Pathfinding in a Dynamically Changing Environment

Eva Volna(✉) and Martin Kotyrba(✉)

Department of Informatics and Computers,
University of Ostrava, 30. dubna 22, 70103 Ostrava, Czech Republic
{eva.volna,martin.kotyrba}@osu.cz

Abstract. This paper is focused on a proposal of an algorithm for dynamic path planning which uses the input speed of the vehicle as a dynamic value based on which paths are searched in a graph. The proposed algorithm AV* stems from algorithm A*, works with proposed weight functions of evaluation function and, once again, uses information about nodes which have already been calculated in order to find out a faster update of a new path. This algorithm has been subjected to tests, whose results were compared with the results of algorithm A*. The proposed algorithm AV* complies with the condition of pathfinding between two points in a finite time based on the criterion of the fastest path using the maximum of the speed limit. From the point of view of dynamic optimization, all tests achieved a higher speed when the algorithm was run again at a different speed. This speed-up corresponds to approximately 25% and is linearly dependent on the amount of searched nodes.

Keywords: Dynamic pathfinding · A* algorithm · AV* algorithm

1 Introduction

Path planning is one of the basic tasks when navigating robots. Path planning is a complex process based on the use of information about an environment. The information is based on the map layout of the environment, but it can also come directly from sensors placed on robots, i.e. visual sensors (cameras), laser or sonar. Overview of algorithms for path planning used in robotics can be found in [1]. Path planning falls into Artificial Intelligence, Image Processing, and Algorithmization. Algorithms for path planning and robotic navigation differentiate depending on what type of work environment, i.e. the static or dynamic algorithms.

There are a number of algorithms and approaches to planning a path from a start point to a final point to plan a path in a static environment, with the resultant proposed trajectory, which is not in collision with obstacles. Gavrilut et al. [3] propose a path planning algorithm based on cellular neural network. This algorithm uses image processing techniques to convert camera output to a sequence of motions. The advantage of this method lies in its easy parallelization. Although this method has been implemented, no comparative tests with existing approaches are recorded. Kroumov and Yu [7] present their model of mobile robots' navigation, which is based on path planning using neural networks. Their methods use the potential field for path planning and

N. T. Nguyen et al. (Eds.): ACIIDS 2018, LNAI 10752, pp. 265–274, 2018.
https://doi.org/10.1007/978-3-319-75420-8_25

propose their own planning algorithm that is easily parallelizable. The inputs of the networks are the coordinates of the points of the path. The output neuron is described by the expression, which is called a repulsive penalty function and has a role of repulsive potential. The algorithm also solves the deadlock in a local minimum, however, simulations and tests are proposed for static environments only. Overview of other algorithms for path planning is given in [8]. Here are described algorithms covering the following areas: discrete planning, logic-based planning methods, motion planning, sampling-based motion planning, combinatorial motion planning, decision-theoretic planning, planning under sensing uncertainty, and planning under differential constraints.

On the other hand, path planning algorithms in a dynamic environment are more complex with regard to the amount of requirements placed on the resulting path, such as real-time requirement, obstacle prediction efficiency, etc. An overview of dynamic path planning algorithms used in robotics can be found in [10]. Authors' own work presents an algorithm capable of negotiating obstacles in static, partially dynamic and dynamic environments. The advantage of dynamic path planning algorithms lies in their ability to reuse information processed from previous searches to find a new solution faster. These algorithms include, for example, LPA*, Dynamic A* (D*) and later D* Lite, etc. Even though path planning algorithms have been developed for over two decades, there is still a space for improvement of existing solutions, either to accelerate the overall algorithm progress or to implement these algorithms in conjunction with the given hardware.

