Domain-Dependent View of Multiple Robots Path Planning

Pavel Surynek
Charles University
Faculty of Mathematics and Physics
Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic
surynek@ktiml.mff.cuni.cz

Abstract. We study a problem of path planning for a group of robots in this paper. The problem is stated as a finding of spatial-temporal paths through which the robots can go from their initial locations to the given goal locations in a certain environment. The robots must subordinate to a variety of physical constraints. The most important constraints from our point of view are that the robots must avoid obstacles and they must avoid each other. Both of these two constraints can be captured by a model where the environment is modeled as an undirected graph. Vertices of this graph represent possible locations for the robots and edges represent possible transitions between locations. The robots are located in the vertices of the graph. Each vertex can be occupied by at most one robot. A robot can move to the neighboring unoccupied vertex. The main result of the paper is the description of a class of the problem for which a polynomial time solving algorithm exists. We also present an experimental evaluation in which we tested the ability of several state-of-the-art planners to solve the problem.

Keywords. path planning, multiple robots, tractable class, graphs

1. Introduction

We deal with the problem of *path planning for a group of robots* that are moving in a certain environment. The two aspects of the situation we are concentrating on are that the robots can collide with each other and they must avoid obstacles in the environment. Both of these aspects can be captured in an abstraction of the situation where the environment is represented as a graph. Vertices of this graph represent locations in the environment and edges represent possible transition between locations. The robots are located in the vertices of the graph. A robot can move from one vertex to another neighboring vertex if the destination vertex is unoccupied. There is a constraint that at most one robot can be located in each vertex. This constraint is used to model collisions between robots. More precisely, we forbid collisions between robots in this model.

This abstraction of the problem was introduced by Ryan [11,12]. The author proposes an efficient technique for reducing the search space based on the decomposition of the graph representing the environment into special sub-graphs. However, the author does not provide any insight into the tractability or intractability of the problem. The exhaustive search is given as the only option for solving the problem. We chose a different approach to the problem. We are trying to exploit the structural properties of the problem for getting insight into the complexity of the problem. We found that such an insight can be found used for developing a solving algorithm.

In this paper we present a quite broad tractable class of the problem of path planning for multiple robots. More precisely, we describe a class of problems that are solvable in polynomial time. The class is quite broad since the problems not belonging into this class seem to be too extreme with respect to the real world and therefore we conceive that the defined class may be useful.

Second, we would like to present some experimental results made with several state-of-the-art *domain-independent planners* [1] on the multiple robot path planning problems (a domain-independent planner is software that solves planning problems described by some kind of a formalism for describing planning problems, the most widely used such a formalism is the *Planning Domain Definition Language - PDDL* [9]). Our question was whether it is really necessary to apply a domain-dependent view of the problem. If for instance domain-independent planners would be efficient on the problem the domain-dependent view of the problem would be less interesting. However, it seems that this is not the case.

We found that the problem of multiple robots path planning problems is difficult for today's state-of-the-art domain dependent planners. Moreover, the results show that today's planners are not able to discover tractability of the problem and even the easy problems are solved by search which consumes too much time. Hence these experimental results justify that the defined tractable class is worthwhile and it justifies domain-dependent approach to the problem from the more global perspective.

The paper is organized as follows. We formally introduce the problem of path planning for multiple robots first (part 2). This part recalls the definitions from the relevant literature. The major part of the paper is dedicated to the definition of the tractable class of the problem (part 3). In the final part (part 4) we evaluate domain-independent planners on several instances of multiple robots path planning problem.

2. Multiple Robot Path Planning

Briefly said, we need to find *spatial-temporal paths* for multiple robots (that is a feasible assignment of locations to the Cartesian product of the set of robots and time) in a certain environment (for example three-dimensional world with obstacles). Each robot starts at a certain location in the environment and its goal is to reach another location in the environment. We assume that the robots can move within the environment while they must respect common physical constraints. From our point of view the two most important physical constraints that must be captured in our reasoning are that at most one robot can occupy a certain location in the environment at a time (the robots can collide with each other and we want to avoid collisions) and the robots must avoid obstacles in the environment.

