ISSN 1691-5402
Environment. Technology. Resources
Proceedings of the 9th International Scientific and Practical Conference. Volume 1I
© Rēzeknes Augstskola, Rēzekne, RA Izdevniecība, 2013

Path Planning Usage for Autonomous Agents

Edvards Valbahs, Peter Grabusts
Rezekne Higher Educational Institution, Faculty of Engineering, Atbrivoshanas alley 76, Rezekne, LV-4601, Latvia.

***Abstract*. In order to achieve the wide range of the robotic application it is necessary to provide iterative motions among points of the goals. For instance, in the industry mobile robots can replace any components between a storehouse and an assembly department. Ammunition replacement is widely used in military services. Working place is possible in ports, airports, waste site and etc. Mobile agents can be used for monitoring if it is necessary to observe control points in the secret place. The paper deals with path planning programme for mobile robots. The aim of the research paper is to analyse motion-planning algorithms that contain the design of modelling programme. The programme is needed as environment modelling to obtain the simulation data. The simulation data give the possibility to conduct the wide analyses for selected algorithm. Analysis means the simulation data interpretation and comparison with other data obtained using the motion-planning. The results of the careful analysis were considered for optimal path planning algorithms. The experimental evidence was proposed to demonstrate the effectiveness of the algorithm for steady covered space. The results described in this work can be extended in a number of directions, and applied to other algorithms.**

***Keywords* – robotic, Simulated Annealing, path planning.**

I INTRODUCTION

The article is connected to the travelling salesman problem (TSP), but with some exceptions and conditions. In the case when the TSP is envisaged the following approximate path planning algorithms are used [2, 3, 4]:

- The closest neighbour algorithm;
- Simulated Annealing (SA);
- Genetic Algorithm (GA);
- Ant colony optimization.

The closest neighbour approach is the simplest and straightforward TSP one [10]. The way to this approach to always visit the closest city. The polynomial complexity of the approach is O(n^2). The algorithm is relatively simple:

1 – Choose a random city;

2 – Find out the nearest city unvisited and visit it;

3 – Are there any unvisited cities left? If yes, repeat step 2;

4 – Return to the first city.

SA is successfully used and adapted to get an approximate solutions for the TSP [10]. SA is basically a randomized local search algorithm similar to *Tabu Search* but do not allow path exchange that deteriorates the solution. The polynomial complexity of the approach is O(n^2) accordingly.

```
Input: ProblemSize, $iterations_{max}$, $temp_{max}$
Output: $S_{best}$
1.  $S_{current}$ ← CreateInitialSolution(ProblemSize)
2.  $S_{best}$ ← $S_{current}$
3.  for $i = 1$ to $iterations_{max}$ do
4.    $S_i$ ← CreateNeighborSolution($S_{current}$)
5.    $temp_{curr}$ ← CalculateTemperature($i$, $temp_{max}$)
6.    if Cost($S_i$) ≤ Cost($S_{current}$) then
7.      $S_{current}$ ← $S_i$
8.      if Cost($S_i$) ≤ Cost($S_{best}$) then
9.        $S_{best}$ ← $S_i$
10.     end
11.   else if Exp((Cost($S_{current}$)-Cost($S_i$))/$temp_{curr}$) > Rand() then
12.     $S_{current}$ ← $S_i$
13.   end
14. end
15. return $S_{best}$
```

Fig. 1. Pseudocode for SA

The SA method [1, 5, 16] is widely used in applied science (Fig. 1). The well-known traveling salesman problem has effectively solved by means of this method. For instance, the arrangement of many circuit elements on a silicon substrate is considerably improved to reduce interference among the elements [15, 18].

GA conducts in a way similar to the nature [3]. A basic GA starts working with a randomly generated population of potential solution. The candidates are then mated to produce offspring and only some of them go through a mutating process. Each candidate has an optimal value demonstrating us how go it is. Choosing the most optimal candidates for mating and mutation the overall optimality of the candidate solutions increases. Using GA to the TSP involves implementing a crossover procedure, a measure of optimality and mutation as well. Optimality of the solution is a length of the solution.

Ant colony optimization is the algorithm that is inspired by the nature [9]. It is based on ant colony moving behaviour. Good results can be achieved by means of the algorithm but not for complex problems.

