Planning algorithm and its modifications for tactical decision support systems

Petr Stodola, Jan Mazal

***Abstract*—**This paper is divided into the two main parts. The first part deals with a planning algorithm being used in tactical decision support systems, which has been developed at the University of Defence in Brno. In the first part, there is presented improved versions of the original algorithm which are demonstrated while searching for an optimal path for a ground autonomous robot in a general environment. The article shows two different approaches for the algorithm improvement, along with their basic principles. The possibilities of the improvement are analyzed on two particular examples and the results of the new versions are compared with the original algorithm. In the second part, the article presents the issue of tactical decision support systems. The state of development of these systems is presented here, along with an example of their utilization.

***Keywords*—**Planning algorithm, optimal path-finding algorithm, autonomous vehicles, tactical decision support systems

I. Introduction

TACTICAL decision support systems have started being trends in combat management recently; all modern armies are interested in these new possibilities. Especially the US Army has been dealing with this issue deeply within its program Deep Green; it is a system for decision support intended for commanders of the US Army [1].

The purpose of those systems is to use computer technology for effective and accurate predictions of possible situation scenarios and to facilitate evaluation of potential results of commanders' decisions. Predictions are based on thorough analysis of a present situation considering huge amount of input parameters. The systems are characteristic by their advanced prediction capabilities.

The University of Defence in Brno also deals with these issues within its research program. Presently, there are several projects on using autonomous vehicles for increasing the efficiency of commanders in operations. One of the key projects is the development of a ground autonomous vehicle for reconnaissance and combat purposes. Within the project, we developed an experimental autonomous robot to demonstrate and verify algorithms and principles of autonomous motion in a general environment [2], [3], [4].

Manuscript received October 7, 2011.

P. Stodola is with the University of Defence, Brno, Czech Republic (e-mail petr.stodola@unob.cz).

J. Mazal is with the University of Defence, Brno, Czech Republic (e-mail: jan.mazal@unob.cz).

The article considers an improvement of the optimal path-finding algorithm which the robot uses during its movement from its current position to the target position (the algorithm is used also for other tasks of tactical decision support – see chapter VII). There are presented two different variants of the improvement in the article, along with deeper analysis and evaluation.

II. Original optimal path-finding algorithm

While moving, the robot uses the algorithm based on Floyd-Warshall principle with essential structural and optimization modifications which makes the algorithm computationally usable for vast data structures with more than 10^6 nodes. The algorithm is designed for parallel processing and is implemented on a graphical processor in CUDA integrated development environment [5].

The basic prerequisite for the algorithm is evaluation of all graph edges by non-negative integers (costs). An example of the algorithm function in the real terrain is shown in Fig. 1. Costs were set according to influence of several factors:

- terrain relief,
- vegetation,
- waters,
- roads,
- visibility of the threatening element.

Fig. 1 Example of the optimal path-finding algorithm function

Fig. 1 presents the path found from the initial position to the target position. In Fig. 2, there is shown the cost map for the graph edges from Fig. 1. Values are coded in gray shades.

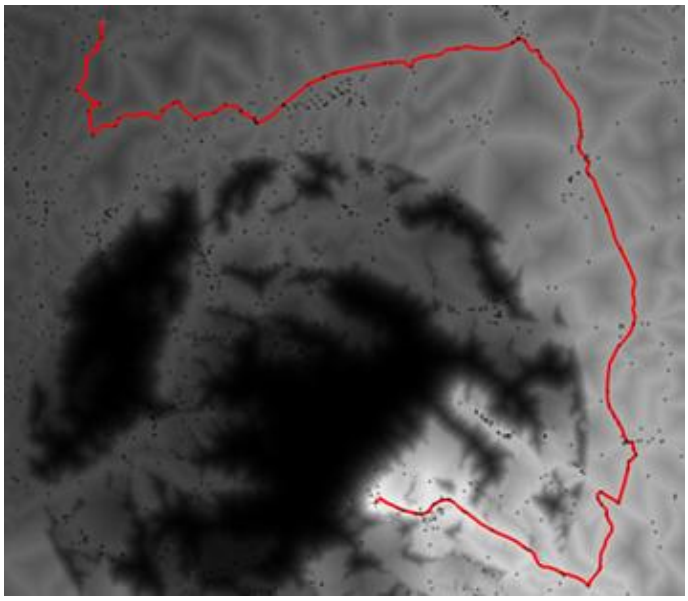

Fig. 2 Cost map for the environment from Fig. 1

The algorithm belongs among discrete methods; the environment is sampled by the Sukharev grid [6]. Adjacent nodes in the grid create graph edges on which the algorithm is applied. Every node has 8 edges to adjacent nodes according to Fig. 3.

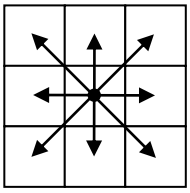

Fig. 3 Graph edges for each node in the grid

The algorithm is able to find all variants of the shortest paths when there are more of them. For illustration, Fig. 4 presents three variants of the shortest path from the initial point to the target point. All three paths are evaluated by the same cost. It is apparent that the Euclidean distance of the middle path is lesser than the distance of the remaining two. The main problem is the fact that the algorithm is not able to recognize the Euclidean shortest path what is a quite important criterion for optimal autonomous motion of unmanned vehicles.