2.1. Multiple Robot Path Planning

The formalization of the problem we introduce here was originally used by Ryan [11, 12]. It is abstracted from the geometric features of the environment while the topological features are emphasized. A natural assumption is that all the robots are the same (they are of the same size and they have the same abilities). The main abstraction is that a map of the environment is given in the form of an undirected graph and this graph is shared by all the robots. Vertices of the graph represent locations in the environment. The robots occupy vertices of the graph. A collision between a pair of robots occurs if and only if they are trying to simultaneously occupy the same vertex of the graph. That is, the locations represented by vertices must be physically spaced far apart enough so that two robots can occupy any pair of distinct vertices without colliding. A robot can move from one vertex to another vertex if both vertices are connected by an edge and the target vertex is unoccupied and no other robot is simultaneously entering (or leaving) this target vertex. This graphical representation provides enough flexibility to reflect broad scale of real world situations (for example obstacles between locations can be represented by absence of edges between vertices).

Having the group of robots placed in a set of vertices the task is to move these robots according to defined rules into a given destination locations. That is for each robot we are given its initial vertex and its goal vertex.

The problem is formally stated as follows. We are given an undirected graph $G=(V,E)$ where $V=\{v_1,v_2,\ldots,v_n\}$. We also have a set of robots $R=\{r_1,r_2,\ldots,r_\mu\}$ where $\mu<n$, and two simple functions $S_0:R\rightarrow V$ and $S^+:R\rightarrow V$, such that $S_0(v_i)\neq S_0(v_j)$ and $S^+(v_i)\neq S^+(v_j)$ for $i\neq j$ representing initial and goal locations of the robots respectively. The task is to find a number m a path $P_r=[p_1^r,p_2^r,\ldots,p_m^r]$ for every robot $r\in R$ where $p_i^r\in V$ for $i=1,2,\ldots,m$ and either $\{p_i^r,p_{i+1}^r\}\in E$ or $p_i^r=p_{i+1}^r$ for $i=1,2,\ldots,m-1$ (the path may contain loops and a robot may stay in a vertex for more than one time step). The path for a robot $r\in R$ starts in its initial location and must terminate in the goal location, that is $p_1^r=S_0(r)$ and $p_m^r=S^+(r)$. Moreover, paths for every pair of distinct robots $r_i\in R$ and $r_j\in R$ must be non-colliding. That is $p_l^{r_i}\neq p_l^{r_j}$ must hold for $l=1,2,\ldots,m$ (in every time point we require the robots to be in different vertices).

We took the above formulation of the problem from [11] except the part describing feasible paths for robots. We simplified the formulation of feasible paths in order to analyze complexity of the problem more easily. Let us note that the problem is equivalent to the original one from [11].

Let us call the problem defined above a *multiple robot path planning problem* (briefly *MRPP problem*). It is natural to ask for the shortest solution of the MRPP problem. In such a case we require the length of a longest path for a robot to be shortest as possible. Formally we require the number m in the definition of the solution to be smallest as possible. Let as call this variant of the problem *multiple robot optimal path planning problem* (*MROPP problem* in short).

The MRPP problem and its optimal version can be formulated as AI planning problems [4]. For space limitations we cannot give any more details (see problems used for experiments on the web).

3. Tractable Class of MRPP

Our algorithm is based on the analysis of a special variant of connectivity of the graph $G=(V,E)$ representing the problem.

First let us develop the solving process for a special case of the MRPP problem where we have a *2-connected* graph $G=(V,E)$ with the set of vertices $V=\{v_1,v_2,\ldots,v_n\}$. Next we have the set of $n-2$ robots $R=\{r_1,r_2,\ldots,r_{n-2}\}$. That is, only two vertices of the graph G remain unoccupied. This special case of the MRPP problem can be solved in polynomial time for arbitrary feasible initial and goal locations of the robots S_0 and S^+.

