УДК 004.896
DOI: 10.26102/2310-6018/2026.59.8.004

Алгоритм определения оптимального маршрута движения робота для мониторинга объектов с учетом препятствий

Т.А. Макаровских✉, М.Э.Х.М.М. Саллам

Южно-Уральский государственный университет, Челябинск, Российская Федерация

Резюме. Одной из актуальных задач является наземный мониторинг состояния различных объектов. Его необходимость может возникать не только на промышленных предприятиях, но и в сельском хозяйстве (осмотр плодовых садов), медицине (мониторинг состояния пациентов инфекционных отделений), при ликвидации чрезвычайных ситуаций. В этих ситуациях невозможно размещение датчиков и умных камер в фиксированных местах. Кроме того, возникают ситуации, когда цели для мониторинга задаются непосредственно перед его проведением. Еще одной актуальной задачей является построение пути робота с учетом наличия препятствий. В данной статье рассматривается алгоритм построения кратчайшего маршрута для проведения мониторинга в конечном множестве точек наземным роботом с учетом препятствий. Задача определения траектории движения робота формулируется как задача о маршрутизации транспортного средства. Данные о координатах робота и местоположении точек мониторинга хранятся в виде взвешенного плоского графа с использованием списка ребер. В статье предложен эвристический алгоритм SearchWithObstacles, использующий функцию SPPA (Shortest Path with Precautionary Avoidance), который основан на методе ближайшего соседа и решает задачу за полиномиальное время. Приведено сравнение алгоритма с известными подходами и обсуждены преимущества и недостатки предложенного алгоритма.

Ключевые слова: маршрутизация транспортных средств, плоский граф, эвристический алгоритм, оптимизация, объезд препятствий, задача коммивояжера.

Благодарности: Исследование выполнено при поддержке регионального гранта Российского научного фонда № 26-21-20007.

Для цитирования: Макаровских Т.А., Саллам М.Э.Х.М.М. Алгоритм определения оптимального маршрута движения робота для мониторинга объектов с учетом препятствий. *Моделирование, оптимизация и информационные технологии.* 2026;14(8). URL: https://moitvivt.ru/ru/journal/article?id=2427 DOI: 10.26102/2310-6018/2026.59.8.004

An algorithm for determining the optimal route of the robot for monitoring objects, taking into account obstacles

T.A. Makarovskikh✉, M.E.H.M.M. Sallam

South Ural State University, Chelyabinsk, the Russian Federation

Abstract. One of the urgent tasks is ground-based monitoring of the condition of various facilities. It may be necessary not only in industrial enterprises, but also in agriculture (inspection of orchards), medicine (monitoring the condition of patients in infectious diseases departments), and emergency situations. In these situations, it is not possible to place sensors and smart cameras in fixed locations. In addition, there are situations when monitoring goals are set immediately before it is carried out. Another urgent task is to build a robot's path, taking into account the presence of obstacles. This article discusses an algorithm for constructing the shortest route for monitoring at a finite set of points by a ground robot, taking into account obstacles. The task of determining the trajectory of a robot is formulated as a vehicle routing task. Data on the coordinates of the robot and the location of monitoring points is stored as a weighted planar graph using a list of edges. The article proposes the algorithm SearchWithObstacles, using function SPPA (Shortest Path with Precautionary Avoidance), which is based on the nearest

© Макаровских Т.А., Саллам М.Э.Х.М.М., 2026

neighbor method and solves the problem in polynomial time. The algorithm is compared with known approaches and the advantages and disadvantages of the proposed algorithm are discussed.

Keywords: vehicle routing, planar graph, heuristic algorithm, optimization, obstacle avoidance, traveling salesman problem.

Acknowledgements: The research is supported by a regional grant from the Russian Science Foundation No. 26-21-20007.

For citation: Makarovskikh T.A., Sallam M.E.H.M.M. An algorithm for determining the optimal route of the robot for monitoring objects, taking into account obstacles. *Modeling, Optimization and Information Technology.* 2026;14(8). (In Russ.). URL: https://moitvivt.ru/ru/journal/article?id=2427 DOI: 10.26102/2310-6018/2026.59.8.004

Введение

Одной из актуальных задач является мониторинг и анализ состояния различных наземных объектов. Эта задача может возникнуть, например, в сельском хозяйстве, промышленности, при ликвидации чрезвычайных ситуаций. Существует множество примеров промышленного мониторинга, когда достаточно установить несколько камер либо датчиков для наблюдения за интересующими объектами. Но в ряде ситуаций такой подход невозможен, нецелесообразен либо опасен. Например, для наблюдения за мобильными объектами невозможно эффективно использовать стационарные датчики. При осмотре садов или лесных угодий в несколько сотен гектаров установка датчиков невозможна. В таких случаях использование человеческих ресурсов является сложной, а то и вовсе неосуществимой задачей. Поэтому при обследовании обширных помещений, сельскохозяйственных угодий, инфекционных отделений больниц, обследовании мест чрезвычайных ситуаций актуальна задача построения кратчайшего маршрута объезда интересующей территории, проведения мониторинга и передачи данных из заданных точек.

Для всех рассмотренных выше случаев на обследуемой территории имеется ряд препятствий. Их можно подразделить на постоянные (те, которые в течение длительного времени не меняют своего местоположения: деревья, крупная мебель, стационарные установки, перегородки) и временные (те, которые могут находиться в зоне проведения мониторинга непродолжительное время: люди, животные, мелкая мебель, подвижная техника). Устройство мониторинга (робот) должно учитывать эти препятствия при планировании оптимальной траектории мониторинга и соответствующим образом корректировать маршрут своего передвижения.

Существующие подходы к решению задачи маршрутизации обычно не учитывают наличие препятствия: в классической постановке задачи учитывается только количество объектов, которые необходимо посетить и уже рассчитанная длина путей между парами объектов. После планирования маршрута внесение изменений уже невозможно.