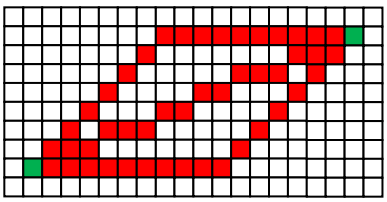

Fig. 4 Three variants of the shortest path

There are designed two new versions of the algorithms in the following text dealing with the presented issue.

III. Version with 16 edges from each node

The version in this chapter is very easy to implement since the only change in comparison with the original version is extending the number of edges from each node of the grid from 8 to 16. The principle is shown in Fig. 5 on the left. Fig. 5 on the right presents integer costs which are based on the Euclidean distances of individual edges.

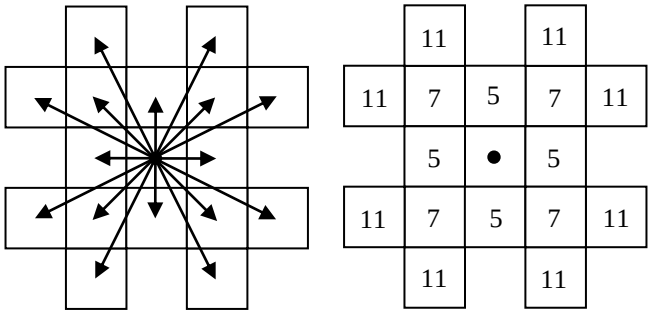

Fig. 5 Graph edges from each node in the grid and their costs

The principle of extending the number of edges does not solve the problem completely. It is only way how to reduce the problem slightly. Moreover, the double amount of edges results in the significant increase of the running time of the algorithm. Deeper evaluation of the method is presented in the following text.

IV. Version with additional analysis of the path

The second approach for the problem solution uses additional analysis of the path found by the original algorithm. When there are more paths (e.g. as in Fig. 4), then it is possible to use whichever one.

The principle consists in sequential processing of particular path sections and checking if the sections are possible to be shortened by lines with the shortest Euclidean distance. The algorithm is presented in Fig. 6 in pseudocode. The function `FindPath (path[], size, A, B)` searches for the shortest path from points `A` to `B` by the original algorithm. Points of this path are stored in array `path`; total number of points is in variable `size`.

```
1.  FindPath (path[], size, A, B)
3.  start = 0
4.  newsize = 0
5.  while ( start <= size )  do
6.    for ( i = start to size ) do
7.      if (TestPath(path[],start,i) == FALSE) do
8.        newsize += AddPath(newpath[],start,i-1)
9.        start = i
10.       break for
11. Store newpath, newsize
```

Fig. 6 The algorithm with additional analysis of the path found

In the algorithm, there are two functions `TestPath` and `AddPath`. The function `TestPath (path[], start, i)` examines the path section stored in array `path` between points with indexes `start` and `i`. The function returns TRUE if cost of the path section is the same as cost of the path given by a line between both points, or FALSE otherwise. The principle is demonstrated in Fig. 4 where the middle path is given by a line between two points and it has the same cost as both adjacent paths.

The function `AddPath (newpath[], start, i-1)` stores a new path section given by a line between points of the original path with indexes `start` and `i-1`. It returns the number of points added to the new path.

As a result of the algorithm, we have a new path with the same cost but with lesser (or the same at worst) Euclidean distance. The big advantage of the principle is its linear running time O (*size*) where *size* is the number of points of the path found by the original algorithm.

V. Evaluation and Comparison of New Versions

This chapter evaluates and compares the original algorithms with the new versions. Evaluation was taken place in our simulator of the experimental autonomous vehicle [7]. Fig. 7 shows the environment configuration where the analysis was conducted. The area is of size about 12 × 12 meters. Black squares represent obstacles in the area.

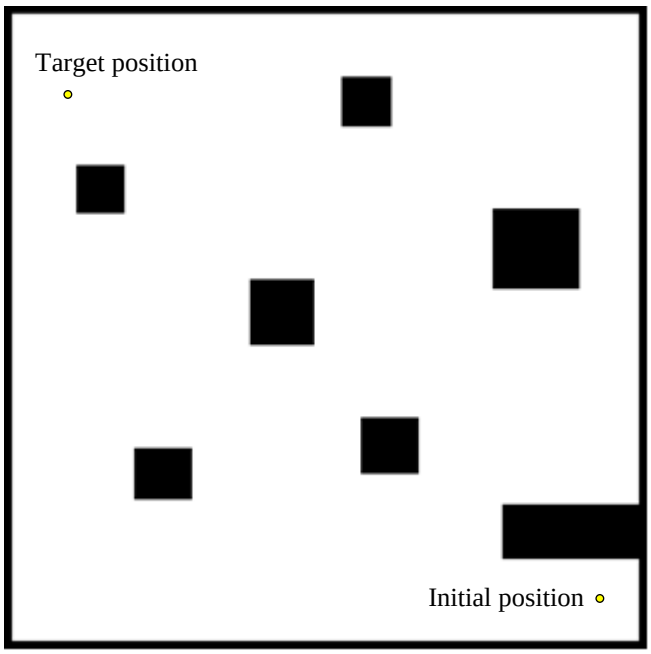

Fig. 7 Environment configuration for analysis

Fig. 8, 9, and 10 present paths of the autonomous vehicle acquired from the simulator. Green color shows the path from the initial to the target position; costs of individual graph nodes are coded in gray shades. The path computed by the original algorithm is in Fig. 8; Fig. 9 presents the path from the algorithms with 16 edges and Fig. 10 presents the path from the version with additional analysis. We can see progressive improvement of the results in figures.

