Надежность, том 25, №3, 2025 / Dependability 3, 2025
Организация производства на транспорте

УДК: 656.022
Оригинальная статья
https://doi.org/10.21683/1729-2646-2025-25-3-60-67

Использование алгоритма поиска в ширину при решении задач пространственного развития инфраструктуры наземного транспорта

Using a breadth-first search algorithm in the process of spatial development of land transportation infrastructure

Кузьмин Д.В.[1]*
Kuzmin D.V.[1]*
Российский университет транспорта, Российская Федерация, Москва
Russian University of Transport, Russian Federation, Moscow
**kuzminmiit@yandex.ru*

Кузьмин Д.В.

Резюме. Цель. *Рассмотреть вопрос применимости алгоритма поиска пути в ширину для решения задач пространственного развития линейных объектов наземной транспортной инфраструктуры*. **Методы.** *В статье применяется алгоритм поиска пути в графе – Поиск в ширину (Breadth-First Search, BFS), широко используемый для различных прикладных задач теории графов, в том числе трассирования и планирования пути. С данным алгоритмом проведен ряд простых экспериментов с целью определения количественных показателей его асимптотической сложности, т.е. количества выполняемых операций и времени выполнения алгоритма. Серия экспериментов имеет различную конфигурацию, определяемую направленностью поиска (однонаправленный и двунаправленны) и способом прохода ячеек (прямой и смешанный).* **Выводы.** *Эксперименты с различной реализацией алгоритма показывают, что двунаправленный поиск может существенным образом сократить количество выполняемых операций и время поиска. Так количество операций при двунаправленном поиске меньше в 2,75 раза при прямом и в 2,78 раза при смешанном (прямом и диагональном) проходе ячеек. Более того, сделан вывод, что применение двунаправленной реализации алгоритма имеет свою область эффективного использования. Во-первых, двунаправленный поиск эффективен в графах с высокой степень ветвления. Сокращение количества операций при двунаправленном поиске в условиях лабиринта составляет 57,07%, а сокращение времени при этой же конфигурации эксперимента 76,92%, по сравнению с однонаправленной реализацией поиска. В среде, представляющей собой коридор и, следовательно, характеризующейся слабым ветвлением, разница в количестве выполняемых операций между двунаправленным и однонаправленным поиском составила 1,06%, а время выполнения осталось неизменным. Во-вторых, эффективность алгоритма существенно снижается при сложной структуре графа. В-третьих, для использования такой реализации необходимо иметь четкое понимание, что путь между стартовым и целевым узлом существует.*

Abstract. Aim. *To examine the applicability of the breadth-first path searching algorithm for spatial development of linear land transportation infrastructure facilities*. **Methods.** *The paper uses Breadth-First Searching, a graph path searching algorithm that is widely used as part of various graph theory applications, including path tracing and path planning. A number of simple experiments were carried out with this algorithm in order to determine the quantitative indicators of its asymptotic complexity, i.e., the number of performed operations and the algorithm execution time. The series of experiments has a different structure that is defined by the search direction (unidirectional and bidirectional) and the method of cell scanning (direct and mixed).* **Conclusion.** *Experiments involving various implementations of the algorithm show that bidirectional search can significantly reduce the number of performed operations and the search time. Thus, the number of operations for bidirectional search is 2.75 times less for direct and 2.78 times less for mixed (direct and diagonal) cell scanning. Moreover, it is concluded that the bidirectional implementation of the algorithm has its own scope of efficient use. First, bidirectional search is effective in highly-branched graphs. The number of operations for bidirectional maze search decreases 57.07%, while the time of the same experiment decreases 76.92% as compared to the unidirectional search. In a corridor environment that, by definition, has weak branching, the difference in the number of performed operations between bidirectional and unidirectional search was 1.06%, while the execution time remained the same. Secondly, the efficiency of the algorithm is significantly reduced when the graph structure is complex. Thirdly, using this implementation requires confidence in the fact that a path between the starting and target nodes exists.*

Ключевые слова: *алгоритмы поиска пути, трассирование, пространственное развитие транспортной инфраструктуры, транспортные системы, теория графов.*
Keywords: *pathfinding algorithms, tracing, spatial development of transportation infrastructure, transportation systems, graph theory.*

Для цитирования: *Кузьмин Д.В. Использование алгоритма поиска в ширину при решении задач пространственного развития инфраструктуры наземного транспорта // Надежность. 2025. №3. С. 60-67. https://doi.org/10.21683/1729-2646-2025-25-3-60-67*
For citation: *Kuzmin D.V. Using a breadth-first search algorithm in the process of spatial development of land transportation infrastructure. Dependability 2025;3: 60-67. https://doi.org/10.21683/1729-2646-2025-25-3-60-67*

Поступила: *15.04.2025* / **После доработки:** *21.04.2025* / **К печати:** *25.07.2025*
Received on: *15.04.2025* / **Revised on:** *21.04.2025* / **For printing:** *25.07.2025*

Введение

Пространственное развитие транспортной инфраструктуры является фундаментальной задачей организации работы и функционирования транспортных систем. Значительная часть подходов к решению данной задачи сводится к декомпозиции рассматриваемого полигона на отдельные территориальные единицы, совокупность которых, в дальнейшем рассматривается как граф. Совокупность рассматриваемых отдельных территориальных единиц, не имеющая разрывов и наложений, представляет собой растровую пространственную модель.

