Оригинальная статья / Original article

https://doi.org/10.21869/2223-1560-2022-26-4-39-56

Исследование свойств алгоритма поиска в ширину для нахождения маршрута передвижения роботов

С. Г. Емельянов [1], М. В. Бобырь [1] ✉, А. Г. Крюков [1]

[1] Юго-Западный государственный университет
ул. 50 лет Октября, д. 94, г. Курск 305040, Российская Федерация

✉ e-mail: fregat_mn@rambler.ru

Резюме

Цель исследования. *Представленное в данной статье исследование нацелено на повышение быстродействия поиска пути для маршрута передвижения роботов. Научной новизной является полученная закономерность отношения времени и размеров поля.*

Методы. *Для нахождения пути в лабиринте использовались алгоритмы поиска в глубину и поиска в ширину, основой которых является цикличное прохождение смежных не посещенных ранее вершин графа. Быстродействие оценивается в скорости выполнения программного кода на подготовленных образцах. Научная новизна была получена за счет исследования влияния размеров карты на быстродействие алгоритмов поиска в глубину и ширину.*

Результаты. *Разработана программная реализация алгоритмов поиска в ширину и в глубину. В статье подробнее представлено описание алгоритма поиска в ширину в виде псевдо- и программного кодов, которые основываются на цикле while, где осуществляется обработка очереди проверяемых вершин графа. На основе оценки быстродействия найденного пути сделан вывод, что поиск в ширину не является быстрейшим. На основе оценки влияния различных факторов на скорость работы алгоритма сделан вывод, что увеличение размеров поля, уменьшение количества препятствий и расстояния между стартовой и финальной точками увеличивает время выполнения алгоритма.*

Заключение. *Был представлен алгоритм поиска в ширину и его программная реализация. В ходе экспериментальных исследований было установлено, что данный алгоритм по времени не является быстрейшим, но во всех тестах находил кратчайший путь. Также была получена закономерность $t_a = f(w, h)$ для подготовленных образцов искомого поля, которая выражается в зависимости времени выполнения алгоритма от длины и ширины поля. И можем заключить, что он применим для поиска пути передвижения роботов так как всегда находит кратчайший путь.*

Ключевые слова: *быстродействие алгоритма; поиск в ширину; поиск в глубину; робот; граф.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Финансирование: Работа выполнена при поддержке гранта РНФ 23-21-00071 – «Разработка модели компьютерного зрения для интеллектуальной навигации робототехнических систем, основанной на построении трехмерных сцен по картам глубин».

Для цитирования: Емельянов С. Г., Бобырь М. В., Крюков А. Г. Исследование свойств алгоритма поиска в ширину для нахождения маршрута передвижения роботов // Известия Юго-Западного государственного университета. 2022; 26(4): 39-56. https://doi.org/10.21869/2223-1560-2022-26-4-39-56.

Поступила в редакцию 17.08.2022 *Подписана в печать 06.10.2022* *Опубликована 14.10.2022*

Research of the Properties of the Breadth-First Search Algorithm for Finding the Movement Route of Robots

Sergei G. Emelianov [1], Maxim V. Bobyr [1] ✉, Aleksander G. Kryukov [1]

[1] Southwest State University
50 Let Oktyabrya str. 94, Kursk 305040, Russian Federation

✉ e-mail: fregat_mn@rambler.ru

Abstract

Purpose of research. *The research presented in this article is aimed at improving the speed of finding a path for the movement route of robots. The scientific novelty is the obtained correlation of time and field size.*

Methods. *To find the path in the maze, the depth-first search and breadth-first search algorithms were used, the basis of which is the cyclic processing of adjacent previously unvisited graph vertices. Performance is estimated in terms of the speed of program code execution on prepared samples. Scientific novelty was obtained by studying the influence of map sizes on the performance of depth-first and breadth-first search algorithms.*

Results. *A software implementation of breadth-first and depth-first search algorithms has been developed. The article provides a more detailed description of the breadth-first search algorithm in the form of pseudo and program codes, which are based on the while loop, where the queue of checked graph vertices is processed. Based on the evaluation of the speed of the found path, it was concluded that the breadth-first search is not the fastest. Based on the assessment of the influence of various factors on the speed of the algorithm, it was concluded that an increase in the size of the field, a decrease in the number of obstacles and a distance between the starting and final points increases the execution time of the algorithm.*

Conclusion. *The breadth-first search algorithm and its software implementation were presented. In the course of experimental studies, it was found that this algorithm is not the fastest in time, but in all tests, it found the shortest path. The correlation $t_a = f(w, h)$ was also obtained for the prepared samples of the desired field, which is expressed as the dependence of the algorithm execution time on the length and width of the field. And we can conclude that it is applicable for finding the movement path of robots, since it always finds the shortest path.*

Keywords: *algorithm performance; breadth-first search; depth-first search; robot; graph.*

Conflict of interest. *The authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

Funding: The work was supported by the grant of the Russian Science Foundation 23-21-00071 – "Development of a computer vision model for intelligent navigation of robotic systems based on the construction of three-dimensional scenes from depth maps".

For citation: Emelianov S. G., Bobyr M. V., Kryukov A. G. Research of the Properties of the Breadth-First Search Algorithm for Finding the Movement Route of Robots. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2022; 26(4): 39-56 (In Russ.). https://doi.org/ 10.21869/2223-1560-2022-26-4-39-56.

Received 17.08.2022 *Accepted 06.10.2022* *Published 14.10.2022*

***

Введение

Роботы используются в различных сферах человеческой жизнедеятельности, от бытовых дел до сложных производственных задач. Например, таких как планирование пути и поиск оптимального (кратчайшего, требующего наименьших затрат ресурсов/времени). Они стали одними из основных направлений разработки в области навигации роботов [1].

Существуют различные решения данной задачи. Используется подгруппа мягких алгоритмов – эволюционные вычисления, а именно методы роя частиц, в основе которых лежит копирование поведения животных или насекомых, например муравьев в одноименном алгоритме. Недостатком базового подходя является локальная минима (local minima), в результате которого робот может застрять в бесконечном цикле пытаясь объехать препятствие. В работе [2] представлены улучшения алгоритма, решающие эту проблему. Также ее решили при помощи нечеткой логики в [3]. В [4] и [5] смогли уменьшить время вычисления и затраты энергии соответственно, используя цифровые карты высот. Есть пример комбинирования двух моделей поведения: кукушки и летучей мыши [6].