Большинство современных работ, например, [1], рассматривают постановку в терминах классической задачи коммивояжера (Travelling Salesman Problem, TSP). В [1], где предлагается графовая модель задания исследуемой области, выполняется построение покрытия с помощью нахождения деревьев решений (взятых авторами как наиболее подходящая эвристика) для повышения эффективности и точности. Каждая вершина графа соответствует точке интереса, а ребра представляют расстояния между этими точками. Результаты проведенных авторами вычислительных экспериментов показывают, что по сравнению с классическими алгоритмами решения TSP предлагаемый ими метод значительно повышает качество полученного решения при одновременном снижении вычислительных затрат. Однако этот подход разработан для

беспилотных летательных аппаратов и поэтому не учитывает наличие препятствий и работает с фиксированными расстояниями между объектами.

Ряд фундаментальных идей по поиску пути, проходящего через все точки интереса, подробно рассмотрен в работах [2, 3], где авторы анализируют так называемый мегаполис (непустое конечное множество). Использование их подхода может быть полезным при решении ряда задач мониторинга с учетом препятствий. При решении задачи маршрутизации авторы задают условия предшествования, которые должны выполняться при определении маршрута. Функции затрат, используемые для формулировки критерия дополнения, могут зависеть от списка задач. Для разработки решения предлагается двухэтапная процедура, основанная на методах динамического программирования [4]. Однако эти алгоритмы не учитывают возможность появления ни статических, ни динамических препятствий и проведение реоптимизации. Возможность обработки информации о препятствиях отсутствует и в работах [5, 6].

В статьях, посвященных анализу препятствий, таких как [7], основное внимание при построении кратчайшего безопасного пути для объезда препятствий уделяется не планированию маршрута, а физическим аспектам осуществления объезда, например, определению поворотной точки на основе информации о свободных участках. Существует ряд работ, к которым можно отнести, например, [8], в которых рассматривается полезная кинематическая модель для навигации по участкам с различными препятствиями и предлагаются алгоритмы определения траекторий. Однако в большинстве этих исследований решается только локальная проблема определения одного сегмента пути.

Несомненно, современные исследователи в области планирования маршрута и анализа локальной информации зачастую прибегают к нейросетевым подходам. Зачастую глубокие нейронные сети используются не сколько для планирования самого маршрута, сколько для детекции и/или классификации препятствий [9].

Часто задача планирования маршрута рассматривается исключительно в терминах научно-технической проблемы, когда в статье не приводится формализации задачи в терминах теории оптимизации либо теории графов. Так, в работе [10] авторы подробно описывают алгоритм планирования траектории движения мобильного робота под названием Generalized Laser Simulator (GLS), предназначенный для автономной навигации мобильных роботов при наличии статических и динамических препятствий. Этот алгоритм определяет оптимальный путь от начальной точки до цели, генерирует последовательность точек во всех возможных направлениях и осуществляет поиск наиболее подходящего продолжения пути для достижения целевой позиции. Этот алгоритм позволяет определить минимальное расстояние до цели при максимальном удалении робота до границы либо препятствия. В своей работе авторы сообщают о многочисленных успешных экспериментах, однако при отсутствии теоретической базы провести сравнение полученных результатов с другими подходами затруднительно.

На практике в мобильной робототехнике используется подход с разделением задачи построения маршрута на стратегический и локальный поиск. Для заданного множества целей определяется опорный маршрут с учетом необходимых ограничений. Далее робот по данным сенсорных систем выполняет поиск пути на локальной карте к точкам опорного маршрута с использованием алгоритмов A-Jump, D* и др. [11]. Обычно (когда используется двухуровневая схема, предполагающая сначала грубое планирование, а затем уточнение) построение опорного маршрута выполняется однократно и его перерасчет выполняется редко. На локальном уровне происходит только коррекция пути между двумя соседними точками опорного маршрута. В данной статье рассматривается задача адаптивного планирования, когда при обнаружении препятствия осуществляется выбор стратегии реагирования: объезд текущего

препятствия (как в известных подходах), перепланирование пути (переупорядочение целей), ожидание либо завершение миссии (при невозможности посещения непосещенных целей). Эту задачу решает предлагаемый в статье алгоритм SPPA (Shortest Path with Precautionary Avoidance, кратчайший путь с соблюдением мер предосторожности). В нем используется вершинно-взвешенная модель окружающей среды, а для стратегического перераспределения целей при обнаружении препятствий используется адаптивная функция оценки их преодолимости. Такой подход позволяет получить допустимое решение с точки зрения минимизации суммарного времени посещения всех объектов для задачи наземного мониторинга. Таким образом, рассмотренный в статье алгоритм SPPA дополняет существующие подходы (D*, A-Jump) и позволяет построить единую стратегию реоптимизации всего маршрута в условиях динамических ограничений.

Проблема, обсуждаемая в статье, имеет некоторые сходства с проблемой беспилотника и грузовика [12], которая также включает в себя устройство мониторинга (беспилотник) и устройство технического обслуживания (грузовик). Однако ключевые отличия в нашей задаче заключаются в том, что робот должен быть способен преодолевать препятствия, и в некоторых случаях грузовик не нужен. Рассмотренные в данной статье результаты имеют некоторое сходство с работой [13], в которой представлены алгоритмы эвристического поиска в реальном времени для кинодинамического планирования движения с динамическими препятствиями. Однако формализация задачи и алгоритмы различаются.

Проведенный анализ публикаций показывает, что построению путей с учетом динамических препятствий зачастую не уделяется должного внимания. В существующих публикациях решение этой задачи либо отсутствует, либо решается без учета реоптимизации дальнейшего маршрута. В данной статье представлен алгоритм определения кратчайшего маршрута для робота, осуществляющего наземный мониторинг при наличии динамически возникающих препятствий. Алгоритм допускает возможность реоптимизации и пересчета пути с учетом вновь поступившей локальной информации, то есть данный алгоритм решает задачу адаптивной стратегической маршрутизации в условиях неопределенности.