Под растровым моделированием пространственных данных понимают способ цифрового описания пространственных объектов и топологических отношений между ними. Описание выполняется с помощью регулярных и нерегулярных сеток, покрывающих рассматриваемый полигон. Сетка делит рассматриваемый полигон на дискретные ячейки – операционно-территориальные единицы (ОТЕ). При решении геоинформационных задач чаще используются регулярные (постоянные) сетки, в которых все ОТЕ имеют одинаковый размер, форму и т.д. Все ОТЕ содержат одинаковый набор параметров, характеризующих их пространственные свойства, т.н. атрибутивные данные, которые могут содержать информацию о топологических, гидрографических и антропогеографических и других свойствах пространства. Соединение центроидов ОТЕ образует граф.

Алгоритмы поиска пути в графах, такие как A*, BFS, Дейкстры, являются базовыми инструментами проектирования пространственного развития транспортной инфраструктуры, определения топологии транспортных сетей и комплексной организации работы транспортных систем.

1. Обзор источников

Растровые сетки широко используются для решения геоинформационных задач поиска пути, например для трассировок трубопроводов [1, 2], автомобильных дорог [3, 4], железных дорог [3, 5], линий электропередач [6, 7]. Логика использования растровых сеток в этом случае заключается в присвоении всем ОТЕ одинакового набора пространственных данных т.н. атрибутов. Анализируя различными методами распределение значений атрибутов ОТЕ, исследователь может определить наилучший маршрут в рамках существующей инфраструктуры или оптимальное пространственное развитие трассы.

Растровые модели поиска имеют слабые стороны. По причине графового рассмотрения пространства (абстракция узлов и связей) возникают неизбежные искажения трассировки, например, трасса может оказаться избыточно длинной или содержать множество геометрических несовершенств. Это приводит к получению нереалистичных результатов определения пространственного развития трассы. Подробно данная проблематика рассмотрена в работе [8]. В частности отмечается, что одномерный граф является приближением к бесконечному числу трассировок в рамках рассматриваемой области пространства, поэтому неизбежны фактические расхождения между расчетной и реальной трассой. Путь, прокладываемый в растровом пространстве, имеет дискретный шаг, определяемый в том числе геометрическими свойствами формы ОТЕ. Это приводит к неизбежным удлинениям и геометрическим несовершенствам трассы. Особенное явно эти негативные эффекты проявляются в неоднородных растровых пространствах. По причине существенной разницы количественных показателей атрибутивных данных путь подвержен частым изменениям направления, тогда как в условиях однородности растра данные искажения менее выражены. [9]

2. Постановка задачи

Задача поиска пути является фундаментальной задачей теории графов, имеющей множество практических приложений. Конфигурация графа определяет возможность использования того или иного подхода в решении задачи поиска пути. Граф может иметь различные свойства: быть взвешенным или невзвешенным, ориентированным или неориентированным, регулярным и нерегулярным и т.д.

Имеется регулярный взвешенный граф:

$$G=(V,E,w),$$

где V – множество вершин;

E – множество ребер (пар вершин);

w: $E \rightarrow R^{+}$ – функция веса, назначающая каждому

ребру положительное число (вес). Начальная вершина $s \in V$, конечная вершина $t \in V$.

Необходимо найти путь $p=(v_0, v_1, \ldots, v_k)$ в графе G таким образом, чтобы путь начинался в начальной вершине $v_0=s$, путь заканчивался в конечной вершине $v_k=t$, каждая пара соседних вершин в пути соединена ребром в графе $(v_i, v_{i+1})\in E$ для всех $i = 0, 1, \ldots, k\text{-}1$. При этом необходимо минимизировать целевую функцию – суммарный вес пути:

$$\Sigma_{(i=0)}^{(k-1)} w(v_i, v_{(i+1)}) \rightarrow min.$$

В контексте задачи пространственного развития транспортной инфраструктуры, когда дискретное пространство образуется за счет множества отдельных территориальных единиц, целесообразно присваивать вес не ребру, а вершине. Такой граф называется взвешенным по вершинам (vertex-weight graph). В этом случае функция веса, отображающая каждую вершину w: $V \rightarrow R$ (или положительное число R^+).

С целью корректной оценки совокупности свойств полигона, отдельные территориальные единицы должны иметь постоянную форму и размер, следовательно, образованный граф будет регулярным. Регулярность графа заключается в том, что каждая вершина имеет одинаковую степень (одинаковое количество соседей). Степень вершины deg(v) означает количество ребер инцидентных вершине v. Граф G является k-регулярным, если существует такое целое число, что $\forall v\in V$:deg(v)=k, то есть для каждой вершины v множества вершин V функция deg(v), определяющая степень вершины, равна k.

3. Методы

Существует множество алгоритмов [10, 11] по поиску кратчайшего или наименее затратного пути. Большинство алгоритмов поиска пути предназначены для работы с произвольными графами, однако, регулярное дискретное пространство формирует граф в виде решетки различной геометрической формы. В зависимости от конфигурации условий решаемой задачи (ограничений поиска, равномерности стоимости пространства, вычислительных мощностей и т.д.) каждый из представленных алгоритмов имеет свою сферу эффективного использования.