В машинном обучении (обучение с подкреплением) были представлены алгоритмы, которые позволил улучшить такие характеристики как стабильность и скорость работы “actor-critic” алгоритма благодаря добавлению мультипоточности в [7]; переиспользованию данных буфера из [8] в [9]; введению ограничения области доверия в [10-11]. В [8] была решена проблема влияния шума на вычисления функции полезности Q за счет добавления еще одной нейронной сети.

Также развиваются и традиционные алгоритмы. Например, в работе [12], авторы исследовали задачу предотвращение роботом столкновений, для навигации мобильного робота используется комбинация жадного алгоритма и улучшенного алгоритма поиска в ширину. В [13] улучшенная версия используется в комбинации с RFID метками. А в [14] поиск в ширину применяется в прямоугольной системе координат.

В данной работе описывается применение алгоритма поиска в ширину. Выбор был сделан на основе оценки преимуществ и недостатков над другими способами. Алгоритмы, основанные на машинном обучении, не рассматриваются, так как требуют дополнительных обширных знаний. Остаются традиционные алгоритмы, среди которых

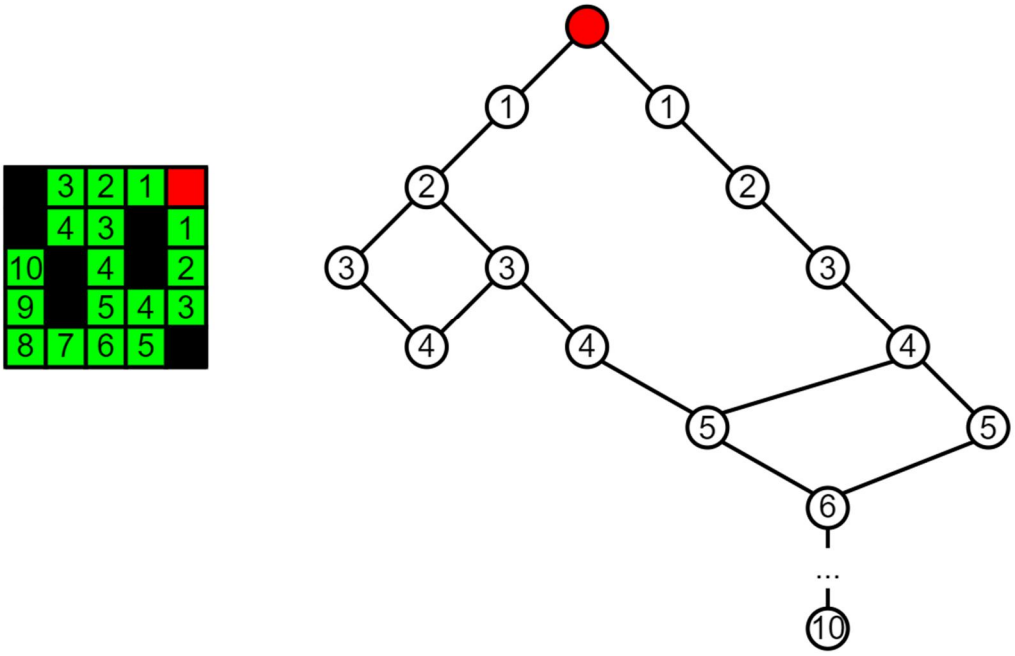
выделяются алгоритмы обхода графа как одни из самых надежных (путь будет найден, если он существует) и детерминированных [15-16]. К ним относятся алгоритм Дейкстры, A*, поиск в ширину и поиск в глубину, каждый из которых обладает своими достоинствами и недостатками. Они делятся на две подгруппы: информированные (алгоритм Дейкстры и A*) и неинформированные (поиск в ширину и глубину) методы поиска. Их отличие заключатся в том, что для информированных методов требуется наличие знаний о конкретной задаче, для неинформированных это не нужно [17]. Следовательно, для работы в режиме реального времени в неизвестной среде подходит вторая подгруппа. Алгоритм поиска в ширину обладает такими преимуществами, как способность находить самый короткий путь, время выполнения алгоритма в сравнении с аналогами [18], а также вышеупомянутые. В алгоритме Дейкстры на каждой итерации рассчитывается наименьшее расстояние от робота до искомой точки и принимается движение о дальнейших передвижениях. Следовательно, если робот не видит точку, например из-за препятствия, то алгоритм Дейкстры работать не будет. Для A* необходимо большее количество памяти робота, чем для поиска в ширину [18].

Помимо робототехники, алгоритм поиска в ширину часто используется для расчётов на графических чипах [19-20] и изучается возможность ускорения данного процесса за счёт оптимизации распараллеливания вычислительной нагрузки [20-23].

Материалы и методы

Для использования алгоритма поиска в ширину необходимо сперва представить карту в виде графа. Для этого существует несколько методов. Первый – это расставить вершины на заданном пространстве таким образом, чтобы они показывали места, в которых может находиться робот. Ребра, соединяющие вершины, являются путями возможного перемещения робота [16]. Такой способ применяется для прокладки маршрутов самолетов [24]. Второй метод, который используется в данной статье, основан на разбиении пространства на клетки [25-26] или иные геометрические фигуры, как например треугольники или шестигранники [16]. Реализация подобных алгоритмов реализуема при прогнозировании траектории режущего инструмента [27-29] или перемещения мобильного робота [30]. Второй способ был нами выбран, так как он обеспечивает автономную работу робота: человеку не нужно рисовать граф вручную на основе имеющейся карты. Зная начальную позицию, пролагаются возможные пути передвижения, из которых и формируется граф. Например, на рис. 1 (а) изображена карта случайной местности, разбитая на клетки разных цветов, где черным помечены непроходимые участки (препятствия), красным – начальная позиция, зеленым – клетки свободные для перемещения. На рис. 1

(б) изображено возможное представление той же самой карты в виде графа, подходящее для прохода алгоритма поиска в ширину. В данной работе рассматривается возможность хождения только в 4 направления: вправо, влево, вниз, вверх. Цифры отображают уровни прохода поиска в ширину.