The research of path planning algorithms often focuses on finding the shortest path to reduce time for calculation, number of nodes, use as few sources as possible, etc. By focusing on these factors, a great deal of generality of design and use is guaranteed. This is well-founded, as path mapping algorithms are useful for various graph simulations, computer games, or robotic navigation. Areas of real and virtual applications are so different that they cannot be treated equally. Virtual applications have their boundaries and complexity defined, while real-world applications have to work not only with the degree of coincidence but also with a large increase in complexity with a number of factors that affect the calculation. Applications in the real world also work with the orientation of the path towards the current orientation of the robot. The path labeled as the shortest can take longer than the other way, which includes more number of nodes, depending on how many times the robot is forced to change the direction. Changing the direction affects the speed of the robot, especially in the case of axial movement, when it is almost forced to stop due to a change of direction. A general solution is thus necessary to adjust not only to find the shortest way, but also on finding the "easiest" way. For example, publications deal with this issue [4, 11], where it was found that the way found with the least number of directional changes was on average about 20% longer than the shortest route. Although the path planning area is considered to be explored, new hybrid algorithms based on existing algorithms are created, e.g. algorithms PRM, D*, D* Lite [3, 10] and existing approaches are optimized [2, 9]. Most of the proposed solutions are tested only in the form of program simulation, which are quite different from real-world use. Another relevant literature is rather focused on creating an overview, categorizing algorithms and comparing their effectiveness. From the conclusions of the current state analysis, we can say that the

algorithms used for dynamic robotic path planning do not reflect the acceleration of robots but only reflect their instantaneous speed. They also do not define time requirements for way calculation and its update in the case of an obstacle, which in their use in the real world is a problem. It is usually calculated with zero speed when changing the direction of movement of the robot. The robot must stop, turn and accelerate at every change of direction, which is time consuming. This solution is presented in [9].

The research goal of this paper is to propose a path planning algorithm in a dynamically changing environment. It will be taken into account the requirement of real time processing in association with the movement of the agent. The theoretically proposed algorithm will be supported by computer simulation models.

2 Algorithm Proposal

The state of the art shows that many algorithms of dynamic pathplanning are based on the A* algorithm. This algorithm belongs to the category of algorithms working with visibility of graphs [4, 5, 9]. Therefore, this algorithm was also chosen as the basis for the pathfinding algorithm in a dynamically changing environment. To ensure a dynamic pathfinding, the criterion of finding the fastest route is chosen for the A* algorithm with respect to the maximum available speed. This speed is specific to each object and represents a changeable component for repeated searches.

The course of the algorithm A* is controlled by means of weights g_{cost} and $h_{cost.}$ These weights are used to calculate the function value f_{cost} (1):

$$f_{cost} = (g_{cost} + h_{cost}) \tag{1}$$

where

g_{cost} indicates the distance from a particular node to the start node;
h_{cost} indicates the distance from a particular node to the stop node;
f_{cost} is the evaluation function.

2.1 Decreasing the Number of Turns

Decreasing the number of turns is one of many criteria of static and dynamic path-planning. Paper [11] defines the path as “the simplest” in the case it contains a minimum number of direction changes. The core of the proposed algorithm for dynamic planning uses the principle of A* algorithm controlled by a heuristic function f_{cost} (1). The value of the heuristic function was added with a coefficient t_{cost}, which enables to prefer path alternatives with the lowest number of direction changes (2). This principle differs from implementation used in literature [11], where breadth-first-search is the subject.

$$f_{cost} = (g_{cost} + h_{cost} + t_{cost}) \tag{2}$$

Value t_{cost} in formula (2) is determined based on the information about the direction for each expanded node, which is recorded in a form of direction vector $\boldsymbol{v}$. This vector is then used to calculate value f_{cost} successor by calculating the vector product of vector $\boldsymbol{v}$ with direction vector $\boldsymbol{u}$ of the currently expanded node (3). With this optimization, algorithm A* prefers nodes which do not change direction.

$$t_{cost} = cos\,\alpha = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{|\boldsymbol{u}| \cdot |\boldsymbol{v}|} \tag{3}$$

2.2 Implementation of a Dynamic Part of Algorithm A*

The dynamic part of algorithm A* represented changeable speed of the vehicle. In real world, there are curves not influencing the speed of a vehicle. In such a case, it is necessary to consider physical and kinematic characteristics of a given vehicle which enable to identify such curves. The following modification of the algorithm focuses on this issue. In order to calculate values of function t_{cost}, parameters such as vehicle weight and speed as well as curve radius are used.