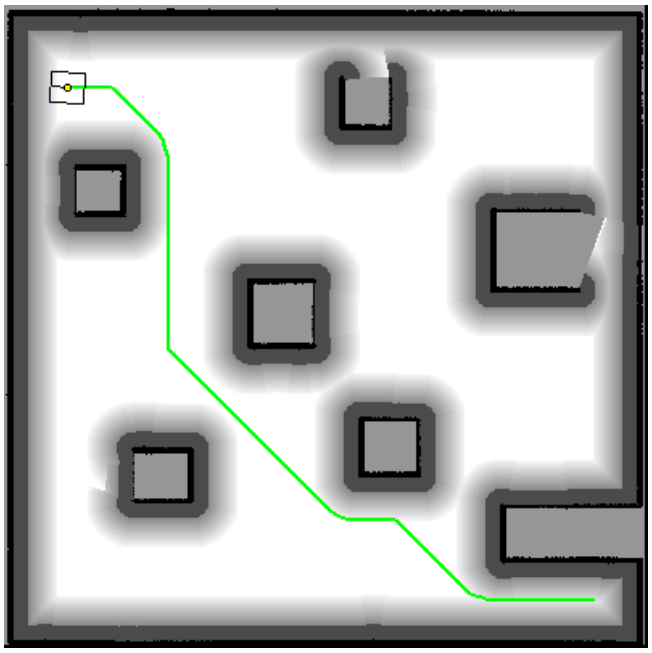

Fig. 8 Optimal path computed by the original algorithm

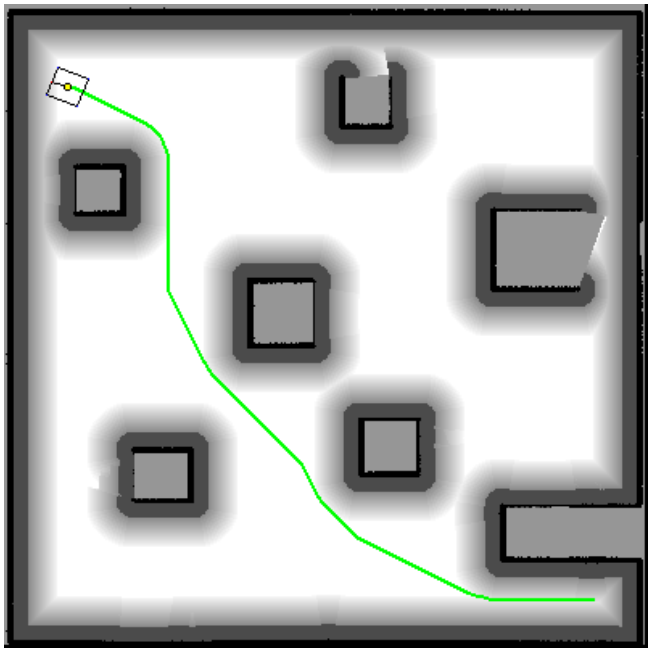

Fig. 9 Optimal path computed by the algorithm with 16 edges from each node

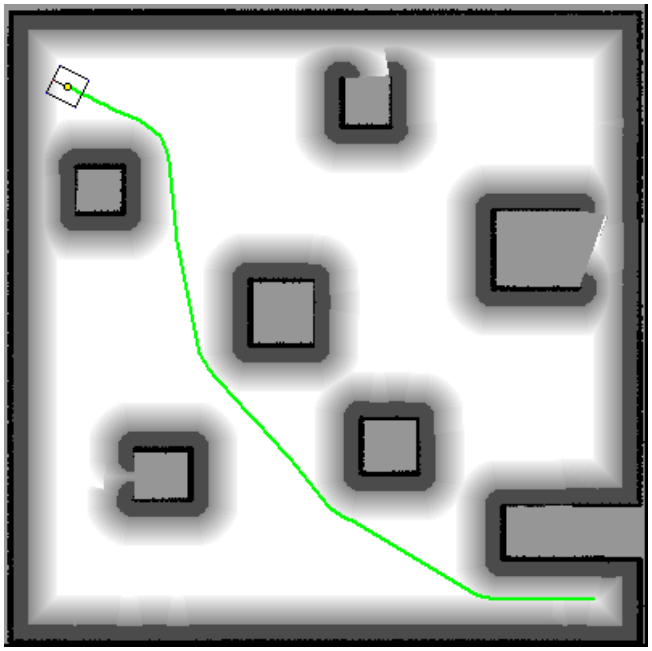

Fig. 10 Optimal path computed by the algorithm with additional analysis

Table I shows several parameters obtained from the experiments. It is apparent that the algorithm with 16 edges provided better results than the original algorithm; nevertheless the average running time was more than twice longer which was caused by the double number of edges. The total distance covered was about 2.5 % shorter. Number of algorithm launches was smaller as the algorithm was able to move by two points in one direction (see Fig. 5).