а) **б**)

Рис. 1. Карта местности, представленная в виде клеток (**а**), представление данной карты в виде графа (**б**)

Fig. 1. Map of the area, presented in the form of cells (**а**), representation of this map in the form of a graph (**б**)

Основная концепция работы алгоритма поиска в ширину заключается в том, что проход по вершинам графа осуществляется горизонтально/поуровнево, то есть сперва исследуются вершины на одном уровне. Например, на рис. 1 (а), сперва проверяются вершины «1», затем «2» и т.д. Порядок посещения вершин основывается на принципе FIFO, который применяется в очереди. Псевдокод алгоритма поиска в ширину представлен на рис. 2. Вначале создается очередь из вершин на обработку, начиная со стартовой. Она помечается как посещенная. В данной работе это реализовано через дополнительный массив из посещенных вершин. Далее, пока очередь из верши не пуста, осуществляется проверка смежных вершин, той, которая только что была убрана из очереди. Смежные вершины проверяют, являются ли какая-то из них искомой вершиной или нет. Если является – алгоритм заканчивается и выдается путь до нужной точки, если нет – помечается как посещенная и добавляется в очередь для дальнейшего выполнения.

Algorithm 2: Breadth-first search for robotic path in a maze

```
Input: Nodes in the maze, Starting Point
BFS (G, s)
    let Q be queue
    Q.enqueue(s)

    mark s as visited
    while (Q is not empty)
       v  =  Q.dequeue( )

      for all neighbours w of v in Graph G
         if w is not visited
                Q.enqueue( w )
                mark w as visited
Output: Robotic Path from Starting Point to Goal Point
```

Рис. 2. Псевдокод алгоритма поиска в ширину [18]

Fig. 2. Breadth-first search algorithm pseudocode [18]

Пример прохода графа алгоритмом поиска в ширину представлен на рис. 3, где цифра перед точкой обозначает уровень поиска, цифра после – порядковый номер на данном уровне, следовательно последовательность проверки. Пунктирными стрелками помечены возможные пути прохода, которые, однако, не будут использованы из-за того, что данные вершины уже были посещены.

Для проведения сравнительного анализа был также использован алгоритм поиска в глубину. Его отличие от поиска в ширину заключается в том, что проход по вершинам графа осуществляется сперва в одном направлении по определенному пути. Если последняя вершина пути не является искомой точкой, то алгоритм возвращается назад до точки разветвления и обрабатывается другой путь.

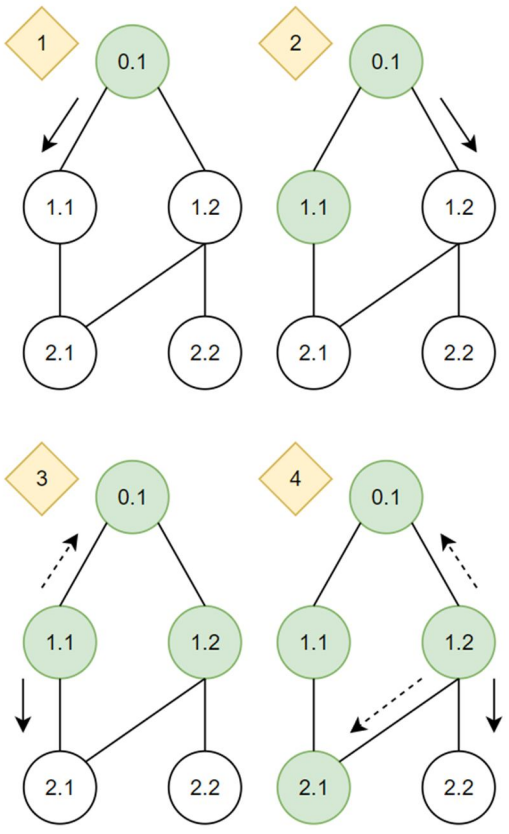


Рис. 3. Пример работы алгоритма поиска в ширину

Fig. 3. An example of the operation of the breadth-first search algorithm

Пример прохода графа алгоритмом поиска в глубину представлен на рис. 4, где цифра обозначает последовательность проверки. Шаг 4 представляет проход по всему правому пути сразу.

Реализация алгоритма поиска в ширину была осуществлена в виде комплекса программного обеспечения, реализованного в среде Microsoft Visual Studio 2019 на языке программирования C# и представлена на рис. 5 и рис. 6.

На рис. 5 представлены основные использованные структуры данных и переменные. На рис. 6 представлен цикл while осуществляющий поиск в ширину. В случае успешного нахождения решения, цикл вернет последовательность направлений движения от начальной до искомой точки. В противном случае, будет возвращено сообщение “no path”.

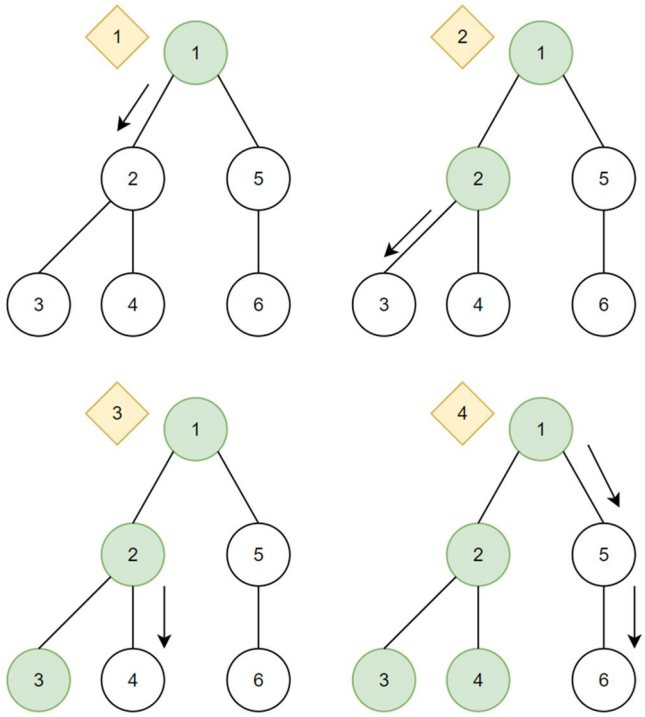


Рис. 4. Пример работы алгоритма поиска в глубину

Fig. 4. An example of the operation of the depth-first search algorithm

Примеры работы программного кода представлены на рис.7. В размер поля также включены его границы. То есть, поле, по которому можно передвигаться, на 2 меньше в высоту и ширину. Например, на рис.6 (а), размер которого 10 на 10 клеток, область передвижения размером 8 на 8 клеток.

