USING CASE-BASED REASONING FOR MOBILE ROBOT PATH PLANNING

Jaroslav Hodál, Jiří Dvořák*

The mobile robot path planning involves finding the shortest and least difficult path from a start to a goal position in a given environment without collisions with known obstacles.

The main idea of case-based reasoning (CBR) is a presumption that similar tasks probably also have similar solutions. New tasks are solved by adapting old proved solutions of similar tasks to new conditions. Tasks and their solutions (cases) are stored in a case base.

The focal point of this paper is the proposition of a path planning method based on CBR combined with graph algorithms in the environment represented by a rectangular grid. On the basis of the experimental results obtained, it is possible to say that case-based reasoning can significantly save computation costs, particularly in large environments.

Key words : *mobile robot, path planning, graph algorithms, case-based reasoning, case graph*

1. Introduction

One of the typical tasks of current robotics is the navigation of autonomous mobile robots. The aim of autonomous mobile robot navigation is the retrieval and passing of the shortest and least difficult path between two points in a given environment without collisions with obstacles. We can distinguish two principal levels of navigation : global navigation (path planning) and local navigation. The focal point of the global navigation is the absolute positioning in the environment and finding a path between selected locations. Local navigation locates relative position against the surrounding objects and ensures correct interaction with them.

If the navigation is performed in a dynamic or partially unknown environment, this task complicates considerably. Information needs to be obtained on the environment in which a robot locomotes and changes monitored in this environment are subsequently used to adapt the robot behavior accordingly.

Case-based reasoning (CBR) solves a new problem by adapting solutions of similar problems solved in the past to the conditions of this new problem. That is why CBR systems store previously solved problems and their solutions in the form of cases. Storage of previous experience for future reasoning leads to learning from past situations.

Benefits resulting from the use of case-based reasoning will be most significant in applications to robots which locomote in less variable environment and often perform similar repetitive tasks. However, in such conditions, industrial robots work perhaps most often.

* Ing. J. Hodál, RNDr. J. Dvořák, CSc., Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic

2. Robot path planning

Many different algorithms have been proposed for path planning in the past (see e.g. [1], [2]). The principal methods of the path planning include methods based on path maps, probabilistic roadmaps [3], rapidly exploring random trees, decomposition of the environment into cells, potential field [4], genetic algorithms [5] and behavior of insect colonies [6].

The choice of a suitable algorithm mostly depends on the model of environment used. Environment models are usually described by means of topological maps and metric maps (grid or continuous).

Topological maps are usually represented by various graphs (e.g. visibility graphs, tangent graphs, Voronoi diagrams). The graph vertices represent locations in the environment and the edges paths between them. Dijkstra's algorithm and its variations are usually used for the shortest path retrieval.

A grid map is a set of traversable or untraversable cells. Two-dimensional rectangular grids are frequently used when the map is flat, but it is also possible to work with other cell shapes. Cells are mostly constructed to contain only a single type of environment. More complex models add further information to the cell (e.g. type of terrain, usage count, frequency of unknown obstacles). The path is constituted by a sequence of cells while through each of them a partial path is defined.

Environment decomposition methods can be divided into two groups: exact decomposition methods (decomposing only obstacle free space into disjoint simple shaped cells) and approximative decomposition methods (decomposing the whole environment including obstacles, most often by a rectangular grid). The paper [7] deals with various environment representations in detail.

3. Case-based reasoning

The keynote of the case-based reasoning is a presumption that similar tasks will probably have similar solutions. Case-based reasoning solves a new problem by adapting known similar, previously solved problems (cases), which leads to the desirable effect of learning from the previous experience. A survey of the fundamental notions related to the case-based reasoning and descriptions of the main methodological approaches on this field can be found in [8].

The function of a CBR system is described by what is called a common CBR cycle (4R cycle), which contains the following four steps [8] – retrieve, reuse, revise and retain. These steps represent a simplified description of the CBR system function: retrieving the most similar case or cases, reusing information and knowledge from the retrieved case to solve a new problem, revising the proposed solution and retaining parts of this experience to be used for future problem solving. Selecting appropriate strategy for maintenance of the case base can increase the efficiency of the CBR system.