Алгоритм поиска в ширину (Breadth-First Search, BFS) систематически исследует граф, начиная с заданной начальной вершины и посещая все соседние вершины, прежде чем перейти к их соседям. Первое описание алгоритма встречается в работе Эдварда Ф. Мура от 1959 г. [12]. В статье автор использовал алгоритм для поиска кратчайшего пути в лабиринте. В значительной степени данный алгоритм рассмотрен голландским программистом и математиком Эдсгером В. Дейкстрой в 1959 г. в работе [13]. Необходимо отметить, что в историческом контексте конец 1950-х годов является началом периода активного изучения проблем обхода графов. В дальнейшем, подробно, этот и другие упомянутые в данном исследовании алгоритмы были рассмотрены множеством авторских коллективов математиков, программистов и инженеров, среди трудов которых можно выделить работы [14, 15].

Пусть имеется неориентированный граф $G=(V,E)$, где V – множество вершин, E – множество ребер $E\geq V\times V$. Также имеются начальная $s\in V$ и конечная вершины $t\in V$.

Необходимо найти путь $P=(v_0, v_1, \ldots, v_k)$ от s к t, где $v_0=s$, $v_k=t$ и $(v_i,v_{i+1})\in E$ для всех i=0, 1, …, k-1. Если пути не существует, $P=\varnothing$. Дополнительно введем следующие определения:

Множество соседей вершины u: $N(u) = \{v\in V \mid (u,v)\in E\}$;

Q: Очередь, организованная по принципу FIFO (первый пришел – первый обслужен);

Vis: Множество посещенных вершин;

Par: $V\rightarrow V\cup\{\text{null}\}$: Функция «родителя», $Par(v)$ возвращает вершину, из которой пришли в v.

На этапе инициализации формируется пустая очередь для хранения вершин, которые необходимо посетить $Q\leftarrow\varnothing$ и изначально пустое множество для хранения уже посещенных вершин $Vis\leftarrow\varnothing$. Задается начальное состояние переменной $Par(v)$ для всех вершин v множества V: $Par(v)\leftarrow$null, $\forall v\in V$, то есть инициализируется указатель (ссылка) на родителя вершины, в данном случае на этапе инициализации вершины родителя не имеют $Par(v)\leftarrow$null, $\forall v\in V$. Последнее необходимо для отслеживания структуры графа и восстановления пути.

На начальном этапе поиска стартовая вершина s добавляется в конец очереди Q, т.е. $Q.enqueue(s)$ и во множество посещенных вершин Vis, т.е. $Vis\leftarrow Vis\cup\{s\}$. При этом, так как вершина s является стартовой, она не имеет родителя $Par(s)\leftarrow$null.

Основной цикл обхода графа выполняется до тех пор, пока очередь Q не станет пустой Q≠Ø. Первая вершина u извлекается из очереди Q, т.е. $u\leftarrow Q.dequeue()$ и если данная вершина является целевой $u=t$, то основной цикл завершается и алгоритм переходит на этап восстановления пути, описанный ниже. Если же $u\neq t$, то для каждой соседствующей с u вершине v, т.е. $v\in N(u)$ выполняется следующий цикл. Если данная вершина v не принадлежит множеству посещенных вершин Vis, т.е. $v\notin Vis$, то она добавляется в конец очереди $Q.enqueue(v)$ для дальнейшей обработки, после чего выполняется ее добавление в данное множество $Vis\leftarrow Vis\cup\{v\}$. В заключении данного цикла устанавливается связь v с родительской вершиной u, т.е. $Par(v)\leftarrow u$.

На заключительном этапе работы алгоритма выполняются операции по восстановлению пути. Если целевая вершина t не имеет родителя $Par(t)$=null, то, следовательно, пути к ней не существует $P\leftarrow\varnothing$. В противном случае инициализируется список $P\leftarrow[]$ для хранения вершин пути. При программной реализации алгоритма целесообразно реализовывать данную структуру в виде списка, в который будут добавляться вершины начиная с целевой вершины t. Выбор списка, как варианта структу-

ры для хранения и организации, обусловлен спецификой добавления новых данных в список *P*, речь о данной специфике пойдет ниже.

Переменной *current* которая является начальной точкой для восстановления пути, присваивается значение вершины *t*, т.е. *current*←*t*. Вершина *current* добавляется в начало списка *P*, *P.insert*(0, *current*) и осуществляется переход к родителю текущей вершины *current*←Par(*current*). Цикл выполняется до тех пор, пока выполняется условие *current*≠null, т.е. восстановление пути выполняется до тех пор, пока не будет достигнута вершина без родителя.

Блок-схема алгоритма поиска пути в ширину представлена на рис. 1.