We managed to use SA algorithm rather successfully in our previous work [17] taking into account the specific side of this work (it will be discussed in detail further). Therefore, it is necessary to discuss some principles of SA realization in detail. In order to calculate the total path it is necessary to know the shortest route among all the cities. As we do not know the distance, we must apply one of the algorithms to define the shortest route among all the cities. It is Dijkstra's algorithm [14] that gives the possibility to get the shortest path tree. The polynomial complexity of the Dijkstra's algorithm is O(n^2).

II Goals

The aim of the research paper is to analyze motion-planning algorithms that contain the design of modelling programme. The programme is needed as environment modelling to obtain the simulation data. The simulation data give the possibility to conduct the wide analyses for selected algorithm. Analysis means the simulation data interpretation and comparison with other data obtained using the motion-planning.

The use in practice and the necessity of it is greatly connected to optimal algorithm and methodological work out for autonomous agents that move in the space and are able to plan route on their own [6, 7, 8, 11, 12, 13]. One of such agent-samples exiting in our everyday life is autonomous vacuum cleaner. Autonomous vacuum cleaners do not usually use the motion-planning algorithm. They are based on some simple algorithms, for example cleaning in a spiral, crossing the premises avoiding the walls and their moving is casual after touching the walls. The philosophy of this design was offered by the scientists of Massachusetts Institute of Technology. Agents must behave as insects having primitive controlling devices in accordance to the environment. As a result, though an autonomous vacuum cleaner is very effective in cleaning premises, it is required much more time as compared with work made by a human. There is a drawback, the autonomous vacuum cleans some spaces many times but other spaces only once. The use of motion-planning algorithms can raise the effectiveness of an autonomous vacuum cleaner.

III Assumptions

In order to fulfill the aim of the research paper the following conditions are introduced for:

- premises where an object moves;
- robot (or object) moves around the premises;
- path the robot moves on in the premises.

The premises are presented as two-dimensional plane. The plane of premises is equally divided into the cells. The cell dimensions are equal to agent dimension that moves in the premises. The space can be represented as a graph with two kinds of edges (see Fig. 2). Horizontal and vertical edges are marked with unbroken lines they are of similar length, but others are longer and marked with dash lines. It is linked with agent movement possibilities.

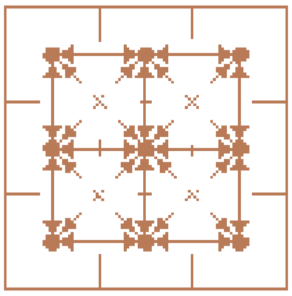

Fig. 2. The example of the graph and 3 x 3 space

The object moves only one cell forward and back i.e. during one motion the object can move to the one cell from empty eight ones (eight cells around one cell) paying attention to that cell is not occupied by the obstacle but if it is occupied, the graph will not have the relevant vertex (see Fig. 3).

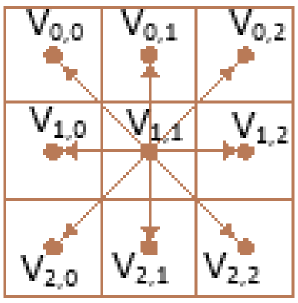

Fig. 3. The example of agent's motion (where $v_{i,j}$ is relevant vertex)

As opposed to classical TSP we take a number of additional rules and it means that the agent can cross the one the same cell several times in succession (it must cross any cell one time obligatory). Thus, the agent's initial vertex does not coincide with its final vertex of total route.

In this research paper both algorithms were compared practically using and combining different placement of obstacles in the unchangeable two-dimensional space. All the results were obtained on one and the same computer (2.66 GHz processor and 2GB RAM), operating systems (Ubuntu 12.04.1 LST Linux were used). The following information was collected about:

- the number of covering for each cell;
- the time which was necessary for both algorithms to plan the route.

The given illustrations (see Fig. 4) show coverage density (it is an example that was obtained in our previous work [17]).

