Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

An Optimization Variant of Multi-Robot Path Planning is Intractable

Pavel Surynek

Charles University in Prague
Faculty of Mathematics and Physics
Department of Theoretical Computer Science and Mathematical Logic
Malostranské náměstí 25, 118 00 Praha 1, Czech Republic
pavel.surynek@mff.cuni.cz

Abstract

An optimization variant of a problem of path planning for multiple robots is addressed in this work. The task is to find spatial-temporal path for each robot of a group of robots such that each robot can reach its destination by navigating through these paths. In the optimization variant of the problem, there is an additional requirement that the makespan of the solution must be as small as possible. A proof of the claim that optimal path planning for multiple robots is NP-complete is sketched in this short paper.

Introduction and Motivation

The problem studied in this paper is a variation of a well known task of *pebble motion on a graph* (also known as *pebble motion puzzle*, specially (n^2-1)-*puzzle*) (Kornhauser et al., 1984). This task is given as an undirected graph and a set of pebbles placed in vertices of the graph. At most one pebble is placed in each vertex and at least one vertex remains unoccupied. The dynamicity of the problem is defined by a notion of an ***allowed move***, which is a move of a pebble into a neighboring vertex that must be unoccupied at the time of commencing the move. The task is to find a partially ordered set of allowed moves such that all the pebbles reach their target vertices. Notice that if there is more than one unoccupied vertex, then several moves can be done in parallel at a single time step.

The problem addressed in this paper differs slightly with regard to the notion allowed moves. Here, the constraint on allowed move is slightly relaxed to make the problem a more realistic abstraction for motion tasks. Again, an undirected graph and a set of entities in vertices (now they are called *robots*) are given and the task is to rearrange them. An allowed move is defined **transitively**: an allowed move is a move into a neighboring unoccupied vertex and it is also a move into a neighboring vertex that is being left by another robot using an allowed move. Let this variation be called a problem of ***path planning for multiple robots***[1] (Ryan, 2008; Surynek, 2009).

It has been shown in (Ratner & Warmuth, 1986) that finding a shortest possible sequence of moves that solves an instance of the problem of pebble motion on a graph when there is a single unoccupied vertex is NP-hard. Unfortunately, the proof does not work for multi-robot path planning since parallelism may occur even if there is only one unoccupied vertex. So the question is how difficult (computationally) is the optimization variant of the problem of path planning for multiple robots? A sketch of proof that this task is NP-complete is given in this short paper.

The NP-Completeness of the Problem

Definition 1 ***(path planning for multiple robots).*** Let $G=(V,E)$ be an undirected graph and let $R=\{\bar{r}_1,\bar{r}_2,\ldots,\bar{r}_\nu\}$ be a set of robots where $\nu<|V|$. The ***initial arrangement*** of robots is defined by a simple function $S_R^0\colon R\longrightarrow V$ (that is $S_R^0(r)\neq S_R^0(s)$ for every $r,s\in R$ such that $r\neq s$); the ***goal arrangement*** of robots is defined by another simple function $S_R^+\colon R\longrightarrow V$. A problem of ***multi-robot path planning*** is the task to find a number ζ called a ***makespan*** and a sequence $\mathcal{S}_R=[S_R^0,S_R^1,\ldots,S_R^\zeta]$ where $S_R^k\colon R\longrightarrow V$ is a simple function for every $k=1,2,\ldots,\zeta$ while $\mathcal{S}_R$ must satisfy the following constraints:

(i) $S_R^\zeta=S_R^+$, that is, robots finally reach their destinations.
(ii) Either $S_R^k(r)=S_R^{k+1}(r)$ or $\{S_R^k(r),S_R^{k+1}(r)\}\in E$ for every $r\in R$ and $k=1,2,\ldots,\zeta-1$.
(iii) If $S_R^k(r)\neq S_R^{k+1}(r)$ and $S_R^k(s)\neq S_R^{k+1}(r)$ $\forall s\in R$ such that $s\neq r$, then the move of r at the time step k is ***allowed*** (a move into an unoccupied vertex). If

This work is supported by The Czech Science Foundation (Grantová agentura České republiky - GAČR) under the contract number 201/09/P318 and by The Ministry of Education, Youth and Sports, Czech Republic (Ministerstvo školství, mládeže a tělovýchovy ČR – MŠMT ČR) under the contract number MSM 0021620838.