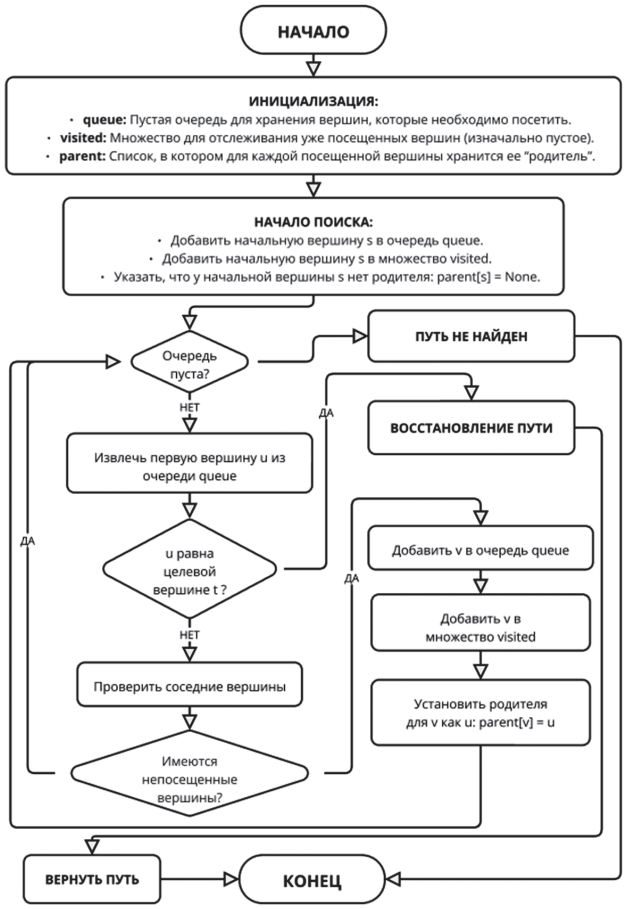
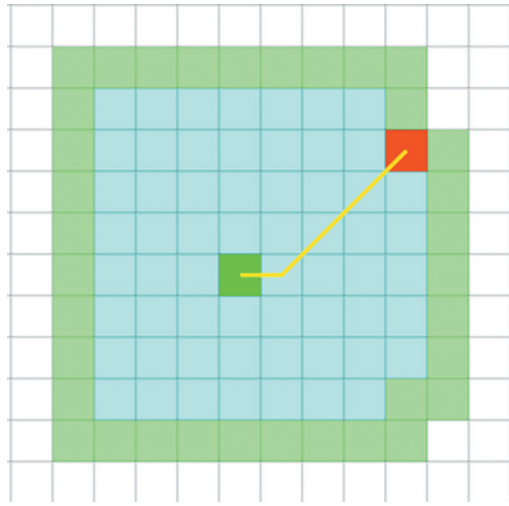


Рис. 1. Блок-схема алгоритма поиска пути в ширину

4. Обсуждение и результаты

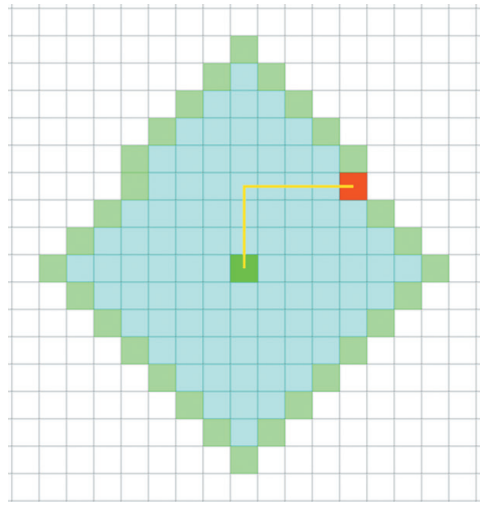
С алгоритмом была проведена серия простых экспериментов, которые конфигурировались вариантом прохода ячейки (прямой и смешанный) и направлением поиска. Среда поиска, в рамках которой осуществлялся эксперимент, с целью сравнения асимптотической сложности алгоритмов, была беспрепятственной и имела постоянные рейтинги ячеек их форму и размер. Расположение стартовой и целевой вершины также оставалось постоянным. Условные координаты стартовой вершины (0, 0), целевой (4, 3).

Визуализация результата экспериментов поиска пути алгоритмом поиска в ширину представлена на рис. 2.



а)

Длина пути – 5,24 ед., количество операций: 159, время выполнения алгоритма: 0,3 мс.



б)

Длина пути – 7 ед., количество операций: 215, время выполнения алгоритма: 0,5 мс.

Рис. 2. Визуализация результата поиска пути алгоритмом поиска в ширину (Breadth-First Search, BFS): а) реализация с возможностью диагонального и прямого прохода ячеек; б) реализация с прямым проходом ячеек. Зеленая и красная ячейки – соответственно стартовый и целевой узлы *s* и *t*. Ячейки светло-зеленого цвета – очередь Q_s; Ячейки голубого цвета –множество посещенных вершин Vis_s.

Наиболее эффективной является программная реализация вышеописанного алгоритма поиска.

Если известно, что путь между вершинами *s* и *t* существует, то данный алгоритм можно реализовать двунаправленным, т.е. поиск пути будет выполняться одновременно из стартовой и целевой вершины. В этом случае решение будет найдено быстрее, чем при однонаправленном поиске, так как волны поиска будут распространятся одновременно из двух точек.