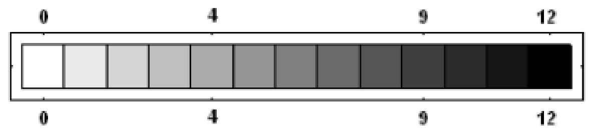

Fig. 4. Density scale (white - uncovered; black - covered the most)

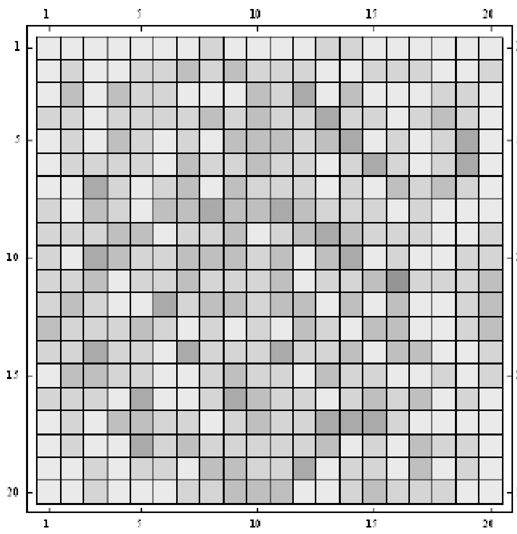

Fig. 5. Coverage density for the space without obstacles for SA

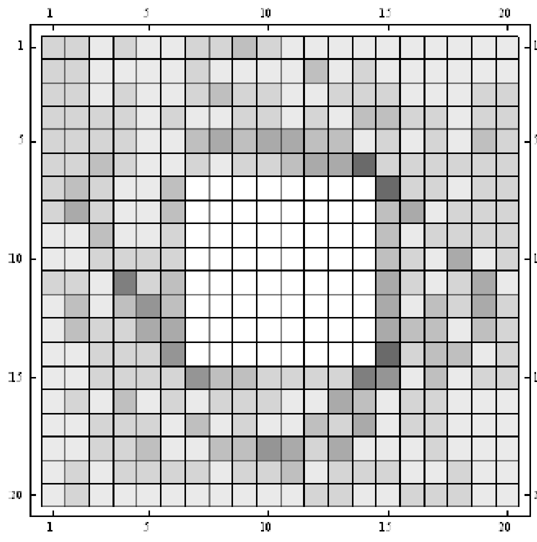

Fig. 6. Coverage density with the obstacle consisting of 64 cells (the obstacle is in the middle of the premises) for SA

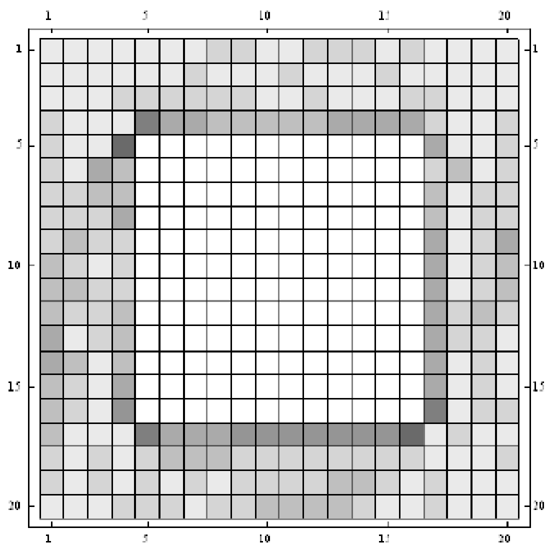

Fig. 7. Coverage density with the obstacle consisting of 144 cells (the obstacle is in the middle of the premises) for SA