In the following paragraphs we formally describe the algorithm. We start with basic definitions from the graph theory [13].

Definition 1 (connectivity). An undirected graph $G = (V, E)$ is *connected* if $|V| \geq 2$ and for every pair of vertices $u \in V$ and $v \in V$, $u \neq v$ there is a path connecting u and v consisting of edges from E. □

Definition 2 (2-connectivity). An undirected graph $G = (V, E)$ is *2-connected* if $|V| \geq 3$ and the graph $H = (V - \{v\}, E)$ is connected for arbitrary vertex $v \in V$. □

The following well known observation is crucial for developing a solving algorithm.

Observation 1. Each 2-connected graph can be constructed from a cycle by adding paths. More precisely suppose that the construction is in the *i*th step and we have a graph $G_i = (V_i, E_i)$. Adding a path means selecting a pair of vertices in the graph $u \in V_i$ and $v \in V_i$, $u \neq v$ and connecting them by a path of a given length l. That is, we obtain a new graph $G_{i+1} = (V_{i+1}, E_{i+1})$ where $V_{i+1} = V_i \cup \{w_1, w_2, \ldots, w_l\}$ and $E_{i+1} = E_i \cup \{\{u, w_1\}, \{w_1, w_2\}, \ldots, \{w_l, v\}\}$. ■

Suppose that we know the sequence of paths for construction of the given graph $G = (V, E)$ defining the problem. This information can be obtained in polynomial time by a variant of the depth-first search algorithm executed over the graph G. Let us call this sequence of path a *path decomposition* of 2-connected graph.

Definition 3 (almost placed robots). Let us have a path $\{\{u, w_1\}, \{w_1, w_2\}, \ldots, \{w_l, v\}\}$ from the path decomposition of G and the set of robots $\rho = \{r \in R \mid S^+(r) \in \{w_1, w_2, \ldots, w_l\}\}$. We call the set of robots of the path ρ *almost placed* if their goal locations can be reached by a sequence of moves while these robots do not leave the set of vertices $\{w_1, w_2, \ldots, w_l\}$. □

By this definition we want to capture the situation within a path where only one vertex that must be finally unoccupied remains to be freed. Notice also, that other robots than that in ρ are allowed to temporarily enter the path.

3.1. Algorithm for 2-connected Graph

The solving algorithm proceeds by taking paths from path decomposition of the graph one by one and places the robots whose goal locations are within this path to satisfy the definition of almost placed robots.

Observation 2. In the 2-connected graph $G = (V, E)$ with at least single unoccupied vertex (in our case we have two unoccupied vertices) it is possible to transport any robot from arbitrary initial vertex to arbitrary goal vertex by a sequence of allowed moves. ■

Sketch of proof. First observe that it is possible to make unoccupied arbitrary vertex in the given graph. For performing this it is enough to have a path between an unoccupied vertex and a vertex that we want to make unoccupied (see figure 1). Since we have the 2-connected graph such a path always exists.

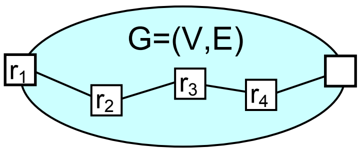

Figure 1: *Freeing an occupied vertex in a graph with a single unoccupied vertex.* It is sufficient to have a path between an unoccupied vertex and a vertex we want to make unoccupied (bold vertex occupied by robot r_1). Then we can perform a plan: move r_4 to unoccupied vertex, move r_3 to unoccupied vertex, move r_2 to unoccupied vertex, and move r_1 to unoccupied vertex that results in freeing the bold vertex.

