями, находящимися вне этого алгоритма. Например, при трассировке печатных плат — минимумом линейной длины проложенного проводника [9].

Список использованных источников

1. Воротников С. А., Информационные устройства робототехнических систем, МГТУ имени Н. Э.Баумана, 2005, 384с.
2. Берцун В. Н., Математическое моделирование на графах. Часть 2, Томск, изд-ва Том. Ун-ту, 2003, 88с.
3. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, Introduction to Algorithms. — 2-е изд. — М.: «Вильямс», 2006, 1296с.
4. Алгоритм Беллмана-Форда. Электронный ресурс. Режим доступа: https://ru.wikipedia.org/wiki/Белмана_Форда
5. Рассел С. Дж., Норвиг, П. Искусственный интеллект: современный подход, Artificial Intelligence: A Modern Approach / Пер. с англ. и ред. К. А. Птицына. — 2-е изд.. — М.: Вильямс, 2006, 162с.
6. Алгоритм D, Электронный ресурс. Режим доступа: http://idm-lab.org/project-a.html
7. Алгоритм Флойда-Уоршелла. Электронный ресурс. Режим доступа:
8. https://habr.com/ru/post/105825/
9. Алгоритм Джонсона. Электронный ресурс. Режим доступа:
10. https://habr.com/ru/company/otus/blog/510942/
11. Алгоритм Ли (Волновой алгоритм). Электронный ресурс. Режим
12. доступа: https://ru.wikipedia.org/wiki/Алгоритм_Ли