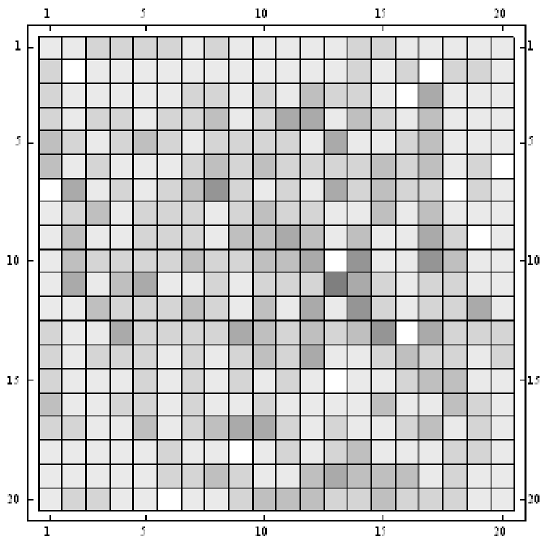

Fig. 8. Coverage density with the 12 random obstacles for SA

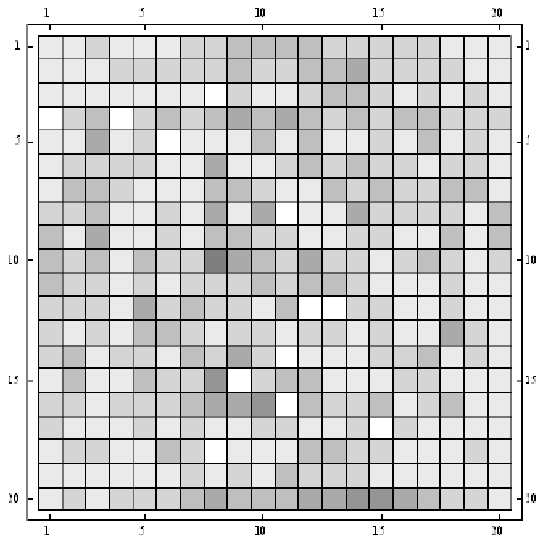

Fig. 9. Coverage density with another set of the 12 random obstacles for SA

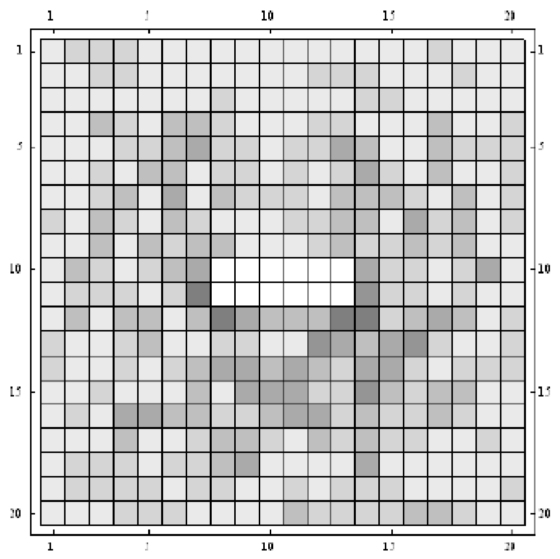

Fig. 10. Coverage density with the obstacle consisting of 12 cells (the obstacle is in the middle of the premises) for SA

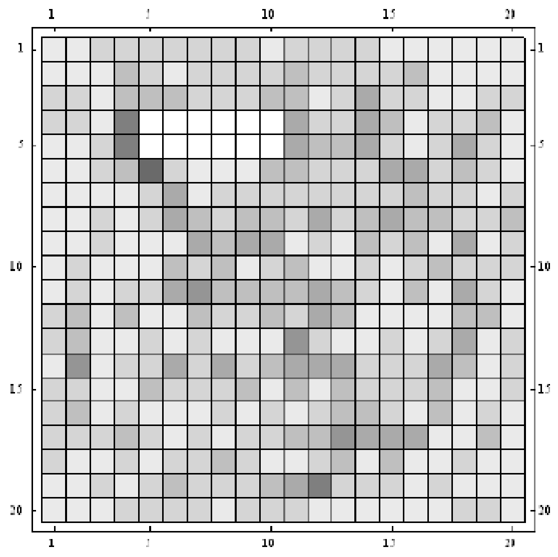

Fig. 11. Coverage density with the obstacle consisting of 12 cells for SA

The density scale (see Fig. 4) is the same for all coverage densities. Coverage density shows how often the robot covers each cell.

IV RESULTS

Taking into account the fact that the distance among all the vertexes (cities) are unknown in the beginning, it is necessary to define the shortest paths among those vertexes mentioned above. Dijkstra's algorithm can be used but increasing the measures of the premises, the time is proportionally increases accordingly that is necessary for evaluating path tree. Therefore, it is needed to simplify the calculation of the shortest path, which is possible, provided the peculiarities and nuances of the task are taken into consideration. In addition, the empty premises should be observed. If all the mentioned above remains valid, the simple algorithm can be worked out to define the shortest paths.

Let us consider the agent's general moving paths. If there are no vertexes between the current initial and goal vertexes, the agent can move only to eight possible positions (cells) depending on goal vertex (see Fig. 3). Admitting that first vertex index i defines the vertical position and the second vertex j defines the horizontal position we can draw a line either horizontally or vertically. And one of the vertexes will have the index with common value (see Fig. 12).