Входные данные подаются в формате .txt файла, в котором каждый символ, включая пробелы, подразумевается, как клетка пространства, а при помощи специально условленных символов обозначаются необходимые элементы: стены и препятствия - #, а начальная позиция – A.

```
struct Brain {
    public int x;
    public int y;
    public string path; }

struct Place {
    public int x;
    public int y; }

Queue<Brain> queue = new Queue<Brain>();
List<Place> visited = new List<Place>();

Brain brain;
brain.x = start.x;
brain.y = start.y;
brain.path = "";

Place place;
queue.Enqueue(brain);
```

Рис. 5. Переменные и их инициализация для алгоритма поиска в ширину

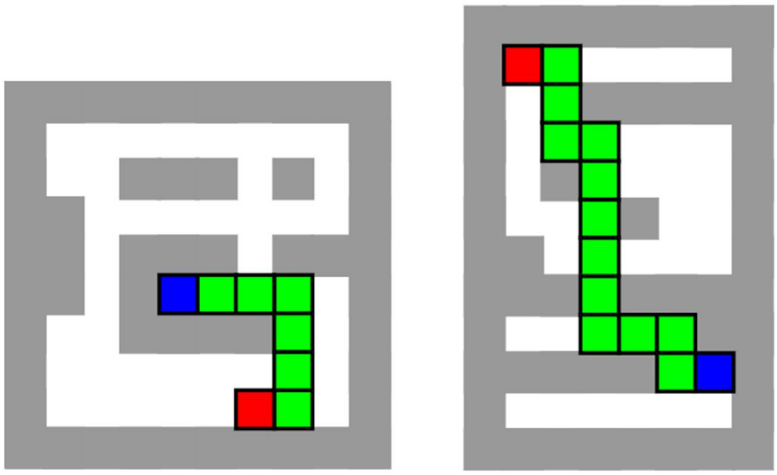
Fig. 5. Variables and their initialization for the breadth-first search algorithm

```
while (queue.Count > 0)
{
    brain = queue.Dequeue();
    foreach (Brain side in directions)
    {
        place.x = brain.x + side.x;
        place.y = brain.y + side.y;
        if (place.x < 0 || place.x >= w)
            continue;
        if (place.y < 0 || place.y >= h)
            continue;
        if (map[place.x, place.y] != ' ')
            continue;
        if (visited.Contains(place))
            continue;
        visited.Add(place);

        Brain step = new Brain(place, brain.path + side.path);
        if (place.Equals(finish))
            return step.path;
        queue.Enqueue(step);
    }
}
return "no path";
```

Рис. 6. Программная реализация алгоритма поиска в ширину

Fig. 6. Implementation of the breadth-first search algorithm



а) **б**)

Рис. 7. Примеры работы поиска в ширину для нахождения пути на поле размером:
а – 10 на 10 клеток; **б** – 8 на 12 клеток

Fig. 7. Breadth-first search examples for finding a path on a field of size:
а – 10 by 10 cells; **б** – 8 by 12 cells

В примере слева была выведена следующая последовательность действий “R D D R D D D D D R R D R”, а в примере справа: “R U U U L L L”, где U – вверх, L – влево, R – вправо, D – вниз.

Результаты и их обсуждение

В ходе проведения эксперимента оценивались факторы выполнения алгоритма на время нахождения пути. В качестве определения новых закономерностей определялись факторы влияние размеров поля, сложности поля (количество, размеры, расположение препятствий) и удаленности стартовой и финальной точек на время выполнения алгоритма. Для того чтобы убрать погрешность загруженности ЦП сторонними программами каждый вариант был просчитан 100 раз. Затем было выведено среднее значение среди 100 запусков.

Для оценки влияния размеров поля были выбраны размеры 20x20, 25x25 и 30x30 клеток. Расположение препятствий, начальной и конечной точек переносилось с маленького поля на поле больших размеров без изменений, увеличение пространства происходило за счет добавления клеток по краям, препятствия добавлялись так же. В табл. 1 приведены полученные результаты:

Таблица 1. Результаты оценки влияния размеров поля

Table 1. Results of evaluating the impact of of field size

Размер / Size	20x20	25x25	30x30
Среднее время выполнения (мс)	~8.55	~20.15	~47.8

На основе полученных результатов сделан вывод, что увеличение размеров поля приводит к увеличению времени выполнения алгоритма за счет того, что увеличивается количество возможных вариантов пути, следовательно, увеличивается и граф для осуществления поиска в ширину. Большее количество вариантов требует большего количества времени для расчётов.

Для оценки влияния сложности поля были пустое поле, поле со случайно расставленными препятствиями и поле, в котором было смоделированные движение змейкой (слева на право/справа на лево). Размеры поля во всех случаях 20х20 клеток. Начальная и конечная позиция расставлены максимально далеко друг от друга в противоположных углах поля. В табл. 2 приведены полученные результаты.

На основе полученных результатов сделан вывод, что изменение сложности поля приводит к уменьшению времени выполнения алгоритма. Пустое поле потребовало больше всего времени так как алгоритму нужно было проверить все точки пространства на пути к финальной точке. С появлением препятствий, количество возможных вариантов уменьшилось, что и повлияло на уменьшение времени работы алгоритма.

Для оценки влияния удаленности стартовой и финальной точек было выбрано поле размером 20х20 клеток. Точки были расставлены на максимальном (на основе теоремы Пифагора на расстоянии ~25.5 точек), среднем (15.5 точек) и маленьком (7.8 точек) удалениях друг от друга. Расположение препятствий оставалось неизменным. В табл. 3 приведены полученные результаты.

