Optimizing the A* search algorithm for mobile robotic devices

IOP Conf. Series: Materials Science and Engineering **516** (2019) 012054 doi:10.1088/1757-899X/516/1/012054

Optimizing the A* search algorithm for mobile robotic devices

V V Maneev, M V Syryamkin

National Research Tomsk State University, Tomsk, Russia
E-mail: worklovery@gmail.com

Abstract. The purposes of this article are to analyze the pathfinding algorithms used to search and calculate the trajectory of movement of mobile autonomous robotic devices and to search for optimization methods in order to reduce the load on computing and memory units of the mobile devices. This paper analyzes the basic pathfinding algorithms including breadth-first search, depth-first search, Dijkstra's algorithm, greedy best-first search, and the A*algorithm, considers principles of their work, and proposes optimization methods. This work results in a new approach for using the A* algorithm for solving problems associated with pathfinding in real dynamic environments. The proposed approach speeds up the process of path calculation and the choice of the movement direction and reduces the requirements for computing power of autonomous mobile devices.

1. Introduction

The A* search algorithm is widely used in games and for planning routs of mobile robots [1]. This is a highly efficient heuristic algorithm for searching an acceptable and shortest path. It allows finding the least-cost path between the starting and ending points. The A*algorithm is considered one of the most acceptable algorithms because it combines the advantages of several search algorithms including breadth-first search and depth-first search algorithms and is already an optimized version of Dijkstra's algorithm. However, there is still a serious problem for small autonomous mobile devices because in some cases, the existing A* search algorithm may require significant computational power, which may adversely affect the speed of making decision about the direction and trajectory of motion [2]. In addition, in its pure form, the A* algorithm does not allow taking into account conditions that may arise in real environments such as dynamic objects or the final goal with an unpredictable motion path. In this paper, we consider methods and approaches that allow optimizing the A* algorithm, reducing the performance costs of computational units, and rebuilding the system to control dynamic changes and goals with an unpredictable motion trajectory.

2. Materials and methods

The pathfinding problem is well studied and the pathfinding algorithms are widely used in designing computer games and planning routes including auxiliary navigation systems. The simplest algorithm for solving the shortest path problem is the breadth-first search algorithm. It performs a sequential search of all possible paths evenly in all directions. Implementation of this algorithm considers neither the 'weight' of each segment nor the cost of calculating the shortest path.

The depth-first search algorithm is more advanced for our purposes. As the main direction of the recursive search, the depth-first search algorithm selects the direction of the target. The depth-first search algorithm searches all routes outgoing from the point under examination. If the route leads to a point that has not been considered earlier, the algorithm starts from this unexamined point, and then returns

to the previous level and continues the search. This algorithm also does not consider the cost of passing a route [3].

Dijkstra's algorithm is a heuristic analogue of the breadth-first search algorithm. When finding the shortest route, the Dijkstra's algorithm considers the 'weights' of each segment of the route. However, as well as the breadth-first search algorithm, the Dijkstra's algorithm is very resource-intensive.

A* search algorithm is a modification of the Dijkstra's algorithm. It has the advantages of the depth-first search algorithm and, like the Dijkstra's algorithm, is heuristic, that is, it allows taking into account the cost of passing a route along each of its segments.

For the purposes of this study, we selected the A* search algorithm as the main algorithm. The basis of the A*search algorithm is the function of estimating the cost of passing through graph points. If the assessment function cannot be correctly selected, the search result may be an accessible but not optimal path. The A*search algorithm uses f (n) = g (n) + h (n) as an evaluation function for expanding nodes options in the OPEN list. In this formula, g (n) is an actual cost between the initial node and the current node n (the cost of the optimal path found), and h (n) is an estimate of the cost of the optimal path between the current node n and the target node. The disadvantage of the algorithm is in too large search area, which leads to an exponential growth of the studied vertices with an increase in the distance to the ending point. It also affects the memory consumption (which also increases exponentially) and the complexity becomes polynomial when the heuristics satisfies the condition: $|h(n) - h^*(n)| \leq O(\log h^*(n))$ [4].

These deficiencies in the A* search algorithm may be fatal when using it for portable mobile robotic devices because miniature robots cannot carry powerful computing systems with a large amount of RAM. So, to bypass a simple obstacle using the A* search algorithm, 760 (92) points will need to be analyzed, and the path length will be only 89 points (Figure 1).