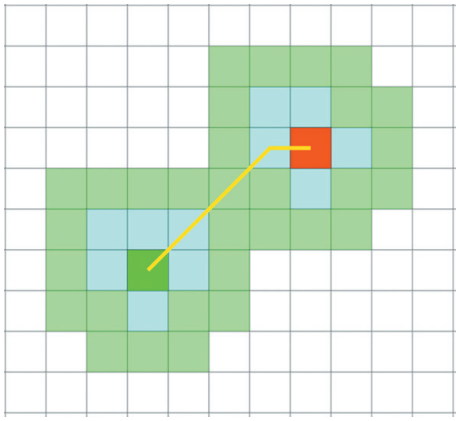
Помимо этого, пространственная сложность графа так же сократится, так как поиском будет покрыта более компактная часть графа, что в свою очередь потребует меньшей вычислительной памяти при программной реализации алгоритма.

В случае двунаправленной реализации, базовый принцип работы алгоритма не изменится, однако, на этапе инициализации задублируются следующие структуры данных: очереди для хранения вершин, необходимых к посещению, для стартовой и целевой вершин Q_t и Q_s, множества посещенных вершин Vis_s и Vis_t, списки родительских вершин для поиска из s и t.

Принципиальным отличием от однонаправленного поиска является условие остановки выполнения алгоритма. Алгоритм прекращает свою работу при обнаружении вершины, посещенной обеими волнами поиска $\exists v \in V$: $v \in Vis_s \wedge v \in Vis_t$, т.е. существует хотя бы одна вершина v графа V, принадлежащая одновременно множествам Vis_s и Vis_t. При этом важным условием реализации является равномерность распространения волн поиска, чередование поиска шагов от точек s и t.

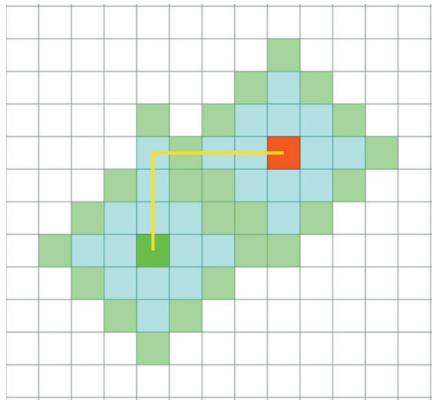
Пример реализации двунаправленного поиска пути алгоритмом поиска в ширину приведен на рис. 3.

Проведенные эксперименты с различной реализацией алгоритма показывают, что двунаправленный поиск может существенным образом сократить количество выполняемых операций и время поиска. Так, количество операций при двунаправленном поиске меньше в 2,75 раза при прямом и в 2,78 раза при смешанном (прямом и диагональном) проходе ячеек. Инфографика с результатами проведенных экспериментов представлена на рис. 4 и 5.



а)

Длина пути – 5,24 ед., количество операций: 57, время выполнения алгоритма: 0,3 мс.



б)

Длина пути – 7 ед., количество операций: 78, время выполнения алгоритма: 0,4 мс.

Рис. 3. Визуализация результата двунаправленного поиска пути алгоритмом поиска в ширину (Breadth-First Search, BFS): а) реализация с возможностью диагонального и прямого прохода ячеек; б) реализация с прямым проходом ячеек. Зеленая и красная ячейки – соответственно стартовый и целевой узлы s и t. Ячейки светло-зеленого цвета – очереди Q_t и Q_s; ячейки голубого цвета – множества посещенных вершин Vis_s и Vis_t.

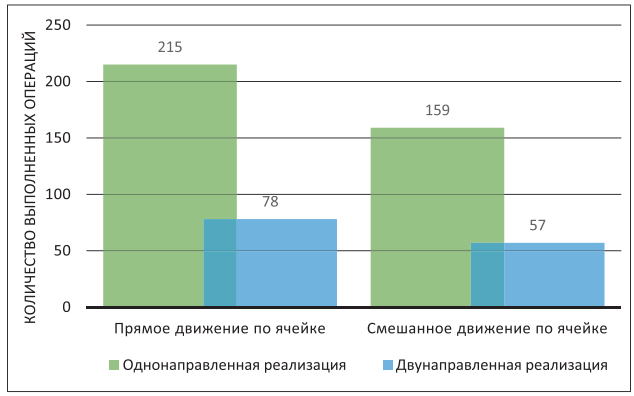


Рис. 4. Результаты экспериментов с различными вариантами реализации алгоритмом поиска в ширину (Breadth-First Search, BFS), по показателю «Количество выполненных операций»