In the path planning, it is possible, through the case base, to reuse the previously found paths or their parts. Instead of searching for a new path, it is sufficient to find the most similar paths or their parts and adapt them to be suitable for actual problem solving. The similarity of path planning problems is often thought of as mutual proximity of the start (goal) points of the new and the stored problem.

A certain shortcoming of some CBR systems is that they do not recognize multiple cases and parts of cases. The use of only whole cases in the path planning considerably reduces the utilization level of the previous experience. This approach, with cases consisting of whole paths, is used, for example, in [9], [10], [11].

A CBR approach for a continuous environment that uses parts of paths was described by Haigh and Shewchuk [12]. They propose a case graph constituted by linear segments of paths with segments intersecting only at their end points.

The CBR mechanisms are often combined with alternative methods in hybrid systems. These methods can be used to solve problems that cannot be solved with the current case base. A combination of CBR with a genetic algorithm is presented, e.g., in [13] and [14], with reinforcement learning in [15], and with probabilistic search algorithm in [9].

4. Path planning method using CBR

In this section, we propose a path planning method combining case-based reasoning with graph searching algorithms.

4.1. Environment representation

We consider an environment in the form of a rectangular grid $[1,m]\times[1,n]$. Each cell c of this grid is determined by a pair of coordinates in the grid $c=(x,y)$ where $x\in\{1,2,\dots,m\}$, $y\in\{1,2,\dots,n\}$. Cells containing known obstacles are initiated by 1. Cells representing free space carry 0, which means no risk. It is possible to consider an initial map containing general risks with values from the interval $(0;1)$. Such an initial map can be prepared if the risk involves a known terrain quality.

During a robot's movement along a previously planned path, each cell can change its risk value. The risk value of a cell increases if, in this cell, the robot meets an obstacle (static or dynamic). Similarly, if the passage through a cell has been successful, the risk value decreases. It is possible to include other influences of the environment in the risk value (type of terrain, obstacle proximity, etc.).

The eligible movements of a robot in the above environment can only be horizontal, vertical, and diagonal. The distance between cells $c_i=(x_i,y_i)$ and $c_j=(x_j,y_j)$ can be defined by these formulas:

$$d(c_i,c_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}\ , \tag{1}$$

$$d(c_i,c_j)=|x_i-x_j|+|y_i-y_j|\ , \tag{2}$$

$$d(c_i,c_j)=\left(\sqrt{2}-1\right)\min\{|x_i-x_j|,|y_i-y_j|\}+\max\{|x_i-x_j|,|y_i-y_j|\}\ . \tag{3}$$

Formula (1) is mostly used to estimate a distance. Formula (2) is suitable for interior environments. Formula (3) is used to evaluate the actual length of the path planned (it corresponds to a situation with only horizontal, vertical, and diagonal directions being eligible). This formula can be separated into two parts: the part without $\sqrt{2}$ corresponds to a horizontal or vertical move and the part with $\sqrt{2}$ corresponds to a diagonal move. Depending on the environment type, it is possible to use any of the above formulas to evaluate the ideal path's length.

4.2. Path representation

A path $P(c_s, c_g)$ from the start cell c_s to the goal cell c_g is specified as a sequence $P(c_s, c_g) = \{c_s, c_{k_1}, \ldots, c_{k_i}, \ldots, c_{k_n}, c_g\}$ where internal cells c_{k_i} of this sequence are only those cells in which the direction changes or where the path intersects another path. A path is thought of as a sequence of linear segments (c_i, c_j). Each linear segment has its cost function F value characterizing its traversability. The cost of a path is the sum of the costs of the segments contained in this path.