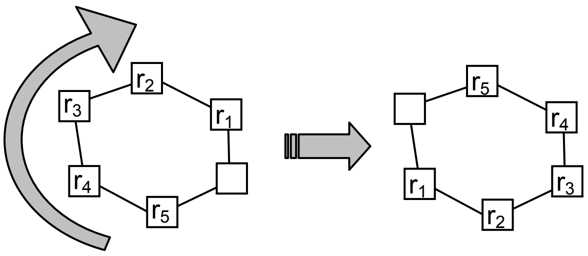

Figure 2: *Rotation of robots along a cycle.* It possible to rotate a group of robots within a cycle with a single unoccupied vertex by a sequence of allowed moves. The ordering of robots along the cycle is preserved.

Observe that if we have a cycle in the graph containing an unoccupied vertex it is possible to rotate robots along this cycle. Formally, the rotation of group of robots within a cycle $C=[c_0,c_1,c_2,\ldots,c_\gamma]$ is a transition from the location of robots S to S' where the following implications hold. If $S(r_i)=c_k$ and $S(r_j)=c_{(k+1)\bmod\gamma}$ then there is l such that $S'(r_i)=c_l$ and $S'(r_j)=c_{(l+1)\bmod\gamma}$ (neighboring robots within the cycle remain neighboring).

Taking these two basic operations together we can rotate robots within arbitrary cycle of the given graph. First we make some vertex within the target cycle unoccupied. Then we perform the rotation (see figure 2).

Now have all the ingredients to perform transition of a robot from its original location to arbitrary goal location. Recall that we have 2-connected graph that is it can be constructed from a cycle by adding paths according to observation 1. Informally we can imagine the graph as many cycles. There is a path from the original vertex where the robot is placed to the goal vertex where we want to place the robot. This path goes through several cycles that can be identified from the given path decomposition. More precisely, for each vertex of the path there is a cycle that this vertex is part of. Moreover, every edge of the path belongs to some cycle. These facts allow us to rotate cycles along the path in a way that the given robot moves from the original vertex to the goal vertex (see figure 3). ■

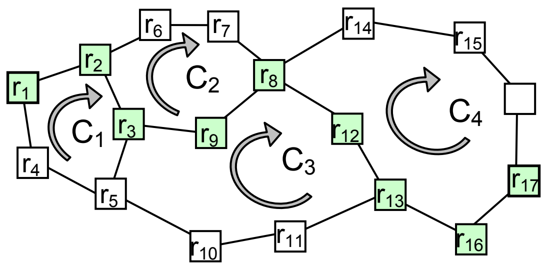

Figure 3: *Transition of robot from one vertex to another vertex in 2-connected graph*. The graph can be constructed from cycle C_1 by incremental adding of path/cycles C_2, C_3, and C_4. The task is to move robot r_1 to the vertex of robot r_{17}. This can be done by rotating cycles along the path connecting the initial and the goal vertex. More precisely it is done by rotating C_1 (r_1 moves to the location of r_3), then by rotating C_2 (r_1 moves to the location of r_8), and finally by rotating C_4 (r_1 moves to the location of r_{17}).