Материалы и методы

Формальная постановка задачи и описание ограничений. Необходимо провести мониторинг группы объектов M_i, где $i = 1, \dots, M$, расположенных в пределах определенной области F. Область F определяется как многоугольник, в общем случае невыпуклый. Координаты N вершин многоугольника $(x_j, y_j), j = 1, \dots N$, представлены в виде вектора и перечислены по часовой стрелке, начиная с вершины с наименьшей координатой x. Также задана карта текущего местоположения и проходимости препятствий $\Omega_{cur}(F)$. Объекты M_i, находятся в области F. Известны расстояния $dist(M_i, M_j), i, j = 1, \dots, M, i \neq j$ между всеми парами объектов. Необходимо определить кратчайший путь, избегающий препятствия, проходящий через каждый объект M_i для $i = 1, \dots, M$, ровно по одному разу. Путь начинается с точки парковки M_0 (парковка) и завершается в той же точке. Данная постановка задачи представляет собой каноническую формулировку задачи коммивояжера. Обычно для построения кратчайшего пути, проходящего через все точки, используется эвристический подход, при котором выбирается ближайшая не посещенная вершина (ближайший сосед). Важно отметить, что вычислительная сложность таких алгоритмов составляет $O(|M|^3)$, где M – количество точек мониторинга. При перемещении между объектами могут возникать препятствия $S_i \in \Omega_{cur}$, которые робот не может преодолеть. Это ведет к пересчету

длины маршрута, изменению траектории робота и, возможно, определению невозможности преодоления оставшейся части маршрута.

Препятствия S_i, $i = 1, \dots, S$, можно классифицировать как временные и стационарные. К временным препятствиям относятся либо подвижные объекты (люди, животные, автомобили и техника), либо неподвижные, но устранимые со временем (шлагбаумы, двери, припаркованная техника, предметы временного хранения). Остальные препятствия относятся к стационарным или малоподвижным (новые стены, конструкции и закрепленная мебель). Будем считать, что в точках окружающей среды $\Omega_{cur}(F)$, покрытых препятствием, задан коэффициент проходимости $\omega(v) \in [0; 1]$. Этот коэффициент будет использоваться при планировании маршрута и выборе стратегии реагирования на препятствия.

Поскольку из-за наличия препятствий может быть ограничена возможность преодоления части маршрута, необходимо оценить разрешимость проблемы маршрутизации на текущем этапе при обнаружении препятствия в процессе мониторинга.

Вся область, предназначенная для перемещения робота, разделена на небольшие квадратные сегменты. Таким образом, эту область целесообразно представить в виде плоского графа $G(V, E)$. Множество вершин V состоит из равноотстоящих точек (сетки), покрывающих исследуемую область, причем $V = V_m \cup V_p \cup V_t$, где множество V_m – множество вершин мониторинга ($M_i \in V_m, i = 1, \dots, M$), V_p – множество вершин парковки (в случае одного депо/парковки $M_0 \in V_p$), V_t – множество транзитных вершин. Это точки, между которыми перемещается робот. Множество ребер E – это связи между ближайшими соседями из множества V. Кодирование графа подробно обсуждается в [14].

Необходимо минимизировать общую длину маршрута

$$W = \sum_{e \in D} l(e) \to min, \tag{1}$$

где ребра e соответствуют траектории движения робота, $l(e)$ – длина ребра e. Пусть x_{ij} – бинарная переменная для принятия решения, указывающая, будет ли робот перемещаться непосредственно из вершины $v_i \in V$ в вершину $v_j \in V$. В этих терминах целевая функция (1) может быть задана следующим образом:

$$W = \sum_{j=0}^{|V|} \sum_{j=0, j \neq i}^{|V|} x_{ij} \cdot l(e_{ij}) \to min, \tag{2}$$

Ограничение (3) указывает, что каждая точка мониторинга $M_i \in V_m$ должна быть посещена транспортным средством только один раз:

$$\sum_{j=0, j \neq \mathrm{i}}^{|V_m|} x_{ij} = 1, \ \forall i \ \in V_m \setminus V_p. \tag{3}$$

Все остальные точки можно посетить более одного раза.

Ограничение (4) требует, чтобы устройство для мониторинга стартовало из вершины парковки (депо, $M_0 \in V_p$) и возвращалось в вершину парковки после обслуживания посещения всех точек $M_i \in V_m$.

$$\sum_{i \in V, i \neq j} x_{ij} - \sum_{h \in V, h \neq j} x_{jh} = 0\,, \ \forall j \ \in V \ \setminus V_p. \tag{4}$$

Далее рассмотрим алгоритм маршрутизации SearchWithObstacles(), который использует процедуру SPPA (Shortest Path with Precautionary Avoidance), используемую для решения задачи планирования маршрута в графовой постановке, т. е. алгоритм SearchWithObstacles() находит гамильтонов цикл допустимой длины в полном взвешенном графе, вершинами которого являются точка парковки и точки мониторинга.

Алгоритм для маршрутизации робота с динамическим перепланированием. Алгоритм маршрутизации для идеального случая, когда вновь возникающие препятствия отсутствуют, обсуждается в [14].

При проектировании алгоритма поиска с препятствиями будем полагать, что маршрут W уже учитывает все внешние воздействия и что ресурсов аккумулятора и карты памяти достаточно для завершения полного цикла мониторинга (ограничений вида (3) нет). Возможность продолжить движение по пути определяется как возможность достичь центральной точки следующего элементарного отрезка (следующей вершины графа G). Если в некоторый момент времени на пути робота к этой точке обнаружено препятствие (при помощи камеры и ультразвука), дальнейшее движение по ранее построенному маршруту считается невозможным и выполняется поиск альтернативного обходного пути до ближайшей точки мониторинга. Если на каком-либо этапе прохождения пути не удается найти обходной путь, робот отправляет оператору сообщение о невозможности продолжения процесса мониторинга и возвращается на парковку. Для классификации полученных маршрутов в [14] введены понятия разрешимых, частично разрешимых и неразрешимых маршрутов.