TABLE I
RESULTS OF THE ALGORITHM FROM THE PRESENTED EXPERIMENT

Parameter	Original algorithm	Algorithm with 16 edges	Algorithm with additional analysis
Number of graph edges	1,280,000	2,560,000	1,280,000
Number of algorithm launches	455	349	455
Average running time	57.6 ms	134.1 ms	58.2 ms
Total running time	26.2 s	46.8 s	26.5 s
Total distance covered by vehicle	15.70 m	15.31 m	15.17 m
Total distance in %	100 %	97.5 %	96.6 %

The algorithm with additional analysis of the path found seems to be the best option. The average running time was affected only insignificantly (about 0.6 ms); it is caused by the linear running time of additional analysis. The total distance covered was about 3.4 % shorter. Fig. 10 shows that there is the path with the Euclidean shortest distance.

The second experiment is more complicated and it is based on the real environment configuration shown in Fig. 11. There we can see 24 rectangular obstacles in the area of size about 17 × 29 meters, along with the positions of the initial and the target points.

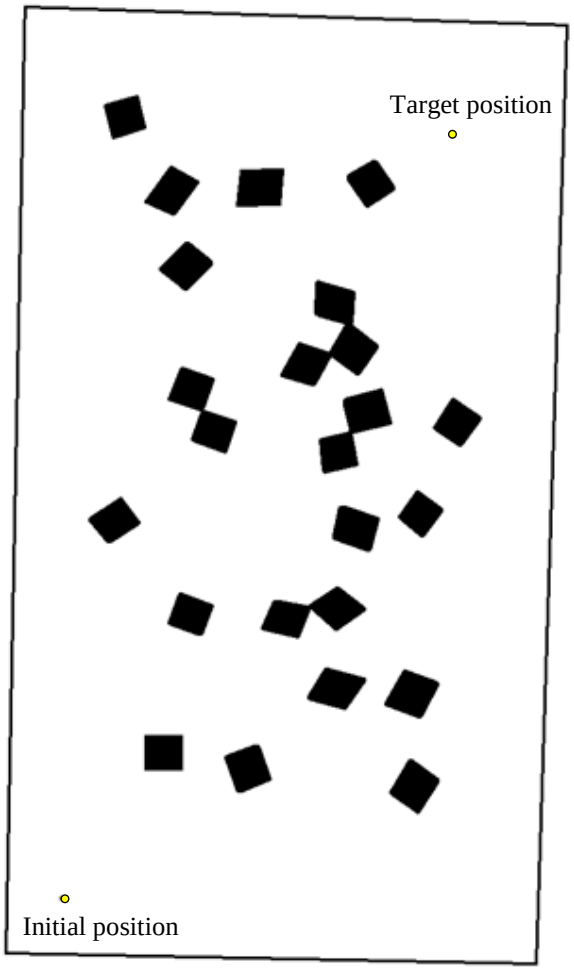

Fig. 11 The second example of environment configuration for analysis

This environment configuration was used for one of many real tests of our autonomous motion principle. We have implemented this principle on our experimental autonomous ground vehicle. The principles of autonomous motion are composed of several key processes [2]:

- **Localization process** – is a summary of activities, which provide an estimate of current vehicle location in an area where there is no GPS signal.
- **Environment reconstruction process** – by means of laser scanner the maps of surrounding environment are obtained in particular steps that are consequently incorporated into the overall environment map.
- **Optimal path searching process** - based upon the data analysis from the reconstructed maps, each point has its cost which serves as an input for our algorithm of the optimal path finding.

- **Motion control process** – in the last step the vehicle motion itself is carried out along a route found according to the vehicle mathematical motion model.

Fig. 12 presents the path along with the vehicle moved in the real test with the above mentioned principle implemented.