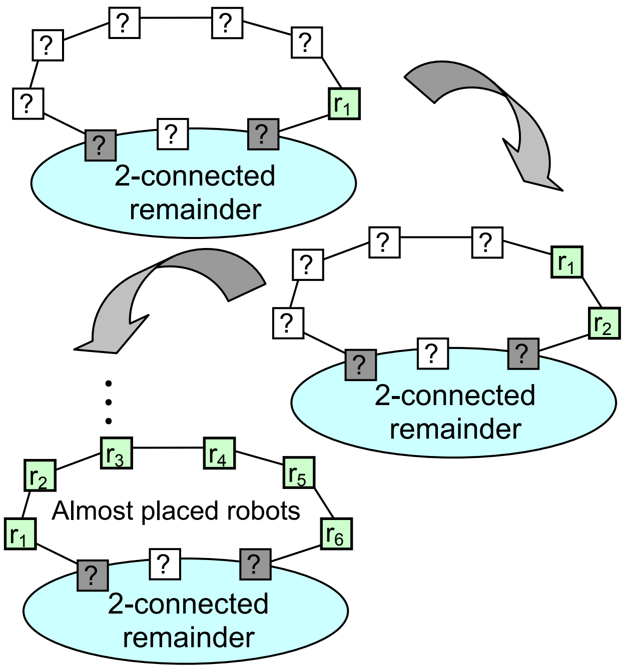

Figure 4: *Placing robots within a path of the path decomposition.* The robots are placed one by one in the right order into the path of the decomposition. The next robot is first rotated to the right starting point of the path (depicted by gray). This is also possible if the robot is already within the path; it can be rotated out without damaging positioning of already placed robots. Then the robot is rotated into the path.

At this point we know how to move a single robot to an arbitrary vertex in the given graph. However the task we want to perform is more complicated. We need to place robots within paths of the path decomposition of the graph to satisfy the condition of almost placed robots. As we stated above we will proceed from one path of the decomposition to another.

We proceed by induction. We place robots into paths of the decomposition starting with the last path and continuing to the original cycle. The paths that have all its robots almost placed are not considered any more in the process. That is, the locations of almost placed robots of already processed paths are not changed.

Inductively we have a path of the path decomposition and the remainder of the graph ($G -$ already processed paths). We place robots into the path as it is described in figure 4. We again proceed by induction. Suppose that several robots are already almost placed within the path. Moreover we can suppose that almost placed robots are placed at the beginning of the path as it is shown in figure 4. The next robot is moved by rotations to the beginning of the path. Notice that if this robot is within the path it can be transported out of the path without affecting the condition of almost placement of already placed robots and then it can be moved to the beginning of the path as well. Having the robot at the beginning of the path it can be placed to satisfy definition of almost placement by the rotation of the cycle formed by the path and part of the graph remainder.

This process is repeated until the original cycle of the path decomposition remains. Until this moment we contrived with a single unoccupied vertex in the graph. Now we need two unoccupied vertices to almost place robots in this last cycle. The following observation provides a recipe how to proceed at this point.

Observation 3. Let us have a cycle $C = [c_0, c_1, c_2, \dots, c_\gamma]$ and a vertex x connected to some vertex of the cycle C. In such a graph it is possible to move robots from arbitrary initial locations to arbitrary goal locations (see figure 5). ■

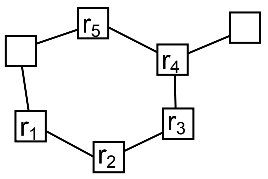

Figure 5: *Illustration to observation 3.* Cycle and a vertex connected to the cycle. Two vertices in the graph are unoccupied. It is possible to reach arbitrary goal locations by the robots in this graph.

Sketch of proof. In fact we need to order robots within the cycle so that they can be placed into the right vertices by rotation only. This can be done easily by moving a robot that we need to order on a different location in the sequence of robots within the cycle to the vertex x. Then we rotate the remaining robots along the cycle so that the right location in the sequence of robots is accessible from the vertex x. In this situation we move the robot from x into the right location in the sequence of robots. By repeating this process we can obtain the required ordering of robots within the cycle. ■

The above observation can be directly applied on the situation with the last cycle of the decomposition. The only thing what is needed is to make unoccupied two vertices in the described sub-graph.

After performing the described steps all the robots are almost placed. It is sufficient now to free vertices that should be finally unoccupied.

3.2. Algorithm for Arbitrary Graph

At this point we have a solving algorithm for the MRPP problem for a 2-connected graph with at least 2 unoccupied vertices. However, not all the graphs are 2-connected. In this section we generalize the algorithm for all the possible graphs.

We use a well known fact from graph theory that a graph can be partitioned to 2-connected components that form a forest of trees [13]. Formally, it is as follows. Let $G = (V, E)$ be a graph, the set of vertices V can be partitioned to 2-connected components $V = B_1 \cup B_2 \cup \ldots \cup B_b$ where B_i is maximum 2-connected component (there is no 2-connected component subsuming B_i) or a single vertex for every $i = 1,2,\ldots,b$. Observe that $B_i \cap B_j = \emptyset$. Let us define a graph $H = (\{B_1, B_2, \ldots, B_b\}, F)$, where $F = \{\{B_i, B_j\} \mid (\exists u \in B_i, \exists v \in B_j)\{u, v\} \in E\}$. The H graph is a forest of trees. We exploit this fact in the algorithm.