The cost function can be constructed in several ways. An intuitive approach to this task leads to the formula

$$F(c_i, c_j) = l(c_i, c_j)\, r(c_i, c_j) \ , \tag{4}$$

where l is the length of linear segment between points c_i and c_j, r is the risk of passing through this segment. The retrieval of the best paths is based on this cost function. Evaluation of the risk r can be performed in several ways according to the application type. We consider a sum of risks of cells constituting the segment

$$r(c_i, c_j) = \frac{1}{2}\, r(c_i) + \sum_{c \in I(c_i, c_j)} r(c) + \frac{1}{2}\, r(c_j) \ , \tag{5}$$

where $I(c_i, c_j)$ is the set of all internal cells of the edge (c_i, c_j). If it is necessary to avoid high risks in individual cells, it will be more suitable to use the maximum risk of all cells in the segment.

The cost function can be also constructed in other ways, e.g., as a weighted sum of the segment length and the risk of passing through the segment. Whether the choice prefers short or reliable paths depends on the cost function used.

4.3. Case base and its maintenance

All paths traversed are stored in the case base. In order to increase the utilization of the information stored, parts of paths, rather than whole paths, are used as cases. A part of a path is defined as any linear segment of a path delimited by two points (each of these points can be the end point of path or a point of turn or intersection of the path with another path). A new case is stored only when the case base does not contain a similar case. Each case in the case base is weighted by the cost function (4).

All path segments stored in the case base constitute the case graph. Thanks to the representation of the case base in the form of a graph the difficult retrieval of the most similar case can be replaced by a shortest path search in the graph using known graph algorithms.

It is necessary to keep the number of edges and vertices of the case graph reasonably limited. A simpler form of the case graph's maintenance is to delete those redundant vertices that seem to lie in the middle of an edge. Another method used to maintain the case graph consists in erasing edges (whole segments of paths). Some strategies of such a maintenance with whole paths as cases are described by [10].

With regard to the character of the application proposed, forgetting the worst (most risky) cases seems to be the most optimal strategy. This strategy also corresponds well with other principles of the case-based reasoning.

4.4. Path planning without CBR

In the event that it is impossible to use case-based reasoning for path planning, it is necessary to equip the system with methods working on other principles. Such methods are used mostly in situations in which the case base does not contain any path whose end points are in the proximity of the given start and goal points. It is also possible to use them when the path proposed by the case-based reasoning is not good enough.

For planning without CBR, it is possible to use Dijkstra's and related algorithms (such as A*). The grid is considered as a graph with vertices being cells and edges connecting each cell with the adjacent cells not occupied by obstacles. Results of these planning algorithms are the sequences of subsequent map cells that constitute the proposed paths.

4.5. CBR path planning

When we use the case graph, the adaptation phase precedes searching for the most similar case. We temporarily extend the case graph with segments connecting the start (goal) point with all proposed near points of the case graph. Then, only one pass through the search algorithm will be sufficient for the path retrieval. The adaptation phase uses some path planning method without CBR (Dijkstra's or A* algorithm). Fig. 1 shows an example of the case graph and the path proposed.

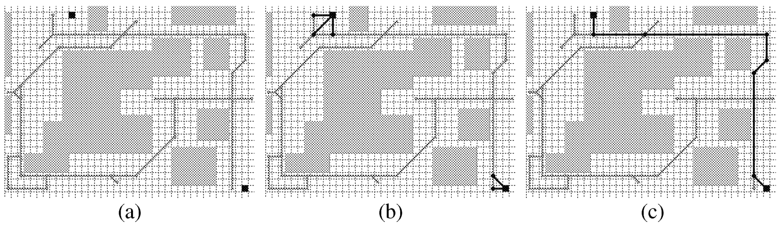

Fig.1: Case graph with start and goal (a), adaptation (b), path proposed (c)

In CBR path planning, it may not be always possible to find the path (it is impossible to join the start or goal with the case graph, the case base contains little information, etc.) or the path found can be of a poor quality. In these situations, an additional planning method is automatically used that does not use CBR.