В качестве исходных данных для алгоритма SPPA задаются размеры ячеек в дискретном представлении карты исследуемой местности Ω_{cur}, причем каждая ячейка на карте имеет вес $\omega \in [0; 1]$, который определяет уровень проходимости (0 – легко проходимая область, 1 – непроходимая область). Такая информация помогает учитывать при планировании маршрута не только наличие непреодолимых препятствий, но и неровности местности (мокрый пол, вспаханная почва, разбросанный мелкий мусор).

При построении результирующего пути будем использовать локальный поиск, который работает по следующему принципу. Алгоритм SearchWithObstacles по сути является эвристикой поиска ближайшего соседа с динамическим перепланированием.

Алгоритм SearchWithObstacles

Входные данные: $G(V, E)$ – граф, задающий рабочую область; $M_0 \in V_p$ — начальная точка (парковка/депо); $M_i \in V_m$ – множество точек мониторинга; Ω_{cur} – текущая карта препятствий с указанием веса каждой ячейки рабочей области (вершины графа $G(V, E)$).

Выходные данные: W – замкнутый маршрут из вершины $M_0 \in V_p$, проходящий через все $M_i \in V_m$.

Шаг 1. Инициализация. Определить множество посещенных точек мониторинга $V_m^{vis} = \emptyset$. Инициализировать маршрут $W = M_0$. Задать текущее положение робота $v_{cur} = M_0$, счетчик реоптимизаций $R = 0$.

Шаг 2. Пока существуют непосещенные точки мониторинга ($\left|V_m^{vis}\right| \neq |V_m|$) либо не достигнута точка парковки $M_0 \in V_p$, выполнить следующие действия.

Шаг 2.1. Выбрать ближайшую непосещенную точку мониторинга:

$$m_{next} = arg \min_{v \in V_m \setminus V_m^{vis}} dist(v_{cur}, v). \quad (5)$$

Шаг 2.2. Рассчитать путь до выбранной точки мониторинга с учетом текущих препятствий:

$$W_{seg} = SPPA(v_{cur}, m_{next}, \Omega_{cur}). \quad (6)$$

Шаг 2.3. Пройти по маршруту до точки m_{next} и на каждом шаге провести мониторинг препятствий.

2.3.1. Просканировать пространство на наличие препятствий и получить текущую карту препятствий: $\Omega_{cur} = ScanEnvironment()$.

2.3.2. При обнаружении препятствия, блокирующего путь, обновить веса ребер графа $G(V, E)$ в зоне обнаружения препятствий: $w(e) = w(e) + \omega$, где ω – коэффициент проходимости препятствия, $\omega \in [0; 1]$.

2.3.3. Проверить доступность текущей цели m_{next} для текущей карты препятствий Ω_{cur}. Если маршрут до m_{next} построить не удается, выполнить пересчет ближайшей цели по формуле (5) и переопределить W_{seg} по формуле (6). $R = R + 1$. Начать движение к новой цели. Иначе перейти на шаг 2.3.4.

2.3.4. Если вершина m_{next} достижима, но на пути W_{seg} обнаружено препятствие, построить обходной путь

$$W_{alt} = CreateDetourPath(v_{cur}, m_{next}, \Omega_{cur})$$

с учетом текущей карты препятствий Ω_{cur}. $W_{seg} = W_{alt}$. Если обходной путь не найден, построить кратчайший маршрут W_{seg} до парковки $m_{next} = M_0$.

2.3.5. Продолжить движение по W_{seg}, пока не будет достигнута целевая вершина m_{next} (повторять шаг 2.3)

Шаг 2.4. Отметить достигнутую вершину m_{next} как посещенную: $V_m^{vis} = V_m^{vis} \cup m_{next}$. Сохранить пройденный путь $W = W \cup W_{seg}$.

Шаг 3. Вернуться на парковку $W_{ret} = \text{SPPA}(v_{cur}, M_0, \Omega_{cur})$, $W = W \cup W_{ret}$.

Шаг 4. Конец алгоритма. W – построенный путь.

В описанной в [14] графовой модели степень каждой вершины $v \in V$ не должна превышать 8 (по два возможных перемещения по горизонтали и вертикали, и 4 перемещения по диагонали). Таким образом, локальный поиск для выбора следующей вершины графа для включения в результирующий маршрут производится среди как максимум восьми смежных вершин. Приоритет выбора отдается инцидентному ребру с наименьшим весом (наиболее легко преодолеваемый маршрут), которое ведет в смежную вершину, наиболее близкую к непройденной точке мониторинга $M_i \in V_m$. Такой подход минимизирует разброс в процессе поиска вершин, которые находятся далеко от $M_i \in V_m$, тем самым сокращая время для поиска допустимого решения. Уже посещенная $M_i \in V_m$ получает соответствующую пометку и более не рассматривается как ближайший сосед при локальном поиске. Такой подход гарантирует, что вершины, входящие в путь, не повторяются более одного раза (выполняется условие (4)) и что количество используемых транзитных вершин минимально.

Задачу определения оптимального маршрута для проведения мониторинга можно разделить на следующие этапы. В качестве вспомогательных структур алгоритм использует два списка L_{open} и L_{closed}. Список L_{open} используется для хранения вершин, доступных для перемещения, а список L_{closed} предназначен для тех вершин, которые недоступны.

Идея алгоритма SPPA имеет много общего с известным алгоритмом A* [15, 16], его можно назвать адаптацией алгоритма A* к специфике задачи последовательного построения маршрута с динамическими препятствиями: он объединяет вершинно-взвешенную модель определения препятствий со стратегией, при которой путь строится не к одной фиксированной цели, а к ближайшей из еще не посещенных точек мониторинга $M_i \in V_m$. В отличие от классического алгоритма A*, в котором целевая вершина фиксирована на протяжении всего поиска, предлагаемый алгоритм SPPA реализует динамическое переключение целей.