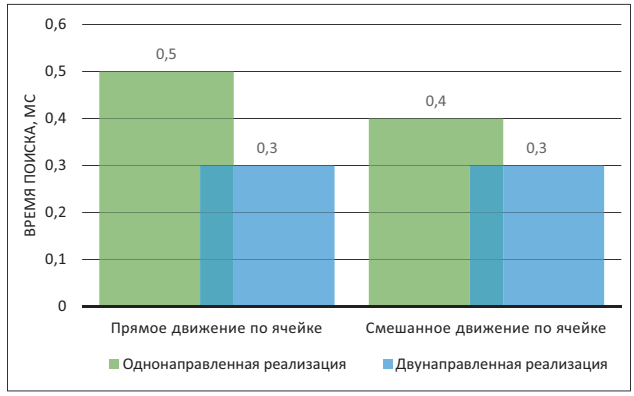
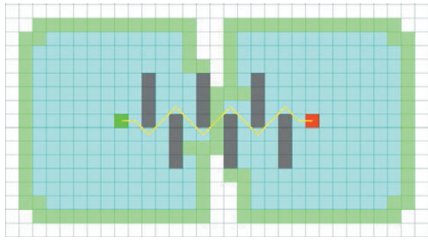
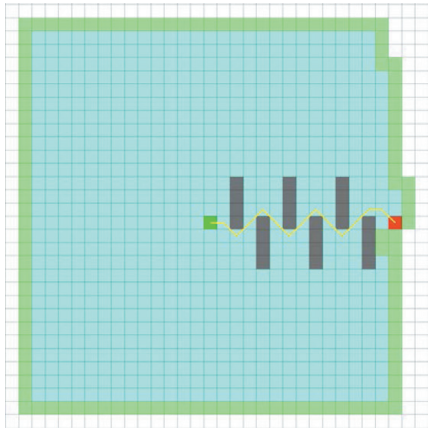


Рис. 5. Результаты экспериментов с различными вариантами реализации алгоритмом поиска в ширину (Breadth-First Search, BFS), по показателю «Время поиска»

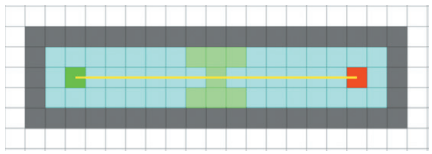
Однако стоит отметить, что применение двунаправленной реализации алгоритма имеет свою область эффективного использования. Двунаправленный поиск эффективен в графах с высокой степень ветвления. В слабоветвящемся графе (с малым количеством ребер связи) область поиска будет расширяться линейно, оба поисковых процесса будут проходить одинаковое или близкое расстояние. Наиболее контрастным примером



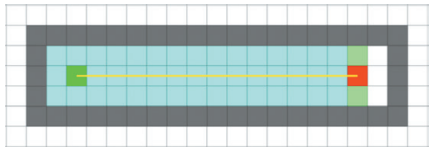
а) Длина пути – 18,97 ед., количество операций: 677, время выполнения алгоритма: 0,3 мс



б) Длина пути – 18,97 ед., количество операций: 1577, время выполнения алгоритма: 1,3 мс



в) Длина пути – 14 ед., количество операций: 95, время выполнения алгоритма: 0,5 мс



г) Длина пути – 14 ед., количество операций: 94, время выполнения алгоритма: 0,5 мс

Рис. 6. Визуализация результата экспериментов поиска пути алгоритмом поиска в ширину (Breadth-First Search, BFS) с различным ветвлением графа: а) реализация двунаправленного поиска в лабиринте; б) реализация однонаправленного поиска в лабиринте; в) реализация двунаправленного поиска в коридоре; г) реализация однонаправленного поиска в коридоре. Зеленая и красная ячейки – соответственно стартовый и целевой узлы s и t. Ячейки светло-зеленого цвета – очереди Q_t и Q_s; ячейки голубого цвета – множества посещенных вершин Vis_s и Vis_t.

является пример поиска пути в лабиринте. Если лабиринт имеет множество путей (граф лабиринта сильно ветвится), то эффективность двустороннего поиска выше, так как выше вероятность более скорой встречи двух фронтов поиска. Если же лабиринт представляет собой прямой коридор, то двусторонний поиск не даст большого преимущества.

Докажем данное утверждение, проведя серию экспериментов. Во всех экспериментах стартовая и целевая вершины имеют постоянное условное координатное положение (0, 0) и (14, 0) соответственно. В первом случае среда имеет препятствия, т.е. является условным лабиринтом, во втором случае среда беспрепятственна и представляет собой прямой коридор. Среда поиска постоянна по размеру ячеек, их форме и рейтингу. Во всех конфигурациях эксперимента используем диагонально-прямой (смешанный) проход ячеек. Визуализация эксперимента приведена на рис. 6.

Результаты эксперимента наглядно подтверждают гипотезу о том, что двунаправленный поиск более эффективен в графах с сильным ветвлением (лабиринт), чем в графах со слабым ветвлением (коридор). Количественное сравнение результатов эксперимента обобщим в табл. 1.

Сокращение количества операций при двунаправленном поиске в условиях лабиринта составляет 57,07%, а сокращение времени при этой же конфигурации эксперимента 76,92%, по сравнению с однонаправленной реализацией поиска. В среде, представляющей собой коридор и, следовательно, характеризующейся слабым ветвлением разница в количестве выполняемых операций между двунаправленным и однонаправленным поиском составила 1,06%, а время выполнения осталось неизменным. Из чего следует вывод, подтверждающий гипотезу о эффективности двунаправленного поиска в графах с сильным ветвлением. В условиях сильного ветвления двунаправленный поиск значительно сокращает область поиска и время выполнения алгоритма, тогда как в условиях слабого ветвления разница количественных оценок экспериментов практически отсутствует.

Более того, эффективность алгоритма существенно снижается при сложной структуре графа. И наконец, для использования такой реализации, необходимо иметь четкое понимание, что путь между стартовым и целевым узлом s и t существует.

Сложность алгоритма оценивается по двум параметрам: временная и пространственная сложность.