Таблица 2. Результаты оценки влияния сложности поля

Table 2. Results of evaluating the impact of field complexity

Сложность / Difficulty	Пустое / Empty	Случайное / Random	“Змейка” / “Snake”
Среднее время выполнения (мс)	~15	~9	~5

Таблица 3. Результаты оценки влияния расстояния между точками

Table 3. Results of evaluating the impact of the distance between points

Расстояние / Distance	Маленькое / Small	Среднее / Medium	Максимальное / Maximum
Среднее время выполнения (мс)	~5	~6	~9

На основе полученных результатов сделан вывод, что чем дальше начальная и конечная точки друг от друга, тем больше времени потребуется на выполнение алгоритма, учитывая, что расположение препятствий не меняется.

В результате проведенного эксперимента получена закономерность $t_a = f(w, h)$ для подготовленных образцов искомого поля, которая выражается в зависимости времени выполнения алгоритма от длины и ширины поля.

Однако стоит отметить, что увеличение размера поля и добавление дополнительных препятствий влечет за собой увеличение использования памяти, так как появляется больше вершин для исследования и увеличивается количество ветвей в графе, то есть понадобится больше места для хранения информации о вершинах в очереди на проверку и посещённых вершинах. Данный факт является одним из недостатков рассмотренного алгоритма.

Также было проведено сравнение алгоритмов поиска в ширину и глубину на 2 типах карт: пустая и лабиринт. На пустой карте точки располагались в максимальном удалении друг от друга, то есть количество вершин графа было самым большим из возможных. Лабиринты были построены случайным образом. Время рассчитано в миллисекундах. Время выполнения является средним временем из 100 прогонов. Результаты сравнения представлены на рис. 8 и рис. 9.

На пустой карте заметно преимущество в скорости у поиска в глубину в 2-2,5 раза. Однако путь, найденный им, не был наикротчайшем, в то время как поиск в ширину показывал оптимальный путь. Причиной медлительности поиска в ширину в данном случае является то, что начальная и искомая точки максимально удалены друг от друга, то есть поиск в ширину производит проверку всей карты, в отличие от поиска в глубину.

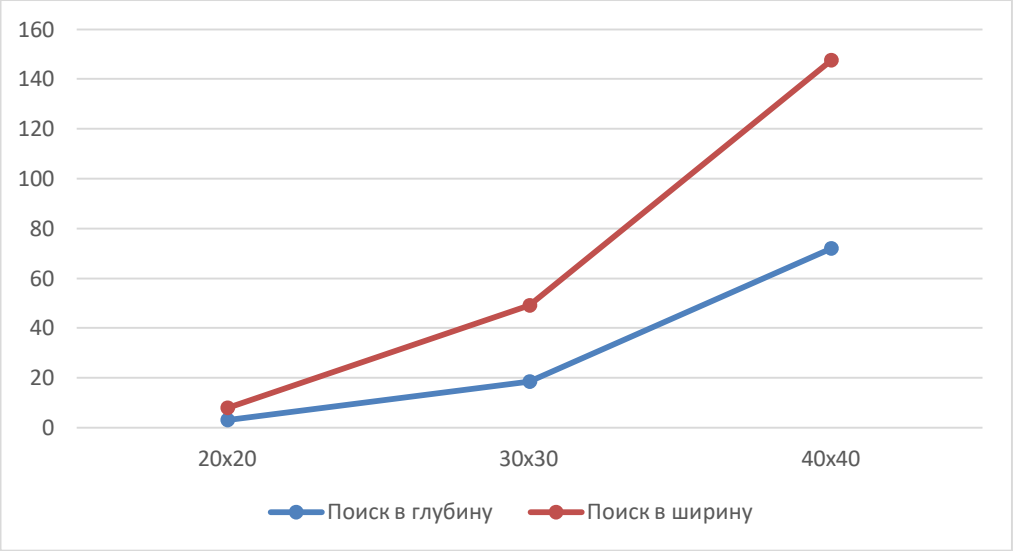


Рис. 8. Результаты сравнения алгоритмов поиска в ширину и глубину на пустой карте

Fig. 8. Comparison results of breadth-first and depth-first search algorithms on an empty map

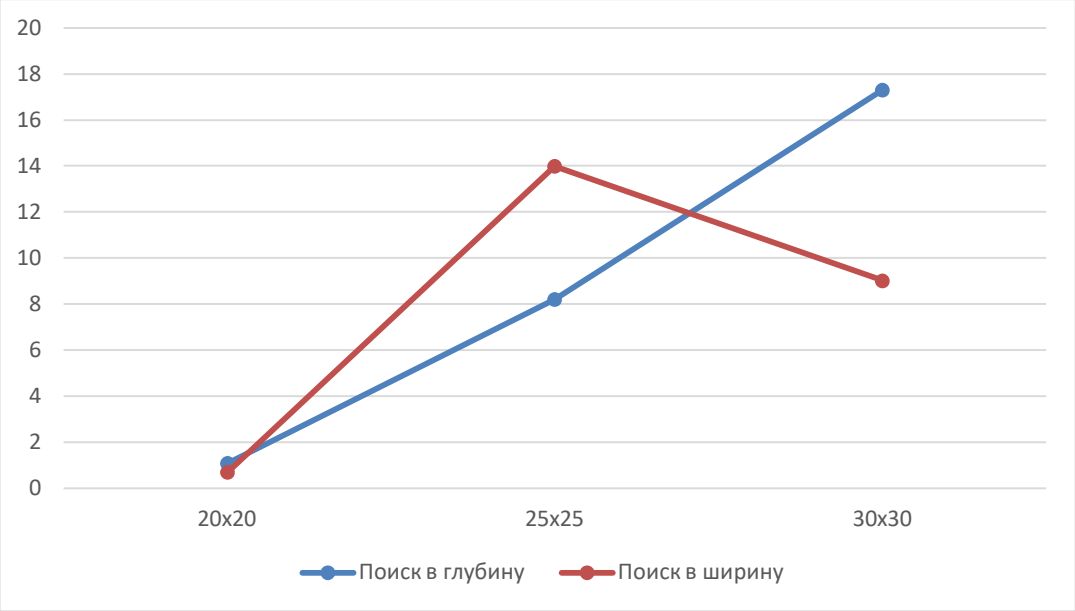


Рис. 9. Результаты сравнения алгоритмов поиска в ширину и глубину в лабиринте

Fig. 9. Comparison results of breadth-first and depth-first search algorithms in a maze

При этом, в лабиринте наблюдаются перемешанные результаты, то есть и тот и другой алгоритм могут быть быстрее. Это зависит от таких факторов, как удаленность точек, конфигурация лабиринта, реализация поиска в глубину, а точнее порядок выбора направления для постройки графа. То есть, если искомая точка находится в левом направлении от начальной, но алгоритм пойдет вправо, то время выполнения увеличится. Как и на пустой карте, в лабиринте путь, рассчитанный поиском в глубину, не был оптимальным. Поиск в ширину выдавал кратчайший путь.