Weight. The influence of weight on the path trajectory can be imagined as a ball on a rope tied to a pole. If the ball is rotated around the axis created by the center of the pole, the ball is affected by a relatively small centripetal force. In the case that the ball is replaced by a bowling ball, its rotation needs a far more force to achieve the same speed. The same principle applies to the movement of an agent.

Speed. If the agent is controlled slowly in a circle, tires do not have problems with generating enough friction to keep it in a circular direction. However, if it speeds up, there is moment when the tires are not able to generate enough friction and skidding occurs. The maximum speed in a curve can be achieved using a general equation to calculate a centripetal force (4).

$$F_d = \frac{m \times v^2}{r} \tag{4}$$

where r is the curve radius, v is the agent speed, and m is its weight.

Curve radius. It influences agent movement similarly to its speed. If the curve radius decreases while the speed is the same, there is a certain point of tires skidding on road surface.

2.3 Speed Modification

Weight function t_{cost} ensures finding the fastest path by decreasing frequent speed drop. During the algorithm, the value of function t_{cost} is calculated and stored into individual nodes. The created map keep information on path surface, roughness, and radii which do not have to be calculated when finding a path, see Fig. 1.

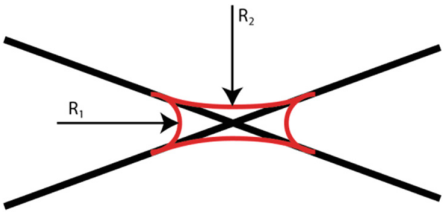


Fig. 1. Radii of non-linear paths

Map preprocessing is done during its uploading, before launching the algorithm respectively. During this process, the graph environment is supplied with data about all points of path crossings and individual curves. Calculating the value of function t_{cost} uses the relationship for centripetal force (4). The input value is the maximum vehicle speed v_{max}. During searching, all suitable alternatives of non-linear paths are tested as well as the maximum speed for radius r_{vmax}. The proposed algorithm AV* works in the following steps [6]:

1. For each expanded node:
   (a) Test if the node has two predecessors.
   (b) If yes, calculate the maximum speed in a curve $r_{vmax.}$
      (i) If $r_{vmax} < v_{max}$: set t_{cost} to $v_{max} - r_{vmax.}$
2. Calculate the value of the heuristic function (2).
3. Continue in algorithm A*.

3 Experimental Verification

The testing was realized on Lenovo IC 510S, 90GB007TCK. For the verification purposes, algorithm A* was compared with the proposed AV* in the following configurations:

1. Classical implementation of A* without any modifications or optimizations.
2. The proposed algorithm AV* (with modified weights to stem from the speed of the navigated agent (vehicle).
3. The proposed algorithm AV* with a pre-set speed. This algorithm is launched every time after the second iteration in order to analyze dynamic behavior of the proposed algorithm.

Input for each test is the graph environment containing nodes and edges. The memory stores the following information about it:

- Number of nodes and their positions.
- Number of paths and their node reference numbers.
- Friction coefficient for a given path.
- Maximum speeds for individual nodes stored in the course of searching.

Output of each test is:

- Number of nodes composing the found path.
- Length of the found path: real number in meters (m).
- Time to go through the found path: integers in seconds (s).
- Time of the calculation: integers in microseconds (μs).
- Average memory workload: integers in bits (bit).

Both tested algorithms A* and AV* are functionally incompatible. The same view cannot be applied on results from algorithm A*, which is focused on finding the shortest path. However, algorithm AV* is to find the shortest path in time. The basic version of algorithm A* is here only for purposes of evaluating the influence of the implemented modifications on the original algorithm. Each test is done fifty times and the stated results are averaged.