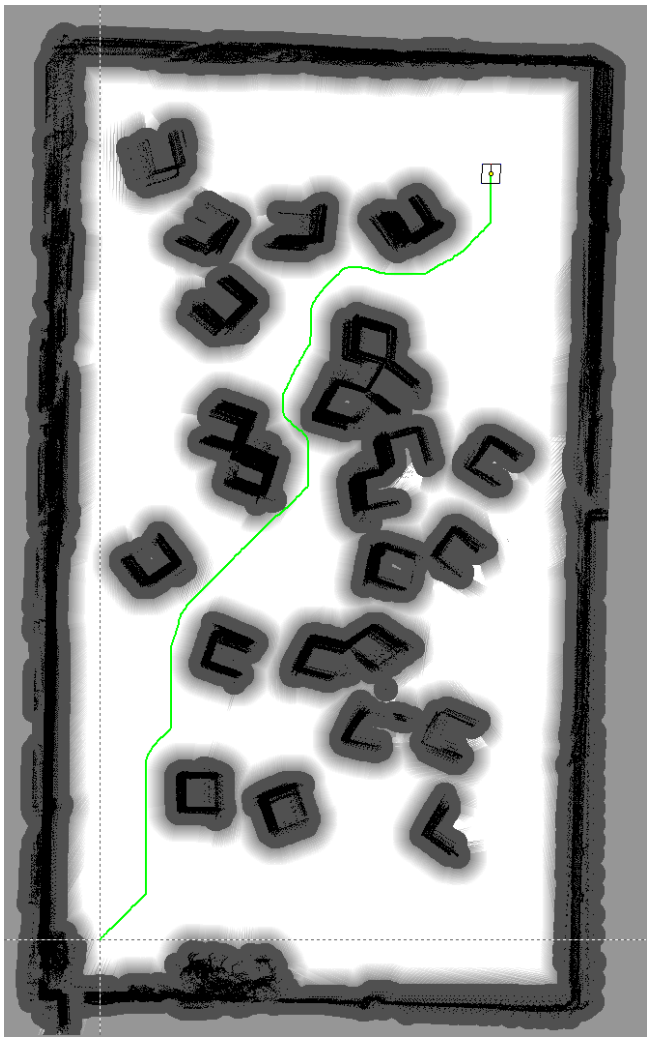

Fig. 12 Motion of the experimental vehicle in the real environment

The major problem in the real environment is the localization of the robot, especially the precise determination of its orientation; any small inaccuracy in determining the angle of orientation is substantially reflected in the reconstruction of the obstacles. This problem can be noticed in the experiment in Fig. 12.

Small errors in determining the orientation were caused by an incorrect synchronization of the time of position and orientation calculation of the vehicle by means of model and time of environment mapping by the laser scanner. Nevertheless, as the results show, this error does not have a significant impact on the overall function of the autonomous motion.

Fig. 13 shows a solution of the same task conducted in our simulator by the original version of the shortest-path algorithm. Both maps of the environment (that from the real space and that from the simulator) are very similar to each other, suggesting the correct implementation of principles of autonomous motion.

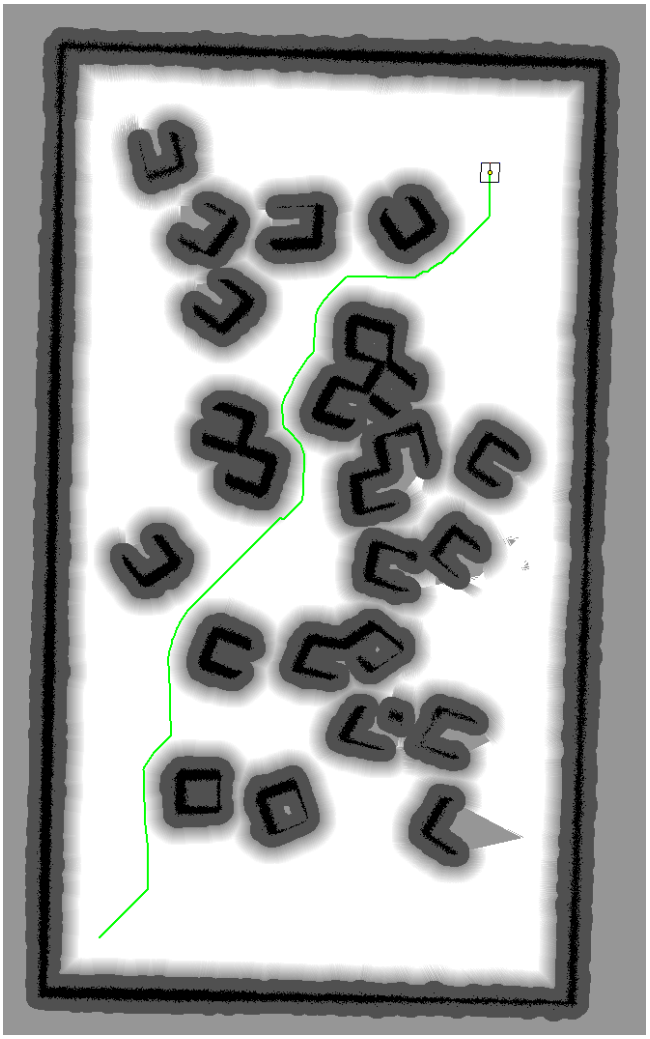

Fig. 12 Solution of the task computed by the original algorithm in the simulator

Fig. 14 presents a graph showing the mutual dependence of the distances run by the vehicle in real application, and in simulation tasks. The total distance of the path in the real experiment was 31.06 meters, whereas in the simulation 30.73 meters.

When conducting the experiment, we had only the original version of the algorithm, therefore we cannot show the same task computed by the new versions in the real environment. Nevertheless, we tried both new version of the algorithm in our simulator. Fig. 15 shows the path computed by the version with 16 edges from each node of the graph.