Выводы

В данной работе был представлен алгоритм поиска в ширину, который обладает такими преимуществами, как способность находить самый короткий путь, время выполнения алгоритма в сравнении с аналогами, надежность (путь будет найден, если он существует) и детерминированность.

Представлена реализация алгоритма поиска в ширину на языке программирования c# и результаты экспериментов: изучение влияния различных факторов на время выполнения алгоритма и сравнение с алгоритмом поиска в глубину. Был сделан выводы, что увеличение размеров поля, уменьшения количества препятствий и расстояния между стартовой и финальной точками увеличивает время выполнения алгоритма. А также, что быстродействие обоих алгоритмов варьируется и зависит от удаленности точек и конфигурации лабиринта. Однако во всех случаях поиск в ширину выдавал кротчайший путь, поиск в глубину – нет.

В будущем исследовании планируется изучение влияния размеров поля на найденную дистанцию.

Список литературы

1. Panigrahi P.K., Tripathy H.K. Analysis on intelligent based navigation and path finding of autonomous mobile robot // Information systems design and intelligent applications. 2015; № 339: 219–232. http://dx.doi.org/10.1007/978-81-322-2250-7.

2. Luo Q., Wang H., Zheng Y., He J. Research on path planning of mobile robot based on improved ant colony algorithm // Neural Computing and Applications. 2020; № 32: 1555–1566. https://doi.org/10.1177/1729881418774673.

3. Fuzzy Gain-Based Dynamic Ant Colony Optimization for Path Planning in Dynamic Environments / V. Sangeetha, R. Krishankumar, K.S. Ravichandran, F. Cavallaro, S. Kar, D. Pamucar, A. A. Mardani // Symmetry. 2021; №13(2): 280. https://doi.org/10.3390/sym13020280.

4. 3D path planning for the ground robot with improved ant colony optimization / L. Wang, J. Kan, J. Guo, C. Wang // Sensors. 2019; №19(4): 815. https://doi.org/10.3390/s19040815.

5. Energy-efficient green ant colony optimization for path planning in dynamic 3D environments / V. Sangeetha, R. Krishankumar, K. Ravichandran, S. Kar // Soft Computing. 2021; №25(15): 4749–4769. https://doi.org/10.1007/s00500-020-05483-6.

6. Saraswathi M., Murali G.B., Deepak B. Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm // Procedia Computer Science. 2018; № 133: 510–517. https://doi.org/10.1016/j.procs.2018.07.064.

7. Gilhyun R. Applying asynchronous deep classification networks and gaming reinforcement learning-based motion planners to mobile robots // 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018; https://doi.org/10.1109/ICRA.2018.8460798.

8. Path Planning of multiagent constrained formation through deep reinforcement Learning / Z. Sui, Z. Pu, J. Yi, X. Tian // 2018 International Joint Conference on Neural Networks (IJCNN), 2018; https://doi.org/10.1109/IJCNN.2018.8489066.

9. Continuous control with deep reinforcement learning / T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra // Journal of Data Analysis and Information Processing. 2016; №4(4). https://doi.org/10.48550/arXiv.1509.02971.

10. Trust region policy optimization / J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel // 31st International Conference on Machine Learning. 2015; https://doi.org/10.48550/arXiv.1502.05477.

11. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms. arXiv 2017, arXiv:1707.06347v2. 2017; https://doi.org/10.48550/arXiv.1707.06347.

12. A Collision-Aware Mobile Robot Navigation in Grid Environment using Improved Breadth First Search / H.K. Tripathy, S. Mishra, H.K. Thakkar, D. Rai // Computers & Electrical Engineering. 2021; №4. https://doi.org/10.1016/j.compeleceng.2021.107327.

13. Panigrahi P.K., Bisoy S.K., Tripathy H.K. Intelligent Path Planning with Improved BFS (I-BFS) Strategy for Mobile Robot in RFID Equipped Unknown Environment // Proceedings of International Conference on Computational Intelligence and Data Engineering. 2022; №99:237-249. https://doi.org/10.1007/978-981-16-7182-1_20.

14. Breadth First Search Approach for Shortest Path Solution in Cartesian Area / R. Rahim, D. Abdullah, S. Nurarif, M. Ramadhan et al. // Journal of Physics: Conference Series. 2018; №1019(1):12-36. https://dx.doi.org/10.1088/1742-6596/1019/1/012036.

15. Zhou C., Huang B., Fränti P. A review of motion planning algorithms for intelligent robots // Journal of Intelligent Manufacturing. 2022; №33: 387–424. https://doi.org/10.1007/s10845-021-01867-z.

16. Sánchez-Ibáñez J.R., Pérez-del-Pulgar C.J., García-Cerezo, A. Path Planning for Autonomous Mobile Robots: A Review // Sensors. 2021; №21: 7898. https://doi.org/10.3390/s21237898.

17. Adaptive path finding algorithm in dynamic environment for warehouse robot / Ng MK., Chong, YW., Ko, Km. et al. // Neural Computing and Applications. 2020; №32: 13155–13171. https://doi.org/10.1007/s00521-020-04764-3.

18. Search Methods in Motion Planning for Mobile Robots / L. Paulino, C. Hannum, A.S. Varde, C.J. Conti // Intelligent Systems and Applications. 2021; № 296: 802-822. https://doi.org/10.1007/978-3-030-82199-9_54 2021.

19. Черноскутов М. А. Использование GPU для ускорения поиска в ширину на графах // Параллельные вычислительные технологии 2013 (ПаВТ'2013). 2013. С. 560-565.