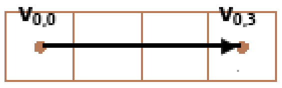

Fig. 12. The example of agent moving horizontally (where i index value is common for both vertexes)

Another situation can be seen if the current initial and goal vertexes are neither on the horizontal nor vertical lines (see Fig. 13-15).

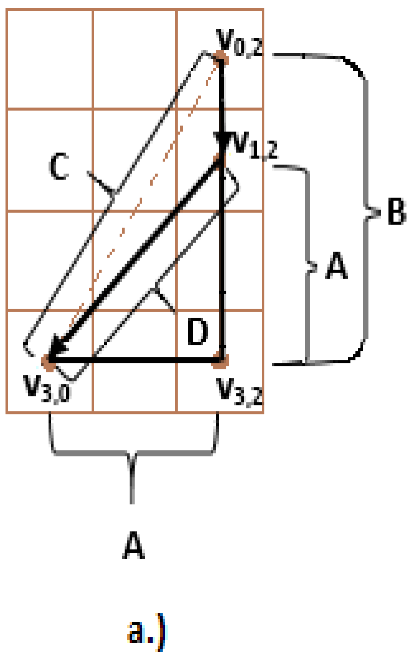

Fig. 13. Three examples of agent moving (where A, B and C are sections among the vertexes): agent moves from $v_{0,2}$ to $v_{3,0}$ crossing $v_{1,2}$

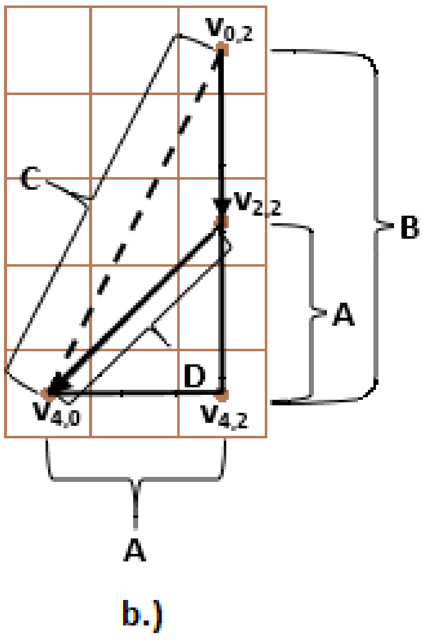

Fig. 14. Three examples of agent moving (where A, B and C are sections among the vertexes): agent moves from $v_{0,2}$ to $v_{4,0}$ crossing $v_{2,2}$

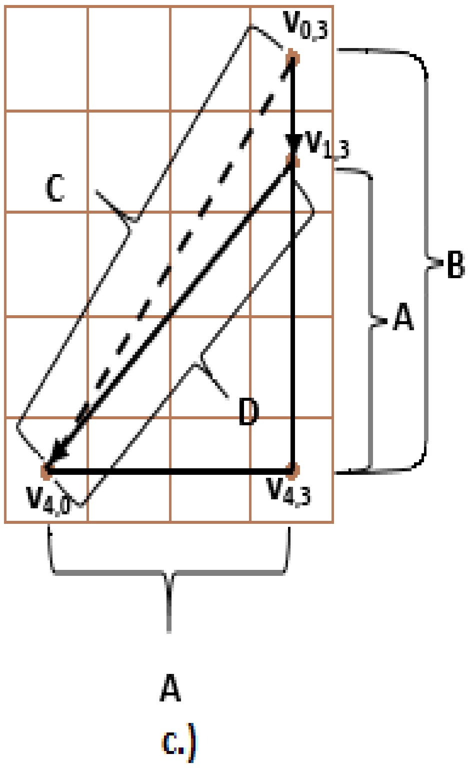

Fig. 15. Three examples of agent moving (where A, B and C are sections among the vertexes agent moves from $v_{0,3}$ to $v_{4,0}$ crossing $v_{1,3}$)

All cases of Fig. 13-15 have common characteristics that unites them. The shortest path from initial vertex to goal vertex is section C but for the agent this path is unavailable because of current task conditions and peculiarities. These cases can be described by the right-angled triangle where C is a side of the triangle. In addition, side B is longer than side A. One of the shortest paths among the relevant (corresponding) vertexes:

- the agent moves along the longest side B of the right-angled triangle until the gap between the covered path and side B is equal to side A;
- if gap between the covered path and side B is equal to side A, then the agent moves along the angle allowed (along the section D) to the goal vertex (let us mark that this action corresponds to the case when side B is equal to side A i.e. the right-angled triangle is the isosceles triangle, too (see Fig. 13) in case initial vertex is $v_{1,2}$, 4, (see Fig. 14) in case initial vertex is $v_{2,2}$ and (see Fig. 15) in case initial vertex is $v_{1,3}$)).