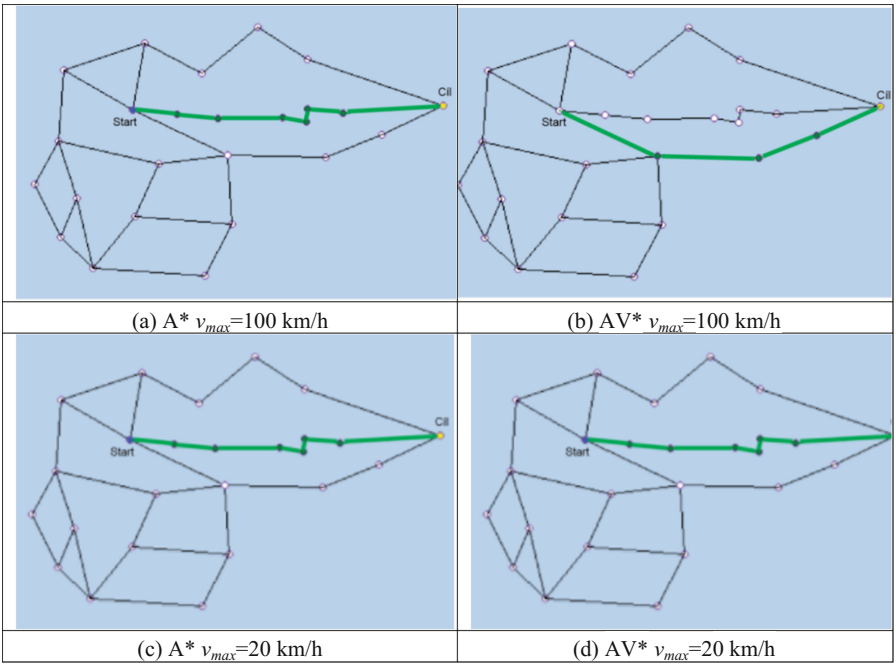
3.1 Test Task 1

Figure 2 depicts a testing environment with 25 nodes, where the start (blue point) and end (yellow point) are defined on nodes with several possible subsequent paths. This experiment verified that reducing the number of turns during searching can result in decreasing the time to go through the path. The experimental results are stated in Table 1.

Table 1. Experimental results for the 1st testing environment

Criterion	A* v_{max} = 100 km/h	AV* v_{max} = 100 km/h	A* v_{max} = 20 km/h	AV* v_{max} = 20 km/h
Number of nodes	8	5	8	8
Path length (m)	300	308	300	300
Time to do the path (s)	14	13	50	50
Time of calculation (μs)	1227	1744	1250	1432
Memory (bit)	1025	1501	990	1496

Both algorithms found a path and their results differed in all values, see Fig. 2. At v_{max} = 100 km/h, the path found by algorithm A* contains a higher number of nodes and is 8 m shorter. Considering the speed at which it can be done, its estimation is set to 14 s, which makes 1 s more than for the AV* algorithm. The characteristics of both paths is also significantly different. At v_{max} = 20 km/h, both algorithms found the same path.



(a) A* v_{max}=100 km/h (b) AV* v_{max}=100 km/h

(c) A* v_{max}=20 km/h (d) AV* v_{max}=20 km/h

Fig. 2. Found paths for testing task 1. (Color figure online)

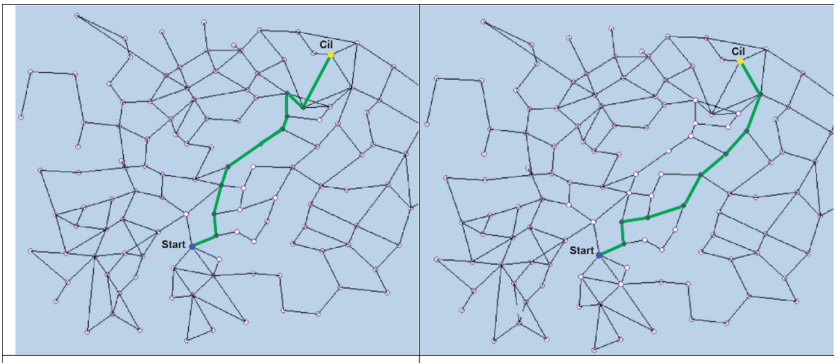
3.2 Test Task 2

Figure 3 depicts a testing environment with 129 nodes. This test uses a graph with a higher density of nodes. It primarily monitors saving in memory sources and using of previously acquired information. Experimental results are stated in Table 2. Decreasing the time to go through the path is a result of implementation of the maximum speed as a dynamic part.