20. Колганов А. С. Самая быстрая и энергоэффективная реализация алгоритма поиска в ширину на одноузловых различных параллельных архитектурах согласно рейтингу Graph500 // Параллельные вычислительные технологии (ПаВТ'2018). 2018. С. 273-285.

21. Козик А. А., Побегайло А.П. Ускорение параллельного поиска в ширину в графах при вычислениях на GPU // Вестник БГУ. 2015; № 2: 102-107.

22. Черноскутов М. А. Методы высокопроизводительной обработки графов // Восьмая Сибирская конференция по параллельным и высокопроизводительным вычислениям. Томск, 2015. С. 73-80. DOI 10.17223/978-5-7511-2389-5/11.

23. Черноскутов М. А. Параллельная высокопроизводительная обработка графов // Параллельные вычислительные технологии (ПаВТ'2016). 2016; С. 736-742.

24. Corey L. Lanum. Visualizing Graph Data. New York: Manning Publications Co; 2016.

25. Correll N. Introduction to autonomous robots. Davis: LibreTexts; 2021.

26. Касьянов Я. В., Федотова Е. В., Штыбина К. В. Анализ развития алгоритмов планирования маршрутов мобильных наземных роботов // Автоматизация, мехатрони-

ка, информационные технологии: материалы X Международной научно-технической интернет-конференции молодых ученых. Омск, 2020. С. 64-70.

27. Бобырь М.В., Титов В.С., Беломестная А.Л. Стабилизация теплового режима в процессе резания // Мехатроника, автоматизация, управление. 2010. № 6. С. 38-41.

28. Bobyr M.V., Titvo V.S. Nasser A.A. Automation of the Cutting-Speed Control Process Based on Soft Fuzzy Logic Computing // Journal of Machinery Manufacture and Reliability. 2015. Vol.44. No7. P. 633-641. DOI 10.3103/S1052618815070067.

29. Титов В.С., Бобырь М.В., Милостная Н.А. Автоматическая компенсация тепловых деформаций шпиндельных узлов прецизионного оборудования с ЧПУ // Промышленные АСУ и контроллеры. 2006. № 11. С. 31-35.

30. Захаров К.С., Савельев А.И. Сглаживание кривизны траектории движения наземного робота в трехмерном пространстве // Известия Юго-Западного государственного университета. 2020; 24(4): 107-125. https://doi.org/10.21869/2223-1560-2020-24-4-107-125.

References

1. Panigrahi P.K., Tripathy H.K. Analysis on intelligent based navigation and path finding of autonomous mobile robot. *Information systems design and intelligent applications*. 2015; № 339: 219–232. http://dx.doi.org/10.1007/978-81-322-2250-7.

2. Luo Q., Wang H., Zheng Y., He J. Research on path planning of mobile robot based on improved ant colony algorithm. *Neural Computing and Applications*. 2020; № 32: 1555–1566. https://doi.org/10.1177/1729881418774673.

3. Sangeetha V., Krishankumar R., Ravichandran K.S., Cavallaro F., Kar S., Pamucar D., Mardani A. A Fuzzy Gain-Based Dynamic Ant Colony Optimization for Path Planning in Dynamic Environments. *Symmetry.* 2021; №13(2): 280. https://doi.org/10.3390/sym13020280.

4. Wang L., Kan J., Guo J., Wang, C. 3D path planning for the ground robot with improved ant colony optimization. *Sensors.* 2019; №19(4): 815. https://doi.org/10.3390/s19040815.

5. Sangeetha V., Krishankumar R., Ravichandran K., Kar S. Energy-efficient green ant colony optimization for path planning in dynamic 3D environments. *Soft Computing*. 2021; №25(15): 4749–4769. https://doi.org/10.1007/s00500-020-05483-6.

6. Saraswathi M., Murali G.B., Deepak B. Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm. *Procedia Computer Science*. 2018; № 133: 510–517. https://doi.org/10.1016/j.procs.2018.07.064.

7. Gilhyun R. Applying asynchronous deep classification networks and gaming reinforcement learning-based motion planners to mobile robots. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018; https://doi.org/10.1109/ICRA.2018.8460798.

8. Sui Z., Pu Z., Yi J., Tian X. Path Planning of multiagent constrained formation through deep reinforcement Learning. *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018; https://doi.org/10.1109/IJCNN.2018.8489066.

9. Lillicrap T. P., Hunt J. J., Pritzel A., Heess N., Erez T., Tassa Y., Silver D., Wierstra D. Continuous control with deep reinforcement learning. *Journal of Data Analysis and Information Processing*. 2016; №4(4). https://doi.org/10.48550/arXiv.1509.02971.

10. Schulman J., Levine S., Moritz P., Jordan M. I., Abbeel P. Trust region policy optimization. *31st International Conference on Machine Learning*. 2015; https://doi.org/10.48550/arXiv.1502.05477.

11. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms. *arXiv 2017, arXiv:1707.06347v2*. 2017; https://doi.org/10.48550/arXiv.1707.06347.

12. Tripathy H.K., Mishra S., Thakkar H.K., Rai D. A Collision-Aware Mobile Robot Navigation in Grid Environment using Improved Breadth First Search. *Computers & Electrical Engineering*. 2021; №4. https://doi.org/10.1016/j.compeleceng.2021.107327.

13. Panigrahi P.K., Bisoy S.K., Tripathy H.K. Intelligent Path Planning with Improved BFS (I-BFS) Strategy for Mobile Robot in RFID Equipped Unknown Environment. *Proceedings of International Conference on Computational Intelligence and Data Engineering*. 2022; №99:237-249. https://doi.org/10.1007/978-981-16-7182-1_20.

14. Rahim R., Abdullah D, Nurarif S., Ramadhan M. et al. Breadth First Search Approach for Shortest Path Solution in Cartesian Area. *Journal of Physics: Conference Series*. 2018; №1019(1):12-36. https://dx.doi.org/10.1088/1742-6596/1019/1/012036.

15. Zhou C., Huang B., Fränti P. A review of motion planning algorithms for intelligent robots. *Journal of Intelligent Manufacturing*. 2022; №33: 387–424. https://doi.org/10.1007/s10845-021-01867-z.

16. Sánchez-Ibáñez J.R., Pérez-del-Pulgar C.J., García-Cerezo, A. Path Planning for Autonomous Mobile Robots: A Review. *Sensors*. 2021; №21: 7898. https://doi.org/10.3390/s21237898.