We can follow that the path is longer than optimal side C. And it can be calculated by the use of following formulae: $L = B-A+2^{0.5}*A$, where L is the length of the path from initial vertex to goal vertex. By turn, C can be calculated from $C = (A^2+B^2)^{0.5}$. It is possible to calculate how match percent L is longer than C (if L is equal to 100 %), then the final result is equal to $P=((L-C)*100)/L$. Our goal premises are 100 x 100 cells. The value of P is reflected with contour line for the given premises depending on A and B (see Fig. 16).

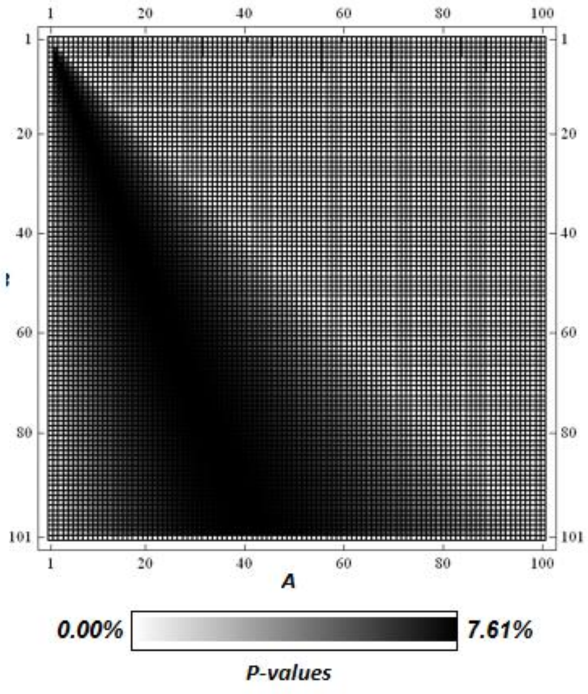

Fig. 16. P value depending on B and A, if A > 1 and B > A

It is possible to calculate maximum P value for 100 x 100 cells big premises (see Fig. 16) that is equal to 7.61 %. The method/algorithm mentioned was applied instead of Dijkstra's algorithm to calculate total path or covering of 100 x 100 cells big premises and it is obstacles free (see Fig. 17).

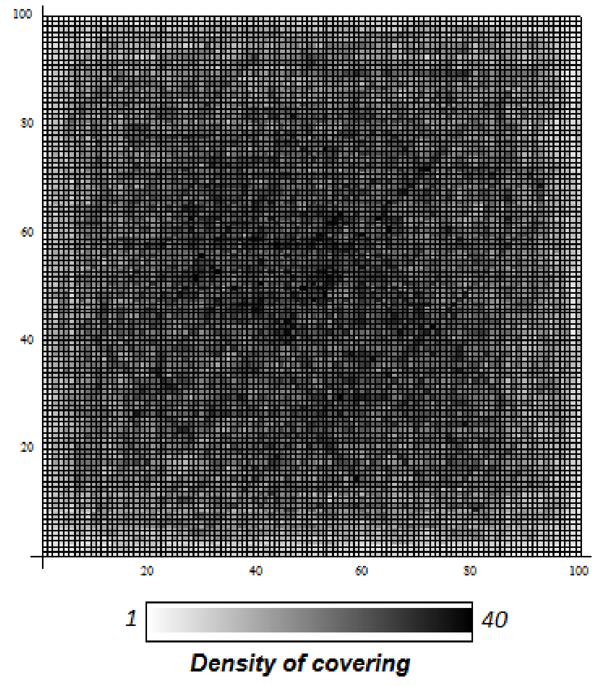

Fig. 17. The density of covering for 100 x 100 cells big premises (it is obstacles free)

Density of covering changes from 1 up to 40 (there are the cells which were covered only once and there are the cells covered maximum 40 times). Totally, the agent performed 192666 steps in order to cross each cell of the premises.