Табл. 1. Количественные оценки результатов проведенных экспериментов

Конфигурация эксперимента		Количество выполненных операций	Время выполнения алгоритма, мс
Характеристика среды поиска	**Направление поиска**		
Лабиринт (сильное ветвление графа)	Однонаправленный	1577	1,3
	Двунаправленный	677	0,3
Коридор (слабое ветвление графа)	Однонаправленный	94	0,5
	Двунаправленный	95	0,5

Табл. 2. Сравнение асимптотической сложности алгоритмов Поиска в ширину и Дейкстры по результатам проведенных экспериментов, показатель «Количество выполняемых операций»

Конфигурация поиска	Поиск в ширину	Дейкстры	Сравнение Дейкстра относительно Поиска в ширину
Однонаправленный, прямой	215	232	+7,9%
Однонаправленный, смешанный	159	205	+28,9%
Двунаправленный, прямой	78	86	+10,3%
Двунаправленный, смешанный	57	59	+3,5%

Временная сложность отражает зависимость скорости роста времени выполнения алгоритма от размера входных данных. Пространственная сложность отражает зависимость объема памяти от размера входных данных.

Данные характеристики выражаются O-нотацией, показателем, который отражает верхние границы временной или пространственной сложности алгоритма. Можно выделить несколько типов O-нотаций: $O(1)$ – константная сложность, алгоритм выполняется за фиксированное время, независимо от размера входных данных; $O(n)$ – линейная сложность, пропорциональное размеру входных данных; $O(n^2)$ – квадратичная сложность, время выполнения пропорционально квадрату размера входных данных и проч.

Временную сложность алгоритма поиска в ширину можно оценить следующим образом. Пусть V – количество вершин, E – количество ребер, тогда в лучшем случае алгоритм посещает каждую вершину графа один раз, т.е. это требует $O(V)$ времени. Каждая посещенная вершина имеет соседей, которые необходимо добавить в очередь для посещения и посетить. Это означает, что каждое ребро графа, будет рассмотрено минимум один раз в ориентированном и дважды в неориентированном графе. Это займет $O(E)$ времени. Так как каждая вершина и ребро посещается минимум 1 раз, то общая сложность составит $O(V + E)$. Другими словами, время выполнение алгоритма линейно зависит от количества вершин и ребер в графе.

Пространственную сложность данного алгоритма необходимо оценивать по следующим процессным составляющим, потребляющим память: очередь рассмотрения вершин, список родительских вершин, множество посещенных вершин. Рассматривая в совокупности перечисленные составляющие пространственную сложность можно оценить как $O(V)$, т.е. объем занимаемой памяти, используемый алгоритмом, линейно зависит количества вершин в графе.

Данный алгоритм в большей степени подходит для поиска кратчайшего пути (минимального количества ребер) в невзвешенном графе. Несмотря на то, что он может быть адаптирован для поиска по взвешенному графу, его использование не гарантирует нахождение кратчайшего пути по весу, в отличие, например, от алгоритма Дейкстры.

Сравнение данных алгоритмов в аспекте асимптотической сложности является предметом отдельной публикации. Исследование на эту тему уже проведено, но в силу тематики данной статьи, а также сложности и объемности данного вопроса, автор считает нецелесообразным проведение полного критического анализ алгоритма Дейкстры в рамках данной публикации. Тем не менее, основные количественные результаты сравнения обобщены в табл. 2.

Представленный алгоритм, а также выводы и рекомендации, полученные в результате проведения экспериментов, представляют практическую ценность, так как могут являться инструментом обоснования и принятия взвешенных управленческих решений при решении задач пространственного развития линейной инфраструктуры наземного транспорта. Их использование позволяет эффективно решать задачи оптимизации трассирования новых путей в части проектирования с учетом капитальных и эксплуатационных затрат, обхода препятствий, учета ограничений и т.д.

Список литературы

1. Automatic Pipeline Route Design with Multi-Criteria Evaluation Based on Least-Cost Path Analysis and Line-Based Cartographic Simplification: A Case Study of the Mus Project in Turkey. URL: https://www.mdpi.com/2220-9964/8/4/173 (дата обращения: 23.05.2025).

2. Kang J.Y., Lee B. Optimisation of pipeline route in the presence of obstacles based on a least cost path algorithm and laplacian smoothing // International Journal of Naval Architecture and Ocean Engineering. 2017. Vol. 9.

3. Scaparra M.P., Church R.L., Medrano F.A. Corridor location: the multi-gateway shortest path model // Journal of Geographical Systems. 2014. Vol. 16. No. 3. Pp. 287-309.

4. Yu C., Lee J., Munro-Stasiuk M. Extensions to least-cost path algorithms for roadway planning // International Journal of Geographical Information Science – GIS. 2003. Vol. 17.

5. Jamali A.A., Esmailian A., Mokhtarisabet S. et al. Path selection by topographic analysis: vector re-classification versus raster fuzzification as spatial multi-criteria using cost-path // Spatial Information Research. 2023. T. 31.

6. Stefano B., Davide G., Francesco O. Routing of power lines through least-cost path analysis and multicriteria evaluation to minimise environmental impacts // Environmental Impact Assessment Review. 2011. Vol. 31. No. 3. Pp. 234-239.