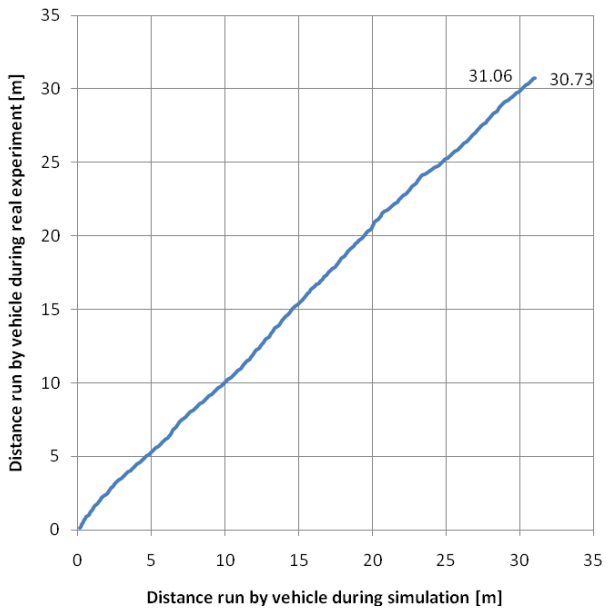

Fig. 14 Relationship between distance in real experiment and simulation

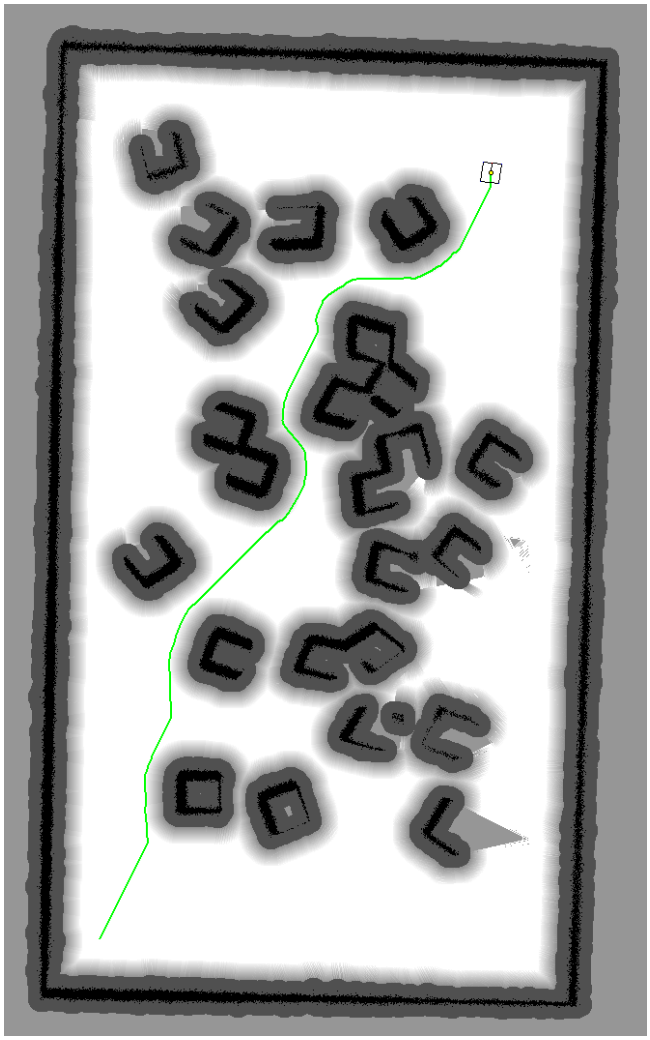

Fig. 15 Optimal path computed by the algorithm with 16 edges from each node

Fig. 16 shows the path computed by the version of the algorithm with additional analysis.

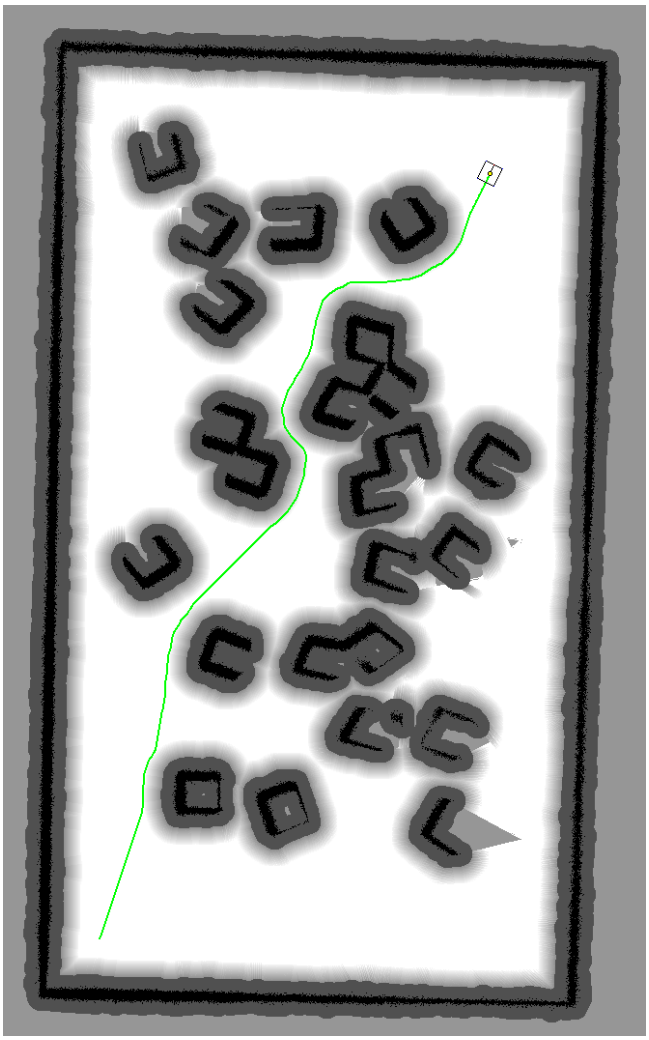

Fig. 16 Optimal path computed by the algorithm with additional analysis

TABLE II
RESULTS OF THE ALGORITHM FROM THE SECOND EXPERIMENT

Parameter	Original algorithm	Algorithm with 16 edges	Algorithm with additional analysis
Number of graph edges	5,693,512	11,387,024	5,693,512
Number of algorithm launches	295	219	295
Average running time	224.3 ms	540.9 ms	238.3 ms
Total running time	66.2 s	118.5 s	70.0 s
Total distance covered by vehicle	30.73 m	29.87 m	29.69 m
Total distance in %	100 %	97.2 %	96.6 %

Table II presents parameters for the second experiment. There were more than 5 million nodes in the original algorithm and in the version with additional analysis. The average running time was longer than 200 ms per one launch. The additional analysis took only about 14 ms on an average.