Observation 4. Let us have 2-connected components B_i and B_j of the above graph H . Suppose that these 2-connected components are connected by a path $[p_1, p_2, \ldots, p_c]$ in the graph H consisting of single vertices only (the path does not contain any 2-connected component) where $p_1 \in B_i$ and $p_c \in B_j$. It is possible to move any robot from the component B_i to any vertex of the component B_j if and only if there are at least $c+1$ unoccupied vertices in the tree of the graph H that contains B_i and B_j (see figure 6). ■

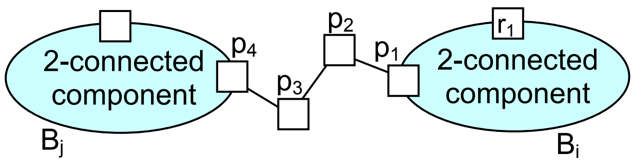

Figure 6: *Illustration to observation 4.* To move a robot r_1 from right 2-connected component to the left 2-connected component (to any vertex of this component) we need unoccupied path between components and an extra unoccupied vertex within the left 2-connected component.

Sketch of proof. We already know that any vertex of a connected component can be made unoccupied assuming that there is at least one unoccupied vertex in this component. Hence we can make unoccupied a single vertex (different from p_c) within the 2-connected component B_j and we can make unoccupied the path $[p_1, p_2, \ldots, p_c]$. Now it is easy using already discussed observations to move any robot from B_i to p_1 while the component B_j and $p_2, p_3 \ldots, p_c$ remain unaffected. Then it is possible to move the robot from the vertex p_1 to the vertex p_c . Finally, we can move the robot to any vertex of the component B_j since we have an additional unoccupied vertex in this component.

If there are less than $c+1$ unoccupied vertices it is easy to construct a counter example where the robot cannot be moved to another 2-connected component. ■

The complete solving algorithm for the general graph is based on the observation 4. First we compute the decomposition of the given graph which of the result is the forest of trees where each vertex of this forest is represented by a 2-connected component. Using observation 4 we move all the robots into their goal 2-connected components. We proceed inductively from leaves to the root of the decomposition. Robots are moved into the leaf 2-connected component which results into the task of the same type but smaller. When all the robots are in their goal 2-connected components then we can switch to the specialized algorithm for 2-connected components. This can be done in polynomial time.

Altogether when we have the MRPP problem with at least two unoccupied vertices we can solve this problem in polynomial time.

4. Domain-Independent Planners and MRPP/MROPP

We performed an experimental evaluation with several state-of-the-art domain-dependent planners in order to find out whether the described tractable class of the MRPP problem is useful. The questions may be: Do the planners perform well on the MRPP problem or not? If not how the defined tractable can help?

We used the following planners for evaluation: *SGPlan 5* [5, 6], *SATPlan* [7, 8], and *LPG* [2, 3]. These planners ranked among the best in the last *International Planning Competition* (IPC-5) [1]. SGPlan and SATPlan search for optimal plans. That is, they solve the MROPP problem in fact. LPG searches for satisfying plans. That is, the solution plans do not need to be optimal (so it the MRPP problem in fact). We tested these three planners on a small collection of MRPP/MROPP problems of various difficulties. The collection of problems used for evaluation is available at the web site: http://ktiml.mff.cuni.cz/~surynek/research/ecai2008/. The results are presented in table 1.

Almost all the tested problems belong to the defined tractable class. Almost all the problems are solvable (that is, a solution exists). Generally, if the problems are viewed as MRPP (optimal solution is not required) they are easy to solve using the above process. When optimal solution is required the situation is different. However, in this case the tractable class would provide a quick answer if the problem is unsolvable.