Table 2. Experimental results for the 2nd testing environment

Criterion	A* v_{max} = 100 km/h	AV* v_{max} = 100 km/h	A* v_{max} = 20 km/h	AV* v_{max} = 20 km/h
Number of nodes	11	10	11	11
Path length (m)	380	391	380	380
Time to do the path (s)	19	17	64	64
Time of calculation (μs)	3356	16038	3412	11878
Memory (bit)	3341	4512	3410	4650

In this area, a path of the same length was found for both algorithms. As anticipated, the more difficult area, the worse speed of algorithm AV* due to its tendency to search non-penalized paths by value T. The character of the paths also varies, see Fig. 3. Algorithm AV* achieved considerable speed-up when re-launched at v_{max} = 20 km/h. The fond paths by both algorithms when the speed was changed correspond to a path found by algorithm A*. It is necessary to note that the final path found by algorithm A* is not affected by a different speed.



(a) A* (b) AV*

Fig. 3. Found paths for test task 2.

3.3 Test Task 3

Figure 4 depicts a testing environment with 129 nodes. This test also uses a graph with a higher density of nodes with a difference that the starting and final node are placed close to each other. The objective is to verify that the algorithms do not search though a high number of nodes. Experimental results are stated in Table 3. In this experiment, there were changes in the dynamic part of the algorithm, which modifies re-use of previously acquired information.

Table 3. Experimental results for the 3rd testing environment

Criterion	A* v_{max} = 100 km/h	AV* v_{max} = 100 km/h	A* v_{max} = 20 km/h	AV* v_{max} = 20 km/h
Number of nodes	14	15	14	14
Path length (m)	515	533	515	531
Time to do the path (s)	26	24	86	80
Time of calculation (μs)	13277	40286	13145	25456
Memory (bit)	3402	4863	3402	5143

Increasing distance between the start and finish in algorithm AV* resulted in multiplied time requirements for the calculation. The character of the found paths is different, see Fig. 4.