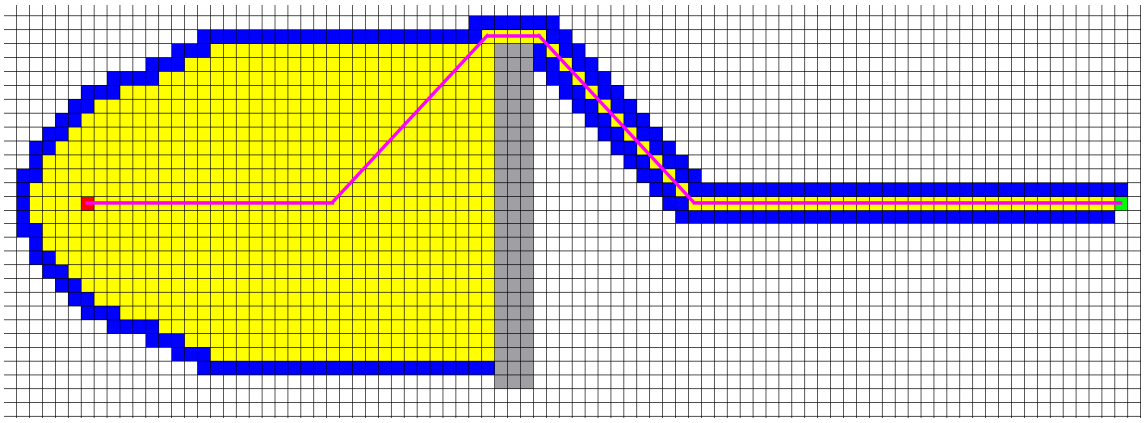

Figure 1. A* search algorithm.

For a 10×10 cm mobile robotic device, the length of the covered path will be only 8.9 meters. As the distance between the mobile device and the ending point of its route (and the number of obstacles in its path) increases, the complexity of calculating the trajectory of movement (as well as the amount of the required memory for storing data) increases exponentially.

3. The methods for optimization of the A* search algorithm.

There are several variations of the A*algorithm proposed for solving problems of exponential growth in the computation of the optimal path. These variations include iterative deepening A* (IDA*), memory-bounded A* (MA*), simplified MA (SMA*) and recursive best first search (RBFS). As a rule, all modifications are aimed at improving or changing the heuristic part of the A* algorithm. Although the heuristic component of the algorithm is the main and most important part of the pathfinding algorithm, optimizing only the heuristic part of the algorithm may not be enough to solve the optimization problems of autonomous mobile devices [5].

In dynamic and natural environments, it is required to make continuous calculations and adjustments to the route of movement in order to bypass unexpected obstacles, moving objects and targets. For example, to intercept a target moving fast along a complex unpredictable trajectory, it is required to recalculate the trajectory and correct the route faster than the target leaves the conditional point of its current location (when the size of the graph points of the A*algorithm is equal to the dimensions of the robot). Resolution of such problems in the usual way is not possible. However, as a rule, in real environments, the obstacles are more fragmented, and the interaction objects such as buildings, cars, or people are much larger than autonomous mobile devices, robots, and drones. Therefore, the distance between them can be significant and the dimensions of the passages between obstacles are many times greater than the dimensions of a mobile autonomous robotic device. This fact allows us to reduce significantly the number of required operations and the amount of RAM used to calculate and store the optimal route if in order to find the path, we use nested cycles with geometrically proportional graph nodes [6, 7].

4. Features of A*algorithm optimization

For the above-mentioned example of finding a path using the A* algorithm, we use nested cycles with geometrically proportional graph nodes (Figure 2).