The presented results show that today's planners do not cope well with MRPP/MROPP problems. Even for the collection of extremely easy problems planners do not provide any answer in the given time. Moreover it seems that requirement of optimal solution is too strong and causes the planner to run out of time. On the other hand the results are unexpectedly bad even without requiring optimality of the solution. These results capacitate us to make following conclusions. The defined tractable class can help today's planners to solve MRPP problem in reasonable time. Moreover, it can be useful for MROPP problem in case of non-existence of the solution.

From the wider point of view the presented experimental results show that a domain-dependent view of the problem is useful since domain-independent planners seem to be too general and not as efficient as the structure of the problem allows.

Table 1: *Experimental evaluation of planners on MRPP problems.* Timeout of 120 seconds is used. Tests were run on an AMD Opteron 1600MHz, 1GB memory, under Mandriva Linux 10.2.

Problem	**Number of vertices/ free**	**Tractable**	**Solvable**	**Plan length/ optimal**	**SGPlan** (seconds)	**SATPlan** (seconds)	**LPG** (seconds)
01	7/6	Yes	Yes	1/1	0.00	0.01	0.01
02	8/7	Yes	Yes	6/6	0.00	0.04	0.02
04	8/6	Yes	Yes	25/16	0.00	0.09	0.01
05	8/5	Yes	Yes	216/32	0.00	0.87	0.98
06	12/10	Yes	Yes	12/12	0.00	0.04	0.01
07	12/8	Yes	Yes	178/26	0.00	0.14	0.08
08	12/7	Yes	Yes	176/36	0.03	0.50	0.37
09	12/6	Yes	Yes	64/46	0.04	0.52	1.54
10	12/5	Yes	Yes	72/56	0.04	>120.0	3.46
11	12/4	Yes	Yes	112/60	0.05	>120.0	4.36
12	12/3	Yes	Yes	170/98	0.17	>120.0	5.25
13	13/4	Yes	Yes	154/112	0.61	>120.0	5.94
14	12/2	No	No	N/A	>120.0	>120.0	<10.0
15	13/3	Yes	Yes	N/A	>120.0	>120.0	>120.0
16	12/10	Yes	Yes	10/10	0.00	0.04	0.02
17	12/8	Yes	Yes	30/24	0.02	0.16	0.03
18	12/4	Yes	Yes	124/N/A	>120.0	>120.0	1.45
19	12/3	Yes	Yes	114/78	0.76	>120.0	6.23
20	12/2	Yes	Yes	208/120	0.33	>120.0	7.58
21	12/1	No	Yes	216/170	0.91	>120.0	8.00
22	16/2	Yes	No	N/A	>120.0	>120.0	>120.0
23	14/2	Yes	No	N/A	>120.0	>120.0	>120.0
24	28/20	Yes	Yes	72/64	0.08	>120.0	0.11
25	28/12	Yes	Yes	564/N/A	>120.0	>120.0	16.39
26	28/4	Yes	Yes	N/A	>120.0	>120.0	>120.0

5. Conclusions and Future Work

In this paper we defined class of the multiple robot path planning problems for that a polynomial time solving algorithm exists. The class covers a broad range of problems of this type. In other words, problems not covered by the class represent quite extreme cases (which should not appear in practice too frequently). This is the first reason why the defined class is worthwhile.

We performed an experimental evaluation with several state-of-the-art domain-independent planners to find out how they are able to solve the multiple robots path planning problems. The experimental evaluation proved that the problem is difficult for today's planners. Moreover, the planners performed very badly on quite easy problems belonging to the tractable class. The conclusion we made upon these experimental results is that it is better to reason about the multiple robots path planning problem from the domain-dependent point of view. Hence specialized algorithms for the problem seem to be promising. The definition of tractable class of the problem is exactly following this research direction.