A CBR path planning algorithm can be described by the following steps:

1. The given start and goal are added to the case graph as vertices.
2. If the start or goal is located on an existing edge and is not identical with any existing vertex of the case graph, the edge is divided into two separated segments.
3. If the start or goal is not located within the case graph, the nearest points of the case graph are found.
4. The start and goal are connected by temporary edges with the nearest points of the case graph found. These edges extend the actual case graph (the adaptation phase of CBR).
5. If it is not possible to create any of these additional edges, the path planning by CBR fails, an alternative path planning method is applied and the algorithm ends.
6. The best path in the extended case graph is searched for by using Dijkstra's or A* algorithm.

7. If this path is found, its quality (the ratio of its length and the ideal path length) is examined. A poor-quality path of is replanned by an alternative method and the algorithm continues by Step 9.
8. If no path is found, an alternative method of path planning is applied.
9. All temporary edges are removed from the case graph and the algorithm ends.

The proposed solution is tested by a robot using the path. If the robot encounters unknown obstacles when moving along the path, it stores the information obtained in a private map. Repetitive occurrence of obstacles at the same location leads the valuation of this location being worsened for further use. A successfully traversed path is retained in the case base.

5. Path planning for nonholonomic robots

The CBR path planning method described can be also applied to nonholonomic robots with kinematic constraints, but the graph searching algorithms used have to be modified. As a consequence of kinematic constrains, the robot cannot continue from a vertex via an arbitrary edge. The choice of the following edge depends on which edge entering to the given vertex has been used.

For simplicity, we will denote the vertices of the graph (grid cells) only by indices i rather than symbols c_i. Let $G = (V, E)$ be a weighted directed graph where V is the set of vertices and E is the set of directed edges (arcs), $E \subseteq \{(i,j) | i,j \in V\}$. Let us note that the case graph described in the previous section is undirected, but can be transformed into a directed one by replacing each undirected edge $\{i,j\}$ by the pair (i,j) and (j,i) of directed edges.

Let us introduce the following symbols:

$\delta(i,j)$	the direction of edge (i,j); this is a number from the set $\{0,1,\dots,7\}$; we have $\delta(i,j) = (\delta(j,i)+4) \bmod 8$;
$F(i,j)$	the weight of edge (i,j); this weight is determined by formula (4);
$d_s(i,j)$	the distance from vertex s to vertex j through edge (i,j); the distance $d_s(i,j)$ means the minimum weight of all paths from s to j through (i,j); if there exists no path from vertex s to vertex j through edge (i,j), we set $d_s(i,j) = \infty$;
$p_s(i,j)$	the initial vertex of the edge immediately preceding edge (i,j) on the shortest path from vertex s to vertex j through edge (i,j); the value $p_s(i,j) = -1$ means that the path from vertex s to vertex j through edge (i,j) does not exist;
$P(i,j,k)$	the condition for a robot movement from edge (i,j) to edge (j,k);
s'	the fictitious predecessor of the start vertex s; we set $\delta(s',s) = w_s$ where w_s is the initial direction of the robot at vertex s;
$E(i)$	the set of edges that are incident with vertex i;
E_D	the set of edges whose values $d_s(i,j)$ are definite.

The determination of the optimal path from the start vertex s to the goal vertex g can be described by the following steps (we assume that the initial direction of the robot at vertex s is given):

1. Set $d_s(s',s) = 0$ and $d_s(i,j) = \infty$, $p_s(i,j) = -1$ for each edge (i,j). Set $E_D = \emptyset$, $(k,r) = (s',s)$.
2. Explore all edges $(r,i) \in E - E_D$ satisfying a condition $P(k,r,i)$. If $d_s(r,i) > d_s(k,r)+$ $+F(r,i)$, then set $d_s(r,i) = d_s(k,r) + F(r,i)$, $p_s(r,i) = k$.