Так, в классическом алгоритме A* функция оценки для вершины v определяется как:

$$f_{A*}(v) = g(v) + h(v),$$

где $g(v)$ – фактическая стоимость пути от начальной точки пути до вершины v; $h(v)$ – эвристическая оценка стоимости пути от вершины v до целевой вершины m_{next}.

В алгоритме SPPA функция оценки расширена путем добавления коэффициента проходимости препятствия:

$$f_{SPPA}(v) = g(v) + h(v) + \lambda \cdot \omega(v), \quad (7)$$

где $\omega(v) \in [0;1]$ – коэффициент препятствия в вершине v (вес ячейки карты, отражающий сложность проходимости); λ – коэффициент влияния препятствий (эмпирически установлен $\lambda = 0{,}5$).

Коэффициент препятствия определяется на основе карты проходимости в текущий момент времени Ω_{cur}:

$$\omega(v)_{max} = \left(\alpha_s \cdot s(v), \alpha_{ss} \cdot ss(v), \alpha_d \cdot d(v)\right), \quad (8)$$

где $s(v) = \{0;1\}$ наличие статического препятствия в вершине v; $ss(v) \in [0;1]$ – коэффициент для полуподвижного препятствия; $d(v) \in [0{,}1]$ – коэффициент близости динамического препятствия; $\alpha_s = 1{,}0$, $\alpha_{ss} = 0{,}7$, $\alpha_d = 0{,}5$ – весовые коэффициенты.

Отличия от функции оценки алгоритма A* можно записать в виде Таблицы 1.

Таблица 1 – Отличия функции оценки SPPA от A*
Table 1 – Differences between the SPPA evaluation function and A*

Характеристика	Классический A*	SPPA
Функция оценки	Формула (5)	Формула (6)
Учет препятствий	Бинарный (проходимо/непроходимо)	Взвешенный (коэффициент проходимости $\omega(v) \in [0;1]$)
Целевая точка	Фиксированная	Динамически переключаемая (ближайшая непосещенная)
Адаптация к изменениям	Требует полного пересчета	Локальное обновление и переключение цели

Основное отличие функции оценки SPPA от классической заключается в добавлении взвешенного коэффициента препятствия $\lambda \cdot \omega(v)$, что позволяет алгоритму учитывать не только геометрическую длину пути, но и проходимость участков местности. Таким образом, предложенный алгоритм адаптивен к реальным условиям эксплуатации мобильных роботов.

На каждом этапе движения в качестве текущей цели выбирается ближайшая непосещенная точка мониторинга m_{next}. При обнаружении препятствия, блокирующего путь к текущей цели, алгоритм переключается на следующую ближайшую точку, не дожидаясь полного пересчета маршрута. Это позволяет перестраивать глобальный порядок обхода в реальном времени без необходимости повторного запуска алгоритма поиска пути с нуля.

Алгоритм SPPA

Входные данные: карта области F с указанными границами, карта препятствий Ω_{cur} с заданными весами всех вершин $\omega(v_i) \in [0;1]$, стартовая вершина v_s, цель – v_g. Параметры для вычисления эвристики: $\alpha = 1{,}5$, $\lambda = 0{,}4$.

Выходные данные: кратчайший маршрут $W = v_s, \dots, v_g$, избегающий препятствий из Ω_{cur}.

Шаг 1. Инициализация. Инициализировать списки достижимых вершин $L_{open} = v_s$ и недоступных вершин $L_{closed} = \emptyset$. Назначить веса вершинам, в которых находятся

препятствия $\omega(v_i) \in [0;1]$. Положить $g(v_s) = 0$, $f(v_s) = h(v_s, v_g) + \lambda \cdot \omega(v_s)$, где $h(v_s, v_g) = dist(v_s, v_g)$, причем, для вычисления расстояния можно использовать любую известную метрику.

Шаг 2. Пока $L_{open} \neq \emptyset$, выполнить следующие действия.

Шаг 2.1. Определить ближайшую доступную вершину $v_{cur} = \{v | v \in L_{open}, f(v) \to min\}$. Если $v_{cur} = v_g$, завершить выполнение алгоритма, путь W найден, восстановить его из v_{cur} по сохраненным меткам родителей $par(v)$.

Шаг 2.2. $L_{open} = L_{open} \setminus v_{cur}$, $L_{closed} = L_{closed} \cup v_{cur}$.

Шаг 2.3. Для каждой вершины v_n, доступной для посещения после выхода из v_{cur} и $v_n \notin L_{closed}$.

Рассчитать предварительную длину пути $t_n = g(v_{cur}) + dist(v_{cur}, v_n)$.

Если $v_n \notin L_{open}$, то $L_{open} = L_{open} \cup v_n$, иначе, если $W_n \geq g(v_n)$ перейти к следующей вершине-соседу.

Определить родителя $\text{par}(v_n) = v_{cur}$ и вычислить $g(v_n) = t_n$, $h_{val} = h(v_n, v_g)$, $\omega_{val} = GetObstCoef(v_n, \Omega_{cur})$(получить коэффициенты препятствий на пути), $f(v_n) = g(v_n) + h_{val} + \lambda \cdot \omega_{val}$.

Шаг 3. Если цикл завершен без нахождения цели, вернуть сообщение об ошибке (Path Not Found).

Конец алгоритма SPPA.

Теорема [17]. Алгоритм SPPA решает задачу маршрутизации с препятствиями и выполняется за время $O(|V|^3)$. Алгоритм получает допустимое решение с заданной сложностью.

Результаты

Рассмотрим задачу проведения мониторинга. Путь робота начинается и заканчивается на парковке и проходит через все отмеченные точки мониторинга. Требуется определить кратчайший замкнутый путь, проходящий через все точки мониторинга с учетом отмеченных на карте препятствий. Для проведения тестирования авторами разработано приложение, в котором пользователь задает карту для проведения мониторинга, на которой указывает положение парковки, точек мониторинга и отмечает препятствия. Каждая точка, принадлежащая препятствию, задается на карте при помощи коэффициента проходимости. Например, стены, мебель и прочие препятствия, которые робот переехать не может, имеют проходимость 1; мокрый пол, мусор, неровности, снижающие проходимость, имеют коэффициент порядка 0,1; и т. д.