7. Monteiro C., Ramírez-Rosado I., Miranda V. et al. GIS Spatial Analysis Applied to Electric Line Routing Optimization // Power Delivery, IEEE Transactions on. 2005. Vol. 20. Pp. 934-942.

8. Antikainen H. Comparison of Different Strategies for Determining Raster-Based Least-Cost Paths with a Minimum Amount of Distortion // Transactions in GIS. 2013. Vol. 17.

9. C. Dana T. Propagating radial waves of travel cost in a grid // International Journal of Geographical Information Science. 2010. Vol. 24(9). Pp. 1391-1413.

10. Ahuja R., Mehlhorn K., Orlin J. et al. Faster Algorithms for the Shortest Path Problem // J. ACM. 1990. Vol. 37. Pp. 213-223.

11. Wayahdi M., Ginting S., Syahputra D. Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path // International Journal of Advances in Data and Information Systems. 2021. Vol. 2. Pp. 45-52.

12. Moore E.F. The Shortest Path through a Maze // In Proc. International Symposium on the Theory of Switching, Part II. Harvard University Press, 1959.

13. Dijkstra E.W. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1. No. 1. Pp. 269-271.

14. Cormen T.H., Leiserson C.E., Rivest R.L. et al. Introduction to Algorithms. MIT Press and McGraw-Hill, 2001. 1216 p.

15. Sedgewick R., Wayne K. Algorithms. Addison-Wesley Professional, 2011. 971 p.

References

1. Automatic Pipeline Route Design with Multi-Criteria Evaluation Based on Least-Cost Path Analysis and Line-Based Cartographic Simplification: A Case Study of the Mus Project in Turkey. (accessed 23.05.2025). Available at: https://www.mdpi.com/2220-9964/8/4/173.

2. Kang J.Y., Lee B. Optimisation of pipeline route in the presence of obstacles based on a least cost path algorithm and laplacian smoothing. *International Journal of Naval Architecture and Ocean Engineering* 2017;9.

3. Scaparra M.P., Church R.L., Medrano F.A. Corridor location: the multi-gateway shortest path model. *Journal of Geographical Systems* 2014;16(3):287-309.

4. Yu C., Lee J., Munro-Stasiuk M. Extensions to least-cost path algorithms for roadway planning. *International Journal of Geographical Information Science – GIS* 2003;17.

5. Jamali A.A., Esmailian A., Mokhtarisabet S. et al. Path selection by topographic analysis: vector re-classification versus raster fuzzification as spatial multi-criteria using cost-path. *Spatial Information Research* 2023;31.

6. Stefano B., Davide G., Francesco O. Routing of power lines through least-cost path analysis and multicriteria evaluation to minimise environmental impacts. *Environmental Impact Assessment Review* 2011;31(3):234-239.

7. Monteiro C., Ramírez-Rosado I., Miranda V. et al. GIS Spatial Analysis Applied to Electric Line Routing Optimization. *Power Delivery, IEEE Transactions on* 2005;20:934-942.

8. Antikainen H. Comparison of Different Strategies for Determining Raster-Based Least-Cost Paths with a Minimum Amount of Distortion. *Transactions in GIS* 2013;17.

9. C. Dana T. Propagating radial waves of travel cost in a grid. *International Journal of Geographical Information Science* 2010;24(9):1391-1413.

10. Ahuja R., Mehlhorn K., Orlin J. et al. Faster Algorithms for the Shortest Path Problem. *J. ACM* 1990;37:213-223.

11. Wayahdi M., Ginting S., Syahputra D. Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path. *International Journal of Advances in Data and Information Systems* 2021;2:45-52.

12. Moore E.F. The Shortest Path through a Maze. In: Proc. International Symposium on the Theory of Switching, Part II. Harvard University Press; 1959.

13. Dijkstra E.W. A note on two problems in connexion with graphs. *Numerische Mathematik* 1959;1(1):269-271.

14. Cormen T.H., Leiserson C.E., Rivest R.L. et al. Introduction to Algorithms. MIT Press and McGraw-Hill; 2001.

15. Sedgewick R., Wayne K. Algorithms. Addison-Wesley Professional; 2011.

Сведения об авторе

Кузьмин Дмитрий Владимирович, Россия, г. Москва, 127994, ул. Образцова 9/9, Федеральное государственное автономное образовательное учреждение высшего образования «Российский университет транспорта», доцент кафедры «Логистика и управление транспортными системами», кандидат технических наук, доцент, e-mail: kuzminmiit@yandex.ru

About the author

Dmitry V. Kuzmin, 9/9 Obraztsova st., Moscow, 127994, Russia, Russian University of Transport, Senior Lecturer, Department of Logistics and Transportation System Management, Candidate of Engineering, Associate Professor, e-mail: kuzminmiit@yandex.ru.

Вклад автора

Исследование проведено автором самостоятельно в полном объеме: выполнен анализ современной теоретической базы поиска пути в графах; сформулирована блок-схема алгоритма поиска в ширину; проведен обзор научных исследований по данной тематике; проведена серия экспериментов по поиску пути в простых графах; обобщены и проанализированы результаты полученных экспериментов; сформулированы рекомендации по практическому применению алгоритма для решения задач поиска пути.

Конфликт интересов

Автор заявляет об отсутствии конфликта интересов.