[1] This term has been first used in (Ryan, 2008); indeed for the already described problem of pebble motion on a graph. Instead of using two terms for the same concept, an attempt to shift the term *multi-robot* towards problems with higher intrinsic parallelism is made here.

$S_R^k(r) \neq S_R^{k+1}(r)$ and there is $s \in R$ such that $s \neq r \wedge S_R^k(s) = S_R^{k+1}(r) \wedge S_R^k(s) \neq S_R^{k+1}(s)$ and the move of s at the time step k is allowed, then the move of r at the time step k is also **allowed**. All the moves of robots must be allowed.

The problem described above is formally a quadruple $\Sigma = (G = (V, E), R, S_R^0, S_R^+)$. □

Notice that an allowed move can be performed either into an unoccupied vertex or into a vertex that is just being left using allowed move (robots can move like a train).

It is expectable, that some solutions are preferred to others in real-life applications. Typically, solutions with the **small makespan** are required. This immediately raises the question whether it is possible to compute a solution of the smallest possible makespan. Let the problem with this additional objective be called an *optimization variant*. A sketch of proof of the ***NP*-completeness** of the optimization variant is shown in the following text.

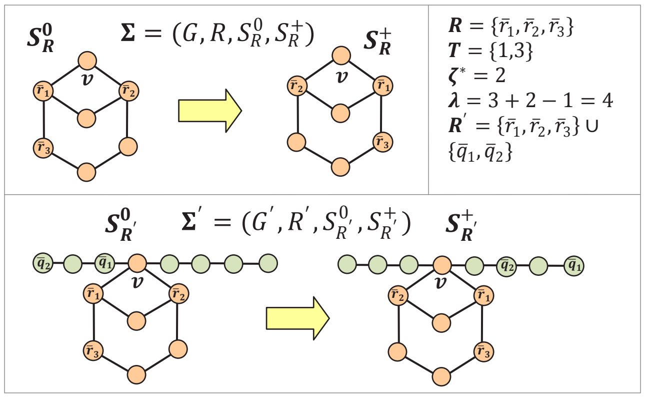

S_R^0
$\Sigma = (G, R, S_R^0, S_R^+)$
S_R^+
v
$\bar{r}_1$
$\bar{r}_2$
$\bar{r}_3$
$R = \{\bar{r}_1, \bar{r}_2, \bar{r}_3\}$
$T = \{1,3\}$
$\zeta^* = 2$
$\lambda = 3 + 2 - 1 = 4$
$R' = \{\bar{r}_1, \bar{r}_2, \bar{r}_3\} \cup \{\bar{q}_1, \bar{q}_2\}$
$S_{R'}^0$
$\Sigma' = (G', R', S_{R'}^0, S_{R'}^+)$
$S_{R'}^+$
$\bar{q}_1$
$\bar{q}_2$

Figure 1. *An illustration of* ***vertex locking***. A vertex v is to be locked at time steps 1 and 3. An augmentation Σ' of an instance Σ is implemented by adding a path around the locked vertex v while newly added robots are enforced to go through v exactly at time step 1 and 3 in any optimal solution of Σ'.