Цель тестирования – определить время в секундах, требуемое алгоритму SPPA, для поиска ближайших соседей при построении маршрута робота между двумя, 5, 10, 15, 20, 25 точками на фиксированной карте препятствий, а также сравнение времени работы алгоритма SPPA с известными и наиболее широко применимыми методами для решения рассматриваемой задачи: современной модификацией A* [15, 18] и муравьиной колонией (Ant Colony Optimization, ACO). В контексте данной работы выбор этих алгоритмов был продиктован желанием охватить три основные парадигмы решения задач навигации и планирования траекторий.

Алгоритм A* выбран как представитель классических детерминированных методов, это традиционный и хорошо известный подход к планированию маршрута [16].

Муравьиный алгоритм (ACO) выбран как представитель класса метаэвристических и биоинспирированных алгоритмов. ACO является стандартом де-факто для оценки эффективности алгоритмов поиска пути в динамических средах и на сложных графах, где классические методы могут застревать в локальных оптимумах.

Сравнение с ACO позволяет показать, насколько предлагаемый SPPA эффективнее в условиях неопределенности по сравнению со стохастическими методами оптимизации.

Таким образом, сравнение алгоритмов проводится на уровне эффективности решения конечной прикладной задачи (минимизация длины пути, время вычисления) различными методологическими подходами. Это позволяет доказать универсальность и робастность предлагаемого SPPA.

Полученные результаты помогут определить целесообразность и тактику использования алгоритмов при пересчете пути в случае изменения карты (возникновения либо удаления ряда препятствий). Для тестирования алгоритма SPPA проведены эксперименты с разными известными метриками для вычисления $dist(a, b)$: манхэттенская, евклидова, евклидова NoSQR и максимальное расстояние по Чебышеву.

В проведенном вычислительном эксперименте сравнивается трудоемкость алгоритма SPPA с трудоемкостью одной из современных модификаций алгоритма A* [19, 20] для решения одной и той же задачи, которая допускает диагональные перемещения. На первый взгляд наиболее очевидным алгоритмом поиска ближайшего соседа представляется давно известный и хорошо изученный алгоритм Дейкстры. Но этот алгоритм позволяет найти расстояние между фиксированной парой вершин и фиксированными расстояниями. В рассматриваемой задаче расстояние между вершинами может меняться, кроме того, на каждом шаге получается эвристическая информация, которая может помочь ускорить поиск решения. В таком случае лучшим выбором являются алгоритмы из семейства A* и подобные подходы. Может показаться целесообразным использование для сравнения волнового алгоритма (известного также как алгоритм Ли). Этот алгоритм привлекает своей простотой в реализации, поскольку использует для поиска решения хорошо известный BFS-поиск, однако он не учитывает веса ребер и неэффективен на больших графах. Кроме того, в статье [17] представлен результат эксперимента, где этот алгоритм превосходит A* на невзвешенном графе с 3600 вершинами. Однако в данной статье рассматриваются взвешенные графы, с которыми алгоритм Ли не работает. Результаты сравнения алгоритма SPPA при построении замкнутого пути между парковкой и заданным множеством точек с такими подходами как A*, D*, ACO приведены в Таблице 4. Вычисления проводились на персональном компьютере Intel (R) Xeon-2670 v2, 2.5 GHz, RAM 64 Gb. Для корректного сравнения алгоритм ACO был реализован со следующими параметрами, выбранными на основе рекомендаций [19, 20] (Таблица 2).

Таблица 2 – Параметры для алгоритма ACO, используемые для проведения эксперимента
Table 2 – Parameters for the ACO algorithm used for conducting the experiment

Параметр	Обозначение	Значение	Обоснование
Количество муравьев	M	50	Обеспечивает достаточное разнообразие исследуемых путей без значительного увеличения вычислительных затрат
Количество итераций	$Niter$	100	Достаточно для сходимости алгоритма к стабильному решению для графов размером до 25 вершин
Коэффициент влияния феромона	A	1,0	Контролирует степень влияния накопленного феромона на выбор пути (значение 1,0 обеспечивает линейную зависимость)

Таблица 2 (продолжение)
Table 2 (continued)

Коэффициент влияния эвристики	B	2,0	Увеличивает важность эвристической информации (расстояния) при выборе пути
Коэффициент испарения феромона	P	0,1	Обеспечивает постепенное забывание устаревших путей со скоростью 10 % за итерацию
Начальное значение феромона	τ_0	1,0	Базовое значение феромона на всех ребрах графа перед началом работы алгоритма
Коэффициент усиления феромона	Q	100	Определяет количество феромона, оставляемого муравьями на пройденном пути

В Таблице 3 приведено краткое описание каждого из рассматриваемых в эксперименте алгоритмов. В Таблице 4 приведены результаты вычислений только при использовании евклидовой метрики при определении расстояния. Для манхэттенской, чебышевской и евклидовой NoSqr метрик результаты аналогичны. Из таблицы видно, что алгоритм SPPA для графов с количеством целей (вершин мониторинга) более 10 по временным затратам превосходит известные аналоги на поиск решения. D* эффективнее при малом числе целей (5–10), но SPPA лучше решает поставленную задачу при большем количестве целей (более 15).