3. Find such an edge $(p,q) \notin E_D$ that $d_s(p,q) = \min\{d_s(i,j) |\, (i,j) \notin E_D\}$. If $d_s(p,q) = \infty$, then stop (no other edge is reachable from the vertex s and no path from s to g exists). In the opposite case, add edge (p,q) to E_D and set $(k,r) = (p,q)$.
4. If $E(g) \subset E_D$, then go to Step 5 (the distances to the goal through all its neighbors have been found). In the opposite case, return to Step 2.
5. Set $S = \{\}$. Find such an edge (j,g) that $d_s(j,g) = \min\{d_s(i,g) |\, (i,g) \in E_D\}$ and set $k = g$.
6. Put edge (j,k) at the beginning of sequence S. If $j = s$, then stop (the sequence S represents the optimal path).
7. Set $i = p_s(j,k)$, $k = j$, $j = i$ and continue by Step 6.

The modified A* algorithm differs from the previous one in Step 3 where an edge $(p,q) \notin E_D$ is determined in such a way that

$$d_s(p,q) + d(q,g) = \min\{d_s(i,j) + d(j,g) |\, (i,j) \notin E_D\} ,$$

where $d(j,g)$ is defined by formula (1) or (2).

The condition $P(i,j,k)$ for a robot movement from edge (i,j) to edge (j,k) depends on the nature of the kinematic constraints. If a robot can move only forward or forward-diagonal, then we define this condition as follows:

$$\min\{\alpha(i,j,k), 8 - \alpha(i,j,k)\} \leq 1 \qquad \text{or} \qquad \alpha(i,j,k) \in \{0,1,7\} ,$$

where $\alpha(i,j,k) = |\delta(i,j) - \delta(j,k)|$. If a robot can move also backward and backward-diagonal, then the condition $P(i,j,k)$ can be defined in this way:

$$\alpha(i,j,k) \in \{0,1,7\} \ \vee \ \exists (j,l) : \alpha(i,j,l), \alpha(l,j,k) \in \{0,1,7\} \ \vee$$
$$\vee \ \exists (j,l),(j,p) : \alpha(i,j,l), \alpha(l,j,p), \alpha(p,j,k) \in \{0,1,7\} .$$

In the case of backward or backward-diagonal movement, it is necessary to add to the expression $d_s(k,r) + F(r,i)$ in Step 2 the weight of that part of edge (j,l), and eventually also edge (j,p), which the robot has to use before moving to edge (j,k).

6. Computational experiments

All experiments have been performed on a personal computer with Mobile Intel Pentium 4-M, 1.70 GHz processor and 768 MB RAM. Most experiments depend on the number of vertices of the case graph and the optional values of some other parameters. Since experimental path planning is a multi-parametric task, certain parameters were selected in a way that appeared the most suitable for conducting further experiments.

6.1. CBR planning time and a number of case graph vertices

In this test, start and goal cells were set in a way that enables path planning without using alternative methods. This test was performed for two adaptation methods used – full and selective. The full adaptation checks each case whether it intersects the adaptation neighborhood. The selective adaptation uses only cases with at least one end point located in the triple extended adaptation neighborhood. Results are presented in Fig. 2.

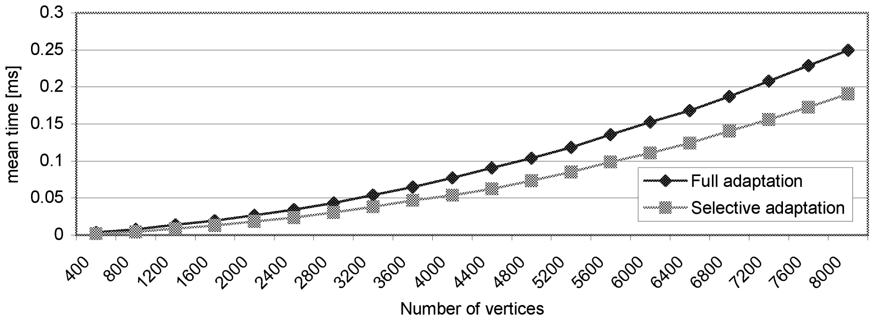

Fig.2: CBR planning times

6.2. Frequency of CBR planning method choice

The frequency of the case-based reasoning method usage depends on the coverage of the environment by the case graph. Case graphs with gradually increasing numbers of vertices on the 200×200 environment map were constructed in this test. Each of them was used for planning 10 000 paths with random start and goal points. The result is the number of paths planned by the case-based reasoning. Failures partially consisted of those paths in which no connection could be found to the case graph for given adaptation parameters, as well as those dismissed for their poor quality on the basis of the ratio of their length to the length of the ideal path.