The algorithm with 16 edges in each node had more than 11 million nodes in the graph. The average running time was more than 0.5 sec per one launch. The total distance covered by the vehicle was about 2.8 % shorter in case of the algorithm with 16 edges and 3.4 % in case of the algorithm with additional analysis. These values are very similar to the values in the previous experiment.

VI. Tactical Decision Support Systems

The next part of this article deals with the issue of tactical decision support systems and its utilization in modern armies. As already mentioned, these state-of-the-art systems have started being trends in combat management recently. As an example can be mentioned the Deep Green concept (see Fig. 17), still in process of development by the US Army (DARPA since 2008), partly inspired by the honored predecessor Deep Blue (the software running on the supercomputer developed by IBM that addresses a chess game).

Fig. 17 The Deep Green concept [1]

US Army is committed to the application of artificial intelligence methods for processes of command and control, i.e. specially keeping sustainable support for the decision-making processes of the commanders, where the optimal reaction of the commander is constantly recalculated and then submitted as a proposal, which could be intuitively further developed.

It is not an effort to replace the role of the commander by computer, but there is the intention to automate most of "routine and rough" analytical work and thus prepare the design concept for new custom variants of solutions (COA-course of action).

On the basis of fundamental similarities of the chess game and tactical tasks, there are carried out experiments to solve operational and tactical tasks in a manner based on a similar approach. Despite the fact the complexity of the conditions in the real environment cannot be compared with the exact rules of the desk games, from a philosophical point of view there exists a close similarity and this similarity may be based on fundamental concepts and strategies solving operational and tactical tasks.

Because of the appropriate approximation degree of mathematical model so many of the tactical tasks is now solvable in a real time. That was unable in the past, because of the computer systems performance. The method of solution converges not only to purely static operational and tactical tasks, but it moves to the context of the dynamic development alternatives, which are relevant at the time of the tactical situation changes.

VII. Example of the Tactical Task

This chapter deals with an example of utilization of our algorithm in a frequent task to be solved in tactical decision support systems. It is a search for optimal location selection for an implementation of an ambush.

To solve this task you need to start from a particular math model of the environment, which in our case may be a 3D array, or a set of multiple 2D arrays, where each layer defines a particular characteristic. In our model, we consider the types of objects as follows:

- Vegetation,
- Altitude model,
- Water obstacles,
- Communication.

Furthermore, we expect that the impact will be made approximately on the same strong opponent and consider for the time being only two custom tactical elements, one conducting the attack and the second performing security tasks. This is the problem of multi-criterion objective, where it is necessary to lay down or implement the default priorities, continuity and facts in a solution, namely:

- The key is the location of the strike element and security element reacts to the position of the strike element and enemy object.
- The set of solutions is applied only to a set of expected positions of enemy object.
- Calculations and analysis are based on data that are currently available and which was quantified and described in the model; if some of the important aspects are not integrated into the model, the resulting solution may not correspond to the reality.
- Criteria and priorities from the perspective of the parameters of the solutions are chosen by the commander and their settings have a decisive influence on the applicability of the solution.

A system that resolves this job, works on the principle of the probability and the results of a solution calculation creates

conditions leading to the highest probability to complete the task, therefore, this system will not ensure domination, but only suggests, where is the optimal location of an element and what would be its optimal behavior. The solution consists of several layers of conditional integration, as shown in Fig. 18.

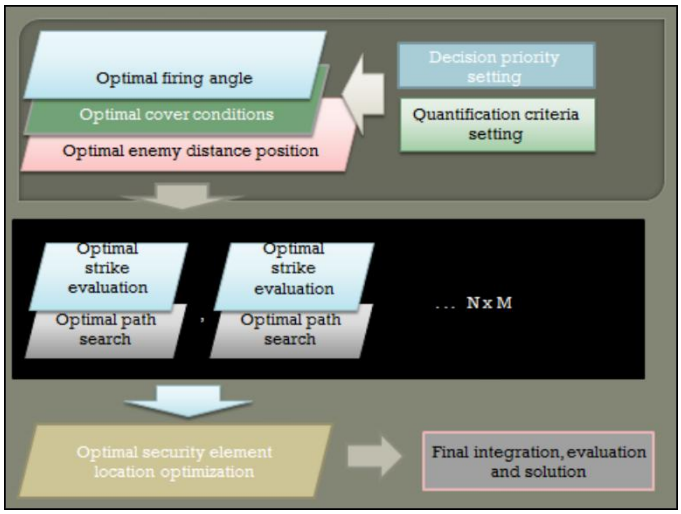

Fig. 18 Sequence of individual processes for the task of finding the optimal solution for the implementation of an ambush

Setting the input criteria of solution (decision) is conceived in the part of the integration model of quantification of input characteristics, whose description would significantly exceed the framework of this article. However, in the general overview it is a set of models of the multi-dimension functions, and their task is to incorporate the influence of external conditions and characteristics to the numeric form of the set of pragmatic coefficients, which are then applied with mathematical methods and transformations leading to the final model construction, in which to find the optimal solution is already trivial.