V Conclusion

It can be concluded that the algorithm offered is rather simple and it replaced Dijkstra's algorithm effectively according to the task. The algorithm allows decreasing the time of calculation, which is necessary to define the shortest route among graph vertexes.

The shortest path can be defined in a simple way (even in such cases mentioned in Fig. 13-15), provided that it is necessary to know the gap between the indexes of initial and goal vertexes. For instance, if initial vertex is $v_{i1,j1}$ and goal vertex is $v_{i2,j2}$, the first gap is $\Delta_1=|i_1-i_2|$ and the second gap is $\Delta_2=|j_1-j_2|$. As to the next step, it is needed to calculate the biggest gap between both the gaps. The shortest path is equal to the biggest gap. For instance, Fig. 13 reflects the shortest path which occupies 4 cells, but in other cases (see Fig. 14-15) it is 5 cells big.

It must be marked that total path can be a bit longer it is connected to the specific task which was envisaged in the chapters “Assumptions” and “Results” in detail. The worst case can be evaluated theoretically for the premises of 100 x 100 cells. If we take into consideration that the total route will consist of path pieces, which are longer than 7.61 % in comparison with C value (see Fig. 16), the total path will be longer than optimal 7.61 % (actually, it is the worst maximal variant. We must pay attention to the fact that SA provides only approximate solution).

The algorithm can be successfully used e.g. in autonomous public transport restricted by means of rules, technical requirements in autonomous robots and military equipment. In addition, the algorithm can be used in various computer games where a path planning is done in dynamic environment.

It is possible to conclude that the algorithm offered can be used in the different application areas not only for path planning of a robot.

VI References

[1] E. Aarts and J. Korst. Simulated annealing and Boltzman machines: A stochastic approach to combinatorial optimization and neural computing. John Wiley and Sons, 1989.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal and W.J. Cook, The Traveling Salesman Problem, Princeton University Press, Princeton, USA, 2007.

[3] W. J. Cook, *In Pursuit of the Traveling Salesman*. Princeton University Press, Princeton, USA 2011.

[4] D. Davendra, *Traveling Salesman Problem, Theory and Applications*. InTech, Rijeka, Croatia, 2010.

[5] R.H.J.M. Otten and L.P.P.P. Ginneken, *The Annealing Algorithm*. Kluwer Academic Publishers, 1989.

[6] R. Siegwart, I. R. Nourbakhsh and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. A Bradford Book The MIT Press Cambridge, Massachusetts London, England, 2011.

[7] P. H. Batavia and I. Nourbakhsh, *Path planning for the Cye personal robot, IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, 2000.

[8] R. Biswas, B. Limketaki, S. Sanner and S. Thrun, *Towards Object Mapping in Dynamic Environments with Mobile Robots, Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

[9] M. Dorigo and L. M. Gambardella, “Ant Colonies for the Traveling Salesman Problem,” University Libre de Bruxelles, Belgium, 1996.

[10] D. S. Johnson and L. A. McGeoch, “The Traveling Salesman Problem: A Case Study in Local Optimization.” in E. H. L. Aarts and J. K. Lenstra (editors), John Wiley and Sons, Ltd., 1997, pp. 215-310.

[11] V. Ashkenazi, D. Park and M. Dumville, “Robot Positioning and the Global Navigation Satellite System,” Industrial Robots: An International Journal, 27(6), pp. 419-426, 2000.

[12] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos and S. Thrun, “The Mobile Robot Rhino,” AI Magazine, 16(1), 1995.

[13] H. Choset, “Coverage of Known Spaces: The Boustrophedon Cellular Decomposition, in Autonomous Robots,” 9:247-253, Kluwer, 2000.

[14] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numerische Mathematik, v. 1, p. 269-271, 1959.

[15] S. Kirkpatrick, “Optimization by Simulated Annealing: Quantitative Studies,” Journal of Statistical Physics, 34, pp. 975-986, 1984.

[16] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, “Optimization by Simulated Annealing,” Science, 220, pp. 671-680, 1983.

[17] E. Valbahs and P. Grabusts, “Motion Planning of an Autonomous Robot in Closed Space with Obstacles,” Scientific Journal of RTU. 5. series., Datorzinatne. - 15. vol., pp. 52-57, 2012.

[18] M. P. Vecchi and S. Kirkpatrick, “Global Wiring by Simulated Annealing,” IEEE Transaction on Computer Aided Design, CAD-2, pp. 215-222, 1983.