Results of this test imply (see Fig. 3) that, already with approximately 6 percent of cells being occupied by vertices of the case graph, it is possible to plan about 50 percent of the paths by case-based reasoning. With 10 percent coverage, there will be as much as 75 percent of successful plans and, with 15 percent coverage, 90 percent of the paths may be planned by case-based reasoning.

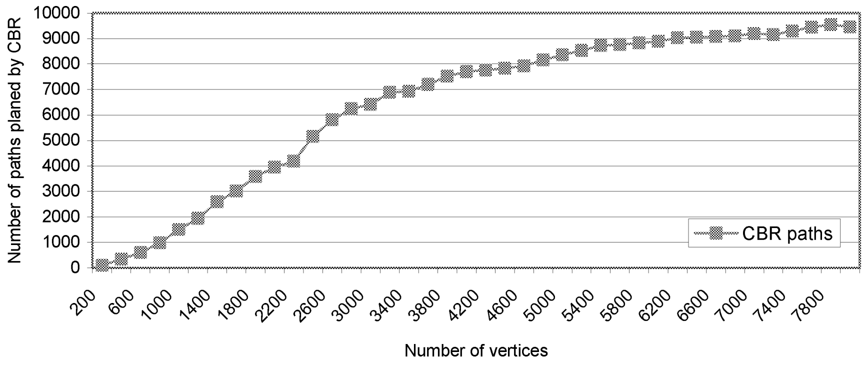

Fig.3: Frequency of CBR method choice

6.3. Planning times of tested path planning methods

We compared the CBR algorithm with Dijkstra's, A* and Ai* algorithms. The Ai* algorithm is a modification of A* algorithm which, as a heuristic, uses an integer approximation of the distance from a distance map constructed using 8-direction propagation method. The usage of integer approximation causes a maximum error of distance estimation of about 8 percent.

The comparison of times necessary to plan the path was performed stepwise on square maps of 50×50 to 600×600 cells. In several series, the start and goal points were set, for a path planned by means of case-based reasoning to be placed between them. The results of CBR planning are shown for case graphs with 1000, 2000 and 3000 vertices (Fig. 4).

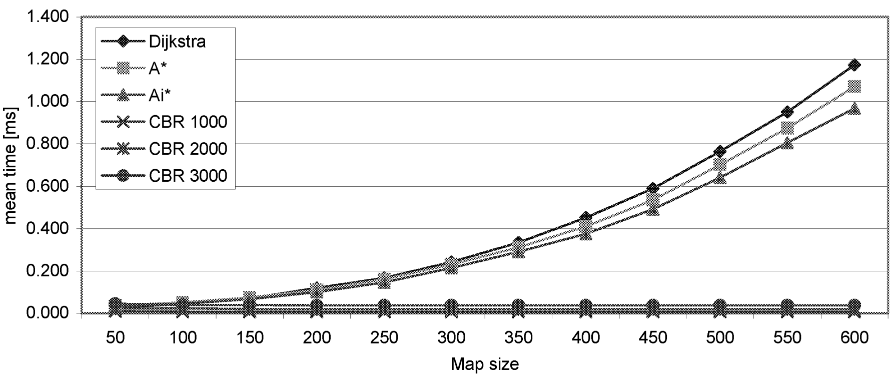

Fig.4: Planning times

We can see that the time spent by alternative planning methods is exponentially increasing. By contrast, it is obvious that the speed of planning by CBR is influenced by the map size only insignificantly. Above all, it depends on the number of vertices and edges of the case graph. It is useful to fix a certain limit on the vertex count for a particular map size. In principle, this limit guarantees an almost constant planning time.