Lemma 1 ***(vertex locking).*** Let $\Sigma = (G = (V, E), R, S_R^0, S_R^+)$ be an instance of the problem of multi-robot path planning and let $v \in V$ be a vertex such that $S_R^0(r) \neq v$ $\forall r \in R$. Next, let $T = \{t_1, t_2, \ldots, t_n\}$ where $t_i \in \mathbb{N}$ and $t_1 < t_2 < \cdots < t_n$ be a set of lock time steps. Then there exists an augmentation of Σ (vertices and robots are added) such that it never happens that a robot enters the vertex v at any time step $t_i \in T$ in any optimal solution. Moreover, the original robots from the set R can move only within G. ■

Sketch of proof. A new path is added to the graph G such that it goes through the vertex v. Then, new robots are placed at the beginning of this path and their destinations are set at the end of this path. The length of the path is set in such a way that it takes at least $\zeta^* + t_n$ time steps before the newly added robots can reach their destinations, where ζ^* is the makespan of any optimal solution of Σ. Thus, the bottleneck on the makespan of an optimal solution is imposed by the newly added path and robots. Additionally new robots are arranged is such a way that they go through the vertex v exactly at time steps from the set T.

It remains to augment the resulting instance to prevent the original robots from entering the newly added vertices (this is an unwanted behavior since new vertices may serve as additional parking place for robots). This can be done by the same technique. Vertices directly preceding v and directly succeeding v on the newly added path are locked for all the time steps except that time steps at which the newly added robots from the first stage of the augmentation need to go through them. ■

Observe that the **size** of the augmented instance is polynomial in size of Σ and ζ^*. Lemma 1 represents the crucial ability to implement so called ***conjugation principle*** for a group of robots. The term conjugation means that all the robots of the group are forced to go either through one way or through the other way in any optimal solution (the group cannot split). The technique of conjugation will be used to simulate ***Boolean consistency***. Observe further, that Lemma 1 can be easily generalized to **lock a set** of vertices instead of only one vertex. A set $W \subseteq V$ can be locked so that at least one vertex of W is not occupied by the original robots from R at selected time steps. This generalization will be used to simulate ***clause satisfaction***.

Lemma 2 ***(conjugation principle).*** Let R^G be a set of robots. There exists an instance of multi-robot path planning problem $\Sigma = (G = (V, E), R, S_R^0, S_R^+)$ with $R^G \subseteq R$ such that conjugation with the group R^G occurs in any optimal solution of Σ. That is, there is $V^{\mathcal{L}} \subseteq V$, $V^{\mathcal{R}} \subseteq V$ with $V^{\mathcal{L}} \cap V^{\mathcal{R}} = \emptyset$ and $|V^{\mathcal{L}}| = |V^{\mathcal{R}}| = |R^G|$ and a time step t such that all the robots of the set R^G are placed either in $V^{\mathcal{L}}$ or in $V^{\mathcal{R}}$ at the time step t in any optimal solution while both alternatives can equally happen. ■

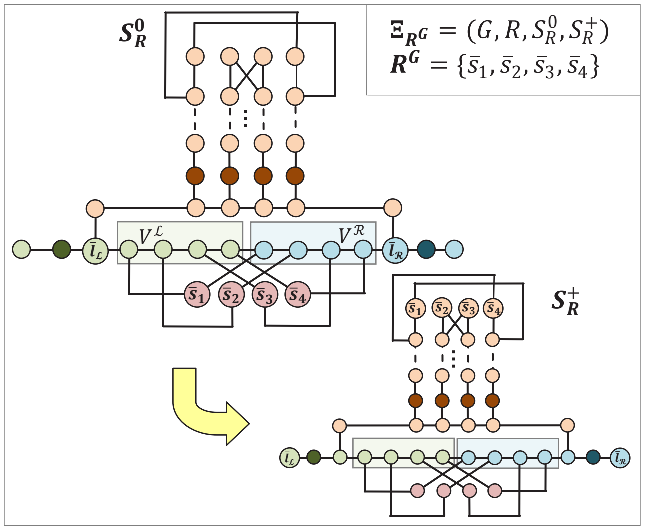
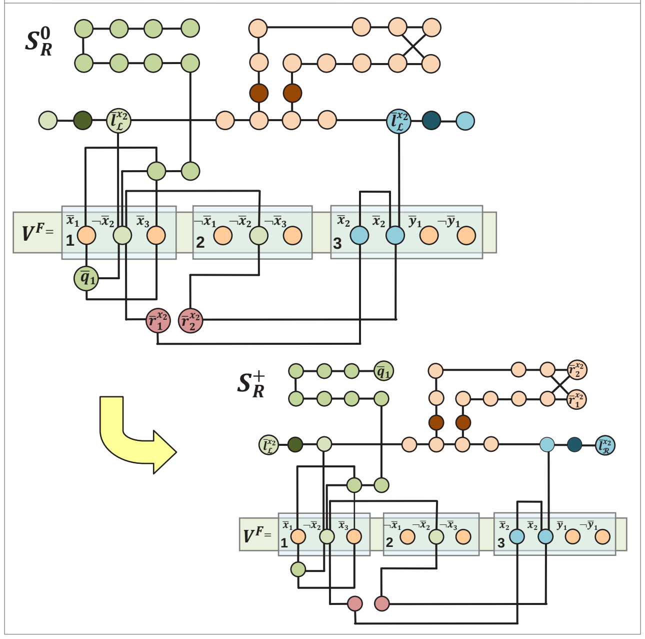