Таблица 3 – Сравнение рассмотренных в эксперименте алгоритмов
Table 3 – Comparison of the algorithms considered in the experiment

Алг.	**Направление поиска**	**Цель**	**Реакция на препятствия**	**Эвристическая функция**	**Область применения**
A*	От старта к цели	Фикс.	Полный пересчет пути	$h(n) = = dist(v_s, v_g)$	Статические среды
D*	От цели к старту (обратный поиск)	Фикс.	Локальное обновление (только затронутые узлы)	$h(n) = = dist(v_g, v_s)$	Динамические среды, малое число целей
ACO	Множественные агенты (муравьи)	Фикс.	Перестроение феромонных следов	Эвристика перехода: $\eta_{ij} = 1/d_{ij}$	Комбинаторная оптимизация
SPPA	От старта к текущей цели	Динамическая (ближайшая непосещенная)	Смена цели и локальный объезд	$h(n) = = dist(v_s, m_{next})$	Динамические среды, большое число целей

Таблица 4 – Сравнение стоимости полученного пути и времени вычисления результата для алгоритмов A*, ACO, D* и SPPA для евклидовой метрики
Table 4 – Comparison of the cost of the obtained path and the calculation time of the result for the A*, ACO, D* and SPPA algorithms for the Euclidean metric

Алгоритм	**A***		**ACO**		**D*[1]**		**SPPA**	
Число точек	Стоим.	Время, с	Стоим.	Время, с	Стоим.	Время, с	Стоим.	Время, с
5	70	7,04	80	9,03	**52**	**3,70**	55	5,04
10	118	8,04	135	9,04	**108**	**5,80**	120	8,04
15	**135**	9,04	260	14,04	170	10,50	160	**8,04**
20	190	22,07	175	19,07	172	19,00	**160**	**16,02**
25	225	25,05	185	26,05	188	23,00	**175**	**22,05**

Обсуждение

Предложенный алгоритм представляет собой гибридную эвристику, сочетающую модифицированный поиск A* с жадной стратегией последовательного обхода множества целевых точек (точек мониторинга). Его особенность заключается в использовании вершинно-взвешенной модели графа, где степень «опасности» препятствия определяется как вес вершины графа. В отличие от муравьиных и генетических алгоритмов, алгоритм SPPA является детерминированным приближенным алгоритмом поиска пути. Проведенные вычислительные эксперименты показали, что алгоритм SPPA превосходит, например, ACO как по производительности, так и по стоимости найденного решения.

Результаты проведенных вычислительных экспериментов также показывают, что алгоритм SPPA эффективен при построении маршрутов для средних и больших наборов точек мониторинга. При этом известный подход D* эффективен на малом наборе данных (до 10 точек). Преимуществом алгоритма SPPA перед известными подходами A* и D* является возможность стратегического переупорядочивания целевых точек при обнаружении препятствия в окружающей среде, что позволяет сократить время на возможную реоптимизацию маршрута.

При помощи предложенного алгоритма можно решить одну из проблем современной логистики и автономной навигации – оперативного стратегического перепланирования маршрута при возникновении непредвиденных препятствий.

Заключение

В статье рассмотрен алгоритм для решения задачи планирования траектории для проведения промышленного мониторинга с учетом динамических препятствий. Предложен алгоритм SearchWithObstacles, который использует эвристику SPPA для планирования последовательности посещения точек мониторинга с выбором стратегии реагирования на динамически возникающие препятствия. Данный алгоритм является адаптацией известного алгоритма A* для задачи последовательного обхода с динамическим штрафом за опасные зоны. Алгоритм может быть использован при решении задачи мониторинга инфраструктуры, доставки в городской среде с учетом дорожной обстановки, а также для управления автономными мобильными роботами на складах или в производственных цехах.

[1] Stentz A. The D* Algorithm for Real-Time Planning of Optimal Traverses. Report No.: CMU-RI-TR-94-37. Pittsburgh: The Robotics Institute, Carnegie Mellon University; 1994. URL: https://archive.org/details/DTIC_ADA289361

К направлениям будущих исследований можно отнести статистическую оценку результатов многочисленных экспериментов, адаптацию алгоритма для работы в динамике, когда требуется пересчет траектории с учетом препятствий, возникших после начала проведения мониторинга. Кроме того, планируется спроектировать систему компьютерного зрения, которая могла бы детектировать динамически возникающие перемещающиеся препятствия на пути движения робота. В данном контексте алгоритм должен использовать модуль искусственного интеллекта для распознавания препятствий, определения их контуров, включения их в карту и пересчета маршрута.

Предполагается разработка системы планирования маршрута, в которой при обнаружении препятствия на пути перемещения робота, размеры обнаруженного препятствия наносятся на карту проведения мониторинга, и в дальнейшем выполняется поиск кратчайшего пути объезда и корректировка длины результирующего маршрута, а также (возможно) последовательности посещения точек мониторинга.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Sundarrajan M., Jothi A., Prabakar D., et al. The Smart Coverage Path Planner for Autonomous Drones Using TSP and Tree Selection. In: *Mining Intelligence and Knowledge Exploration (MIKE 2023), 28–30 June 2023, Kristiansand, Norway*. Cham: Springer; 2023. P. 161–172. https://doi.org/10.1007/978-3-031-44084-7_16
2. Ченцов А.Г. Задача маршрутизации «на узкие места» с системой первоочередных заданий. *Известия Института математики и информатики Удмуртского государственного университета*. 2023;61:156–186. https://doi.org/10.35634/2226-3594-2023-61-09
   Chentsov A.G. A bottleneck routing problem with a system of priority tasks. *Izvestiya Instituta Matematiki i Informatiki Udmurtskogo Gosudarstvennogo Universiteta*. 2023;61:156–186. (In Russ.). https://doi.org/10.35634/2226-3594-2023-61-09
3. Ченцов А.Г., Ченцов П.А. Экстремальная двухэтапная задача маршрутизации и процедуры на основе динамического программирования. *Труды Института математики и механики УрО РАН.* 2022;28(2):215–248. https://doi.org/10.21538/0134-4889-2022-28-2-215-248
   Chentsov A.G., Chentsov P.A. An extremal two-stage routing problem and procedures based on dynamic programming. *Trudy Instituta Matematiki i Mekhaniki UrO RAN.* 2022;28(2):215–248. (In Russ.). https://doi.org/10.21538/0134-4889-2022-28-2-215-248
4. Петунин А.А., Ченцов А.Г., Ченцов П.А. Оптимальная маршрутизация в задачах последовательного обхода мегаполисов при наличии ограничений. *Челябинский физико-математический журнал.* 2022;7(2):209–233. https://doi.org/10.47475/2500-0101-2022-17205
   Petunin A.A., Chentsov A.G., Chentsov P.A. Optimal routing in problems of sequential traversal of megapolises in the presence of constraints. *Chelyabinsk Physical and Mathematical Journal.* 2022;7(2):209–233. (In Russ.). https://doi.org/10.47475/2500-0101-2022-17205
5. Wu J., Li M., Gao Ch., et al. Research on Deployment Scheme and Routing Optimization Algorithm of Distribution Cable Condition Monitoring Devices. *Energies*. 2023;16(19):6930. https://doi.org/10.3390/en16196930
6. Makarovskikh T., Panyukov A., Abotaleb M., et al. Optimal Route for Drone for Monitoring of Crop Yields. In: *Advances in Optimization and Applications (OPTIMA 2023), 18–22 September 2023, Petrovac, Montenegro.* Cham: Springer; 2024. P. 228–240. https://doi.org/10.1007/978-3-031-48751-4_17