6.4. Parameters of paths proposed by the tested methods

Path parameters were also compared to show their dependence on the coverage of the environment by the vertices of the case graph (a 200×200 map was used). 1000 tasks were randomly generated for each case graph. Paths were planned between each pair of points generated using all four planning methods. The method with the lowest value for every parameter monitored was selected in each series with the deviation percentages for other methods subsequently calculated. Average deviations for all 1000 paths form the results of particular tests.

The most important parameter of paths monitored was the path cost. At 5 percent coverage of the environment by the case graph vertices, the deviation of costs is about 16 percent of the ideal costs obtained by Dijkstra's algorithm (see Fig. 5 and Tab. 1). At 10 percent coverage of the map, the deviation drops below 8 percent and, at 15 percent coverage, even below 5 percent of the best costs obtained.

	Number of vertices											
	2000			4000			6000			8000		
Path	cost	length	turns	cost	length	turns	cost	length	turns	cost	length	turns
CBR	16.01	18.09	28.89	7.60	9.59	21.51	4.46	6.47	20.97	2.96	4.53	16.69
Dijkstra	0.00	1.32	13.92	0.00	1.53	14.46	0.00	1.63	15.25	0.00	1.32	14.87
A*	5.43	0.30	7.53	5.45	0.38	7.30	5.67	0.27	5.87	5.57	0.40	7.66
Ai*	11.10	2.24	13.01	11.37	2.51	13.42	11.22	2.28	15.92	11.06	2.19	12.91

Tab.1: Relative deviations from the best values of path costs, path lengths and number of turns (%)

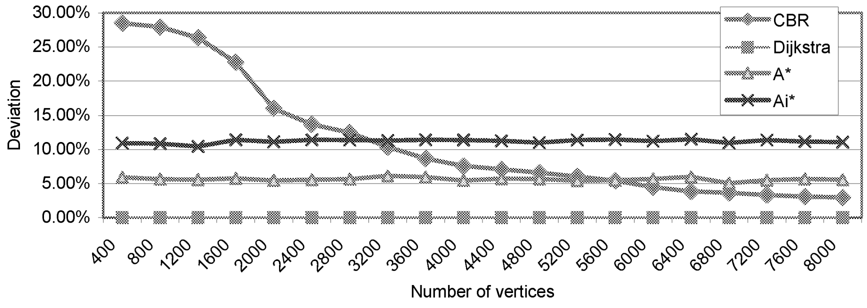

Fig.5: Relative deviations from the best costs

In addition to path costs, the path length and number of turns were monitored, too (see Tab. 1). In this test, the deviation of the lengths of the paths planned by CBR drops below 10 percent at 10 percent map coverage. This logically corresponds to the results of the previous criterion. The number of turns depends significantly on the alternative method of path planning used. If the CBR method uses Dijkstra's algorithm, then it can achieve almost the same results as this algorithm from 20 percent coverage of the environment. Better results using a similar case graph could be obtained with A* as alternative method. In that case, a simultaneous decrease of path length is achieved as well as an increase of costs (similarly to A* algorithm itself).

7. Conclusion

This paper investigates the applicability of case-based reasoning for mobile robot path planning. Paths are planned in an environment represented by a rectangular grid. When this representation is used, work with cases is faster and simpler at the expense of a small downgrade of the planned path quality (about 4 percent). General principles described in this paper are also applicable to continuous environments.

We proposed and implemented a method of path planning by means of case-based reasoning. This method uses a special representation of a case base in the form of a case graph. This method combines case-based reasoning with alternative path planning methods (Dijkstra's algorithm, A* algorithm and its integer modification Ai*).

The method described was implemented for path planning of holonomic robots. The proposed approach is also applicable to nonholonomic robots with kinematic constraints. For this case, a modification of graph algorithms to be used was proposed.