Figure 2. *An instance of the problem of path planning for multiple robots with the* ***conjugation principle*** *for a group of robots* $R^G = \{\bar{s}_1, \bar{s}_2, \bar{s}_3, \bar{s}_4\}$. The bottom line of vertices is open (not locked) only at time step 0. Dark vertices in the 4th line (1st line in on the top) can be entered only at time step 8; dark vertices in the 6th line can be entered only at time step 14. The principle of conjugation is that all the robots $\bar{s}_1$, $\bar{s}_2$, $\bar{s}_3$, and $\bar{s}_4$ are located either in $V^{\mathcal{L}}$ or in $V^{\mathcal{R}}$ at time step 1 within any optimal solution.

$\boldsymbol{F} = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$

$\boldsymbol{F}_= = \underset{1}{(x_1 \vee \neg x_2 \vee x_3)} \wedge \underset{2}{(\neg x_1 \vee \neg x_2 \vee \neg x_3)} \wedge \underset{3}{(x_2 \vee x_2 \vee y_1 \vee \neg y_1)}$

$\boldsymbol{\Sigma/\eta} = (G, R, S_R^0, S_R^+)/11$



S_R^0
S_R^+
$V^{F=}$
$\bar{x}_1$ $\neg\bar{x}_2$ $\bar{x}_3$
$\neg\bar{x}_1$ $\neg\bar{x}_2$ $\neg\bar{x}_3$
$\bar{x}_2$ $\bar{x}_2$ $\bar{y}_1$ $\neg\bar{y}_1$
1
2
3
$l_{\mathcal{L}}^{x_2}$
$l_{\mathcal{R}}^{x_2}$
$\bar{q}_1$
$\bar{r}_1^{x_2}$
$\bar{r}_2^{x_2}$

Figure 3. *A polynomial time* ***reduction*** *of a Boolean formula to a decision instance of multi-robot path planning.* The conjugation technique is used to simulate Boolean consistency and the set locking technique is used to simulate clause satisfaction (reduction of one variable/clause is shown). There exists a solution of Σ of the makespan $\eta = 11$ if and only if the formula F is satisfiable.

Sketch of proof. The instance Σ for a group of robots $R^G = \{\bar{s}_1, \bar{s}_2, \bar{s}_3, \bar{s}_4\}$ is shown in Figure 2. The proof will just briefly comment the figure. The robots need to go from the bottom line to the top line of the graph. Using the construction from Lemma 1, the initial vertices are open only at time step 0. Thus, robots must go to the set of vertices $V^{\mathcal{L}}$ and $V^{\mathcal{R}}$. The dark vertices in 4th and 6th line of the graph are open only at time steps 8 and 14 respectively. Since there are robots $l_{\mathcal{L}}$ and $l_{\mathcal{R}}$ causing an obstruction in entering line 4 of