7. Duan Y.P., Yang Y.Zh., Cao Y., et al. Path planning optimization for swine manure-cleaning robots through enhanced slime mold algorithm with cellular automata. *Animal Science Journal*. 2024;95(1):e13992. https://doi.org/10.1111/asj.13992
8. Hassani I., Maalej I., Rekik Ch. Robot Path Planning with Avoiding Obstacles in Known Environment Using Free Segments and Turning Points Algorithm. *Mathematical Problems in Engineering*. 2018;2018:2163278. http://doi.org/10.1155/2018/2163278
9. Zhao Y., Seong S.J., Fan Ch., et al. Robot Automatic Path Planning by Avoiding Obstacle using Double Deep Q Networks on the Testbed. In: *International Conference on Convergence Content (ICCC 2022), 21–23 December 2022, Jeju, South Korea.* 2023. P. 19–20.
10. Muhammad A., Ali M.A.H., Turaev Sh., et al. A Generalized Laser Simulator Algorithm for Mobile Robot Path Planning with Obstacle Avoidance. *Sensors*. 2022;22(21):8177. https://doi.org/10.3390/s22218177
11. Daniel K., Nash A., Koenig S., et al. Theta*: Any-Angle Path Planning on Grids. *Journal of Artificial Intelligence Research*. 2010;39:533–579. https://doi.org/10.1613/jair.2994
12. Chung S.H., Sah Bh., Lee J. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*. 2020;123:105004. https://doi.org/10.1016/j.cor.2020.105004
13. Cannon J., Rose K., Ruml W. Real-time motion planning with dynamic obstacles. *Proceedings of the International Symposium on Combinatorial Search*. 2012;3(1):33–40. https://doi.org/10.1609/socs.v3i1.18249
14. Саллам М., Макаровских Т.А. Проектирование оптимальной траектории проведения мониторинга объектов. *Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика*. 2024;13(3):32–46. https://doi.org/10.14529/cmse240302
Sallam M., Makarovskikh T.A. Designing the Optimal Trajectory for Monitoring Objects. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering*. 2024;13(3):32–46. (In Russ.). https://doi.org/10.14529/cmse240302
15. Dai Y., Lv W., Li Sh., et al. Improving the Lifelong Planning A-star algorithm to satisfy path planning for space truss cellular robots with dynamic obstacles. *Robotica*. 2025;43(4):1243–1257. http://doi.org/10.1017/S0263574725000256
16. Chatzisavvas A., Dossis M., Dasygenis M. Optimizing Mobile Robot Navigation Based on A-Star Algorithm for Obstacle Avoidance in Smart Agriculture. *Electronics*. 2024;13(11):2057. https://doi.org/10.3390/electronics13112057
17. Makarovskikh T., Sallam M. Optimal Trajectory for Monitoring Objects with Obstacles. In: *Mathematical Optimization Theory and Operations Research (MOTOR 2025), 07–11 July 2025, Novosibirsk, Russia.* Cham: Springer; 2025. P. 255–269. https://doi.org/10.1007/978-3-031-97077-1_18
18. Михайлов И.Е. Практическое сравнение алгоритма А* с алгоритмом волновой трассировки (алгоритмом Ли) по быстродействию. *Наука, техника и образование*. 2016;(1):47–49.
19. Dorigo M., Gambardella L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*. 1997;1(1):53–66. https://doi.org/10.1109/4235.585892
20. Dorigo M., Stützle Th. *Ant Colony Optimization*. Cambridge: MIT Press; 2004. 319 p.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Макаровских Татьяна Анатольевна, доктор физико-математических наук, доцент, профессор центра «ВиртУм», Южно-Уральский государственный университет, Челябинск, Российская Федерация.
e-mail: makarovskikh.t.a@susu.ru
ORCID: 0000-0002-3656-9632

Tatiana A. Makarovskikh, Doctor of Physical and Mathematical Sciences, Associated Professof, Professor of Center "VirtUm", South Ural State University, Chelyabinsk, the Russian Federation.

Саллам Мохамед Эльсайед Хуссейн Мохамед Мохамед, аспирант кафедры системного программирования, Южно-Уральский государственный университет, Челябинск, Российская Федерация.
e-mail: mohamedslam2000@yahoo.com
ORCID: 0000-0002-8703-4523

Mohamed E.H.M.M. Sallam, Postgraduate, Department of Computer Science, South Ural State University, Chelyabinsk, the Russian Federation.

Статья поступила в редакцию 30.05.2026; одобрена после рецензирования 06.08.2026; принята к публикации 13.08.2026.

The article was submitted 30.05.2026; approved after reviewing 06.08.2026; accepted for publication 13.08.2026.