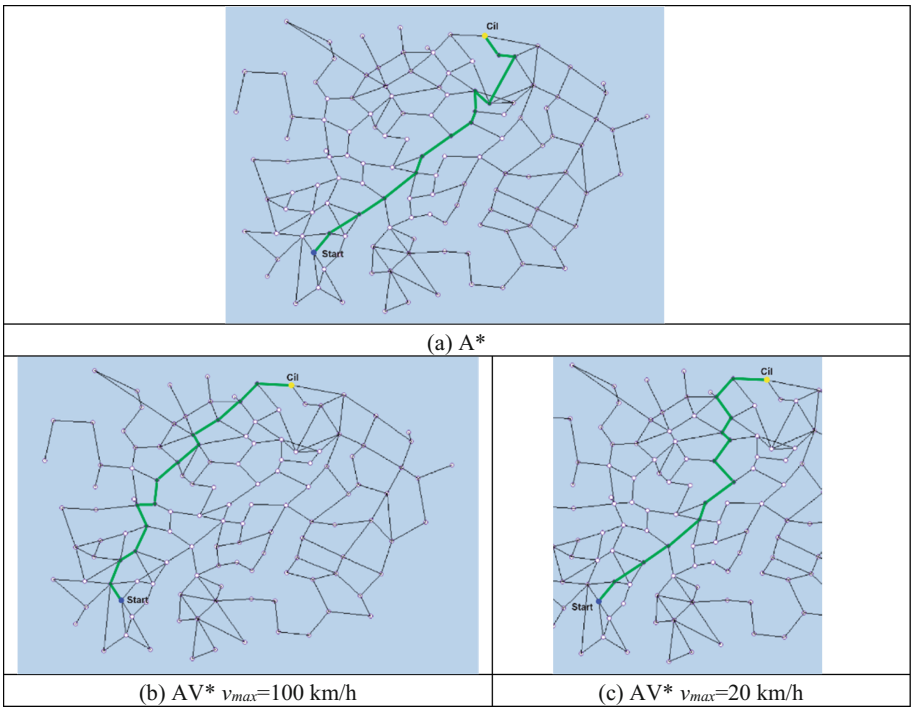


Fig. 4. Found paths for test task 3.

4 Conclusions

All tested algorithms succeeded in finding a path in all performed tests. The proposed algorithm AV* differs from algorithm A* by the amount of searched nodes due to repetitive search with a different value t_{cost}. The number of searched nodes depends on environment complexity and the position of the start and finish. There is a correlation between the number of found nodes and the character of the final path as the number of the found nodes serves for geographical orientation of the algorithm. It affect the algorithm run only from the point of view of memory and time of the calculation. On the contrary, the real length has direct influence on the character of the found path and it is crucial for both compared algorithms as it influences the choice of expansion in the next step. The time to go through the path represents the main criterion for algorithm AV* as it is proposed to its minimization. All tested algorithms succeeded in finding a path shorter in time.

Based on the results, it is possible to claim that the algorithm complies with the condition of finding a path between the start and finish in a finite time based on the criterion of the fastest alternative using the maximum speed limit possible. Considering dynamic optimization, all tests achieved a speed increase when the algorithm was repeated at a different speed. This speed-up corresponds to approximately 25% and it is linearly dependent on the increasing amount of the searched nodes.

Acknowledgments. The research described here has been financially supported by University of Ostrava grant SGS07/PřF/2017. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of the sponsors.

References

1. Algfoor, Z.A., Sunar, M.S., Kolivand, H.: A comprehensive study on pathfinding techniques for robotics and video games. Int. J. Comput. Games Technol. **2015**, 11 (2015). Article ID 736138
2. El Khaili, M.: Path planning in a dynamic environment. Int. J. Adv. Comput. Sci. Appl. **1**(5), 86–92 (2014)
3. Gavrilut, I., Tiponut, V., Gacsádi, A.: An integrated environment for mobile robot navigation based on CNN images processing. In: Proceedings of the 11th WSEAS International Conference on SYSTEMS, Agios Nicolaos, Crete, Greece, vol. 5117, pp. 81–86 (2007)
4. Jiang, B., Liu, X.: Computing the fewest-turn map directions based on the connectivity of natural roads. Int. J. Geogr. Inf. Sci. **25**(7), 1069–1082 (2011)
5. El Khaili, M.: Planning in a dynamic environment. Int. J. Adv. Comput. Sci. Appl. **5**(8), 86–92 (2014)
6. Kosík, D.: Pathfinding in dynamically changing environment (in Czech). Master thesis, University of Ostrava, Czech Republic (2017)
7. Kroumov, V., Yu, J.: Neural networks based path planning and navigation of mobile robots. In: Recent Advances in Mobile Robotics, pp. 174–190. INTECH Open Access Publisher (2011)
8. Lavalle, S.M.: Planning Algorithms, 1st edn. Cambridge University Press, New York (2006)
9. Narayanan, V., Phillips, M., Likhachev, M.: Anytime safe interval path planning for dynamic environments. In: Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura-Algarve, Portugal, pp. 4708–4715 (2012)
10. Raja, P., Pugazhenthi, S.: Path planning for a mobile robot in dynamic environments. Int. J. Phys. Sci. **6**(20), 4721–4731 (2011)
11. Zhou, Y., Wang, W., He, D., Wang, Z.: A fewest-turn-and-shortest path algorithm based on breadth-first search. Geo-Spatial Inf. Sci. **17**(4), 201–207 (2014)