Fig. 19 presents the result of the task solution in a real terrain. The yellow circle is the initial position of our unit. The upper blue area in the red circle is the best position to attack the enemy; the bottom red circle represents the position designed to secure our units.

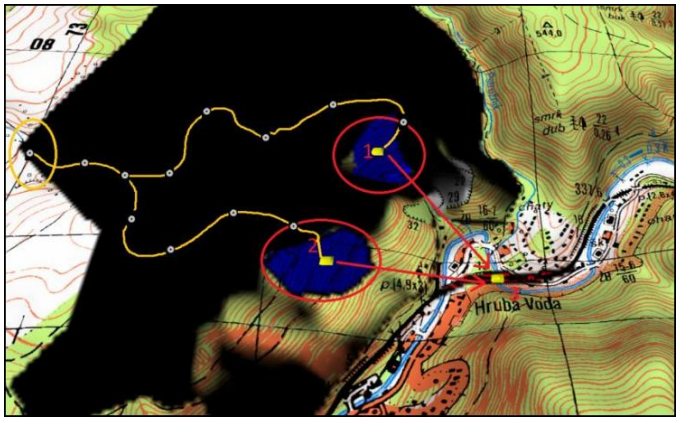

Fig. 19 Result of a task solution in a real terrain

VIII. Conclusion

This paper deals with issues of improving the shortest path-finding algorithm in discrete state space. We designed two improving variants; both were analyzed and compared on the two particular examples in our simulator of the experimental autonomous vehicle.

The obtained results are valid only for the selected examples. Results can differ in other situations. Nonetheless, the both versions of the algorithm ensure the same or shorter path in comparison with the original algorithm. The best results are provided by the algorithm with additional analysis of the path.

An advantage of the algorithm with 16 edges from each node is its easy implementation. On the other hand the running time is more than double. The additional analysis affects the total running time negligibly as the running time of the analysis is linear $O(n)$.

In the second part of the article, there are introduced the possibilities and importance of tactical decision support systems. Optimization of tactical activities, although it is not apparent at the first sight, is subjected to the algorithmic schema and, therefore there is wide possibility of their automation as shown in the presented example.

References

[1] J. R. Surdu, K. Kittka, "The Deep Green Concept". Spring Simulation Multiconference (SpringSim08). Otawa, 2008, pp. 623-631, ISBN 1-56555-319-5.

[2] P. Stodola, J. Mazal, "Optimal Location and Motion of Autonomous Unmanned Ground Vehicles". WSEAS Transactions on Signal Processing, vol. 6, no. 2, 2010, pp. 68-77, ISSN 1790-5052.

[3] P. Stodola, "Extended Motion Model of Autonomous Ground Vehicle". International Journal of Mathematics and Computers in Simulation, vol. 5, no. 1, 2011, pp. 28-35, ISSN 1998-0159.

[4] P. Stodola, J. Mazal, M. Podhorec, M. Ovesny, "3D Laser Scanning System for Experimental Autonomous Robot". International Conference on Military Technologies 2011 (ICMT11), Brno: University of Defence, 2011, pp. 983-988, ISBN 978-80-7231-787-5.

[5] J. Mazal, "Real Time Maneuver Optimization in General Environment". Recent Advances in Mechatronics 2008 – 2009, Springer Berlin Heidelberg, 2009, pp. 191-196, ISBN 978-3-642-05021-3.

[6] M. S. LaValle, "Planning Algorithms". University of Illinois, 2006, 842 p.

[7] P. Stodola, J. Mazal, M. Ovesny, R. Kremiec, "Simulator of Real Experimental Autonomous Robot". International Conference on Military Technologies 2011 (ICMT11), Brno: University of Defence, 2011, pp. 989-996, ISBN 978-80-7231-787-5.

[8] S. Behzadi, A. A. Alesheikh, "Developing a Genetic Algorithm for Solving Shortest Path Problem". WSEAS International Conference on Urban Planning and transportation (UPT07), Heraklion, 2008. pp. 28-32, ISBN 978-960-6766-87-9, ISSN 1790-2769.

[9] S. Ebrahimnejad, R. Tavakoli-Moghaddam, "Solving the fuzzy shortest path problem on networks by a new algorithm". WSEAS international Conference on Fuzzy Systems (FS09), Stevens Point, 2009, pp. 28-34, ISBN 978-960-474-066-6, ISSN 1790-5109.

[10] M. K. Jha, G. A. Karri, M. A. Kang, "Military Path Planning Algorithm Using Visualization and dynamic GIS". WSEAS International Conference on Computer Engineering and Applications (CEA10), Stevens Point, 2010, pp. 188-193, ISBN 978-960-474-151-9, ISSN 1790-5117.