Nevertheless, there are still some questions remaining open. How is it with tractability where there is only one unoccupied vertex in the 2-connected graph? What about tractability of optimal version of the problem with more than one unoccupied vertex (for a single unoccupied vertex a negative result is already known [10])? These are the theoretical questions that should be answered in the future work. There is certain evidence that the MRPP problem with a single unoccupied vertex can be solved in polynomial time as well [14].

We also need to fill in the gap in the experimental evaluation. First, it is necessary to implement the solving algorithm for the MRPP based on the defined tractable class. Next, we need to make a performance comparison with the standard search algorithms as well as with the approach proposed in [11, 12].

6. Acknowledgement

This work is supported by the Grant Agency of Charles University (Grantová agentura Univerzity Karlovy - GAUK) under the contract number 356/2006/A-INF/MFF and by the Ministry of Education, Youth and Sports, Czech Republic (Ministerstvo školství, mládeže a tělovýchovy - MŠMT) under the contract number MSM 0021620838.

7. References

[1] Gerevini, A., Bonet, B., Givan, B. (editors): 5th International Planning Competition. Event in the context of ICAPS 2006 conference, United Kingdom, http://ipc5.ing.unibs.it, University of Brescia, Italy, (January 2008), 2006.

[2] Gerevini, A., Serina, I.: LPG: a Planner based on Local Search for Planning Graphs. In Proceedings of the 6th International Conference on Artificial Intelligence Planning and Scheduling (AIPS-02), 13-22, AAAI Press, 2002.

[3] Gerevini, A., Serina, I.: Homepage of LPG. Research web page. http://zeus.ing.unibs.it/lpg/, University of Brescia, Italy, (January 2008).

[4] Ghallab, M., Nau, D. S., Traverso, P.: Automated Planning: theory and practice. Morgan Kaufmann Publishers, 2004.

[5] Hsu, C. W., Wah, B. W., Huang, R., Chen, Y. X.: Handling Soft Constraints and Prefer-ences in SGPlan. In Proceedings of the ICAPS Workshop on Preferences and Soft Con-straints in Planning, event in the context of the ICAPS 2006 conference, United Kingdom, 2006.

[6] Hsu, C. W., Wah, B. W., Huang, R., Chen, Y. X.: SGPlan 5: Subgoal Partitioning and Resolution in Planning. Research web page. http://manip.crhc.uiuc.edu/programs/SGPlan/index.html, University of Illinois, USA, (January 2008).

[7] Kautz, H., Selman, B., Hoffmann, J.: SATPlan: Planning as Satisfiability. Abstracts of the 5th International Planning Competition, event in the context of ICAPS 2006 conference, United Kingdom, 2006.

[8] Kautz, H., Selman, B., Hoffmann, J.: SATPLAN. Research web page. http://www.cs.rochester.edu/u/kautz/satplan/index.htm, University of Rochester, USA, (January 2008).

[9] McDermott, D.: PDDL: the Planning Domain Definition Language. Technical Report. Yale Center for Computational Vision and Control, Yale University, CT, USA, 1998.

[10] Ratner, D., Warmuth, M. K.: Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable. Proceedings of the 5th National Conference on Artificial Intelligence. (AAAI 1986), 168-172, Morgan Kaufmann Publishers, 1986.

[11] Ryan, M. R. K.: Multi-robot path planning with sub-graphs. Proceedings of the 19th Australasian Conference on Robotics and Automation, Auckland, New Zeland, Australian Robotics & Automation Association, 2006.

[12] Ryan, M. R. K.: Graph Decomposition for Efficient Multi-Robot Path Planning. Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, 2003-2008, IJCAI Conference, http://www.ijcai.org, (February 2008).

[13] West, D. B.: Introduction to Graph Theory, second edition. Prentice-Hall, 2000.

[14] Wilson, R. M.: Graph Puzzles, Homotopy, and the Alternating Group. Journal of Combinatorial Theory, Ser. B 16, 86-96, Elsevier, 1974.