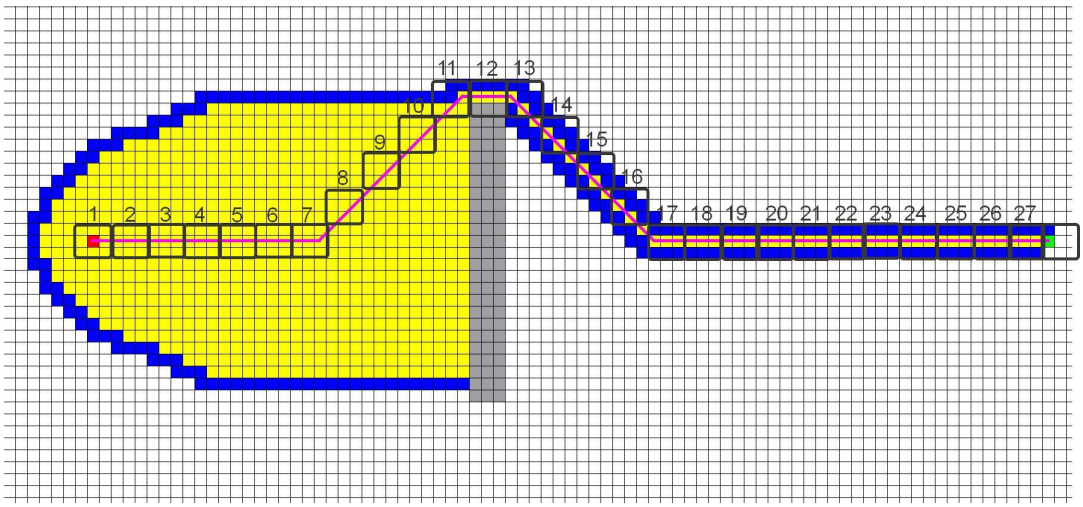

Figure 2. Simplified A*.

Figure 2 shows that in order to build a complete route, we need to perform 27 simplified pathfinding operations, using the A* algorithm and another 27 operations to search for a route until each next block of calculations. Table 1 shows these calculations.

Table 1. Calculating a path.

Step	Points checked			Step	Points checked		
	Main block	Additional block	Full solution		Main block	Additional block	Full solution
1	109	4	760	15	14	4	39
2	102	4	690	16	13	4	36
3	94	4	621	17	12	4	33
4	85	4	553	18	11	4	30
5	75	4	483	19	10	4	27
6	67	4	415	20	9	4	24
7	55	4	345	21	8	4	21
8	43	4	250	22	7	4	18
9	32	4	167	23	6	4	15
10	23	4	95	24	5	4	12

11	18	4	52	25	4	4	9
12	17	4	48	26	3	4	6
13	16	4	45	27	3	4	3
14	15	4	42				
Total	751	56	4566		105	52	273
			964				**4839**

Table 1 shows that the simplified A* algorithm (example shown in Figure 2) in case of the need to clarify the route periodically, needs to analyze only 964 points of the graph, and the maximum amount of required memory cells is 109. At the same time, using the A* algorithm (when clarifying the route at the same points and at the same intervals), it was required to analyze 4839 points of the graph and to use 760 memory cells to store data. For this case, the increase in performance using our approach to build a route was 4839/964 = 5.02 times, and the memory usage decreased by 760/109 = 6.97 times. Moreover, the efficiency will multiply increase in case of an increase in the distance to the final object, the frequency of route clarification and the degree of nesting of the algorithm. This will significantly reduce the requirements for the computing power of autonomous robotic devices.

Implementation of this approach requires substantial development and complication of the heuristic function of the algorithm, which may adversely affect performance, but the overall effectiveness of the solution will remain.

5. Conclusion

In this work, we were able to develop a fundamentally new approach to using the A* algorithm for solving the pathfinding problem in conditions of the need to continually adjust the route or track a moving target. This approach significantly reduces the computing power requirements of mobile robots and drones and allows scaling solutions immediately depending on the required accuracy of routing or the requirements for the speed of response to environmental changes.

Acknowledgments

The paper was supported by “The Tomsk State University competitiveness improvement programme” under grant (No 8.2.24.2018) and by the Russian Foundation for Basic Research (grant No 16-29-04388). The authors are grateful to Tatiana B. Rumyantseva from Tomsk State University for English language editing.

References

[1] Zheng-hong Hu, Jin Li. *2010 International Conference on Computational Aspects of Social Networks* 738–739
[2] Shashev D. V., Shidlovskiy S. V. 2017 *Journal of Physics Conference Series* **881** 19–26
[3] Aggarwal A., Bhalla J.S. 2013 *The Next Generation Information Technology Summit (4th International Conference)* 69–75
[4] Cheng Rao, Lianqing Yu. 2010 *The 2nd International Conference on Industrial Mechatronics and Automation* 221–224
[5] Miao Wang, Hanyu Lu *2012 International Conference on Industrial Control and Electronics Engineering* 1739–1742
[6] Shikhman M. V., Shidlovskiy S. V. 2017 *IOP Conference Series-Materials Science and Engineering* 363.
[7] Yao J., Zhang B., Zhou Q. 2009 *World Congress on Software Engineering* 515