In the implemented environment, experiments for testing the efficiency of case-based reasoning methods were performed. Results of these experiments suggest that, with a reasonable size of the case graph, the use of case-based reasoning brings considerable computing time saving. Most of the precision, quality and planning-time requirements are already fulfilled at a 5–15 percent uniform coverage of the environment by the case graph vertices.

The method proposed uses graph algorithms as alternative path planning methods to case-based reasoning. In general, any path planning method with explicit path representation can be combined with case-based reasoning.

Because of the repetitive performing of similar tasks, the most frequent environments enabling the ideal use of described methods seem to be industrial and office plants. In such applications, we can expect the greatest practical use of these methods.

Acknowledgement

This paper was supported by the project MSM 0021630518 'Simulation Modelling of Mechatronic Systems'.

References

[1] Latombe J.C.: Robot Motion Planning, Kluwer Academic Publishers 1991, Norwell MA
[2] LaValle S.M.: Planning Algorithms, Cambridge University Press 2006, Cambridge
[3] Geraerts R., Overmars M.H.: Sampling and Node Adding in Probabilistic Roadmap Planners, Robotics and Autonomous Systems 54 (2006) 165–173
[4] Agirrebeitia J., Avilés R., De Bustos I.F., Ajuria G.: A New APF Strategy for Path Planning in Environments with Obstacles, Mechanism and Machine Theory 40 (2005) 645–658
[5] Burchardt H., Salomon R.: Implementation of Path Planning Using Genetic Algorithms on Mobile Robots, In: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2006), Congress on Evolutionary Computation (CEC 2006). Vancouver, Canada 2006, pp. 1831–1836
[6] Kose M., Acan A.: Knowledge Incorporation into ACO-Based Autonomous Mobile Robot Navigation, In: Computer and Information Sciences – ISCIS 2004, Lecture Notes in Computer Science 3280, eds. C. Aykanat et al., Springer-Verlag 2004, pp. 41–50
[7] Meyer J.-A., Filliat D.: Map-Based Navigation in Mobile Robots: II. A Review of Map-Learning and Path-Planning Strategies, Cognitive Systems Research 4 (2003) 283–317
[8] Aamodt A., Plaza E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, AI Communications 7 (1994) 39–59
[9] Kruusmaa M., Svensson B.: Combined Map-Based and Case-Based Path Planning for Mobile Robot Navigation, In: Proceedings of International Symposium of Intelligent Robotic Systems, ed. Vidyasagar M., Tata McGraw Hill 1998, pp. 11–16
[10] Kruusmaa M., Svensson B.: Using Case-Based Reasoning for Mobile Robot Navigation, In: Proceedings of the 6th German Workshop on Case-Based Reasoning, Berlin 1998, 8 pp.
[11] Kruusmaa M., Willemson J.: Covering the Path Space: a Casebase Analysis for Mobile Robot Path Planning, Knowledge-Based Systems, 16 (2003) 235–242
[12] Haigh K.Z., Shewchuk J.R.: Geometric Similarity Metrics for Case-Based Reasoning, In: Case-Based Reasoning: Working Notes from the AAAI-94 Workshop, Seattle, WA, AAAI Press 1994, pp. 182–187
[13] Louis S.J., Li G.: Combining Robot Control Strategies Using Genetic Algorithms with Memory, In: Proceedings of the 6th International Conference Evolutionary Programming, Indianapolis, Springer-Verlag 1997, pp. 431–441
[14] Dvořák J., Krček P., Samohýl P.: Using Case-Based Reasoning and Genetic Algorithms for Mobile Robot Path Planning, Elektronika (Poland), no. 8–9, 2004, pp. 55–57
[15] Ram A., Santamaría J. C.: Continuous Case-Based Reasoning, Artificial Intelligence, 90 (1997) 25–77

Received in editor's office: November 26, 2007
Approved for publishing: July 7, 2008