17. Ng MK., Chong, YW., Ko, Km. et al. Adaptive path finding algorithm in dynamic environment for warehouse robot. *Neural Computing and Applications*. 2020; №32: 13155–13171. https://doi.org/10.1007/s00521-020-04764-3.

18. Paulino L., Hannum C., Varde A.S., Conti, C.J. Search Methods in Motion Planning for Mobile Robots. *Intelligent Systems and Applications*. 2021; № 296: 802-822. https://doi.org/10.1007/978-3-030-82199-9_54 2021.

19. Chernoskutov M. A. Ispol'zovanie GPU dlya uskoreniya poiska v shirinu na grafakh [Using GPU to speed up breadth-first search on graphs]. *Parallel'nye vychislitel'nye tekhnologii 2013 (PaVT'2013) = Parallel computing technologies 2013 (PaVT'2013)*, 2013; pp. 560-565.

20. Kolganov A. S. Samaya bystraya i energoeffektivnaya realizatsiya algoritma poiska v shirinu na odnouzlovykh razlichnykh parallel'nykh arkhitekturakh soglasno reitingu Graph500 [The fastest and most energy-efficient implementation of breadth-first search algorithm on single-node various parallel architectures according to Graph500 rating]. *Parallel'nye vychislitel'nye tekhnologii (PaVT'2018) = Parallel computing technologies (PaVT'2018)*, 2018, pp. 273-285.

21. Kozik A.A., Pobegailo A.P. Uskorenie parallel'nogo poiska v shirinu v gra-fakh pri vychisleniyakh na GPU [Acceleration of parallel breadth-first search in graphs when computing on the GPU]. *Vestnik BGU = Bulletin of BSU*. 2015, no. 2: 102-107.

22. Chernoskutov M.A. [Methods of high-performance processing of graphs]. *Vos'maya Sibirskaya konferentsiya po parallel'nym i vysokoproizvoditel'nym vychisleniyam [Eighth Siberian Conference on Parallel and High-Performance Computing]*. Tomsk, 2015; pp. 73-80. DOI 10.17223/978-5-7511-2389-5/11 (In Russ.).

23. Chernoskutov M. A. Parallel'naya vysokoproizvoditel'naya obrabotka grafov [Parallel high-performance processing of graphs]. *Parallel'nye vychislitel'nye tekhnologii (PaVT'2016) = Parallel computing technologies (PaVT'2016).* 2016; pp. 736-742.

24. Corey L. Lanum. *Visualizing Graph Data*. New York: Manning Publications Co; 2016.

25. Correll N. *Introduction to autonomous robots*. Davis: LibreTexts; 2021.

26. Kasyanov Ya. V., Fedotova E. V., Shtybina K. V. [Analysis of the development of route planning algorithms for mobile ground robots]. *Avtomatizatsiya, mekhatronika, informatsionnye tekhnologii. Materialy X Mezhdunarodnoi nauchno-tekhnicheskoi internet-konferentsii molodykh uchenykh* [Automation, mechatronics, information technology. Proceedings of the X International Scientific and Technical Internet Conference for Young Scientists]. Omsk, 2020, pp. 64-70 (In Russ.).

27. Bobyr M.V., Titov V.S., Belomestnaja A.L. Stabilizatsiya teplovogo rezhima v protsesse rezaniya [Stabilizacija teplovogo rezhima v processe rezanija]. *Mekhatronika, avtomatizatsiya, upravlenie = Mehatronika, Avtomatizacija, Upravlenie*, 2010, no. 6, pp. 38-41.

28. Bobyr M.V., Titvo V.S. Nasser A.A. Automation of the Cutting-Speed Control Process Based on Soft Fuzzy Logic Computing. *Journal of Machinery Manufacture and Reliability*. 2015, vol.44, no.7, pp. 633-641. DOI 10.3103/S1052618815070067.

29. Titov V.S., Bobyr M.V., Milostnaya N.A. Avtomaticheskaya kompensatsiya teplovykh deformatsii shpindel'nykh uzlov pretsizionnogo oborudovaniya s ChPU [Avtomaticheskaja kompensacija teplovyh deformacij shpindel'nyh uzlov precizionnogo oborudovanija s ChPU]. *Promyshlennye ASU i kontrollery = Industrial Automatic Control Systems and Controllers*, 2006, no. 11,pp. 31-35.

30. Zakharov K. S., Saveliev A. I. Smoothing the Curvature of Trajectory of Ground Robot in 3D Space. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2020; 24(4): 107-125 (In Russ.). https://doi.org/10.21869/2223-1560-2020-24-4-107-125.

Информация об авторах / Information about the Authors

Емельянов Сергей Геннадьевич, доктор технических наук, профессор, ректор, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: rector@swsu.ru, ORCID: https://orcid.org/0000-0002-3012-0383, Researcher ID: E-3511-2013

Sergei G. Emelianov, Dr. of Sci. (Engineering), Professor, Rector, Southwest State University, Kursk, Russian Federation, e-mail: rector@swsu.ru, ORCID: https://orcid.org/0000-0002-3012-0383, Researcher ID: E-3511-2013

Бобырь Максим Владимирович, доктор технических наук, профессор кафедры вычислительной техники, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: fregat_mn@rambler.ru, ORCID: http://orcid.org/0000-0002-5400-6817, Researcher ID: G-2604-2013

Maxim V. Bobyr, Dr. of Sci. (Engineering), Associate Professor of the Computer Engineering Department, Southwest State University, Kursk, Russian Federation, e-mail: fregat_mn@rambler.ru, ORCID: http://orcid.org/0000-0002-5400-6817, Researcher ID: G-2604-2013

Крюков Александр Георгиевич, аспирант, Юго-Западный государственный университет, г. Курск, Российская Федерация, e-mail: saskryukov@yandex.ru, ORCID: http://orcid.org/0000-0002-4235-6975, Researcher ID: AFA-6564-2022

Aleksander G. Kryukov, Post-Graduate Student, Southwest State University, Kursk, Russian Federation, e-mail: saskryukov@yandex.ru, ORCID: http://orcid.org/0000-0002-4235-6975, Researcher ID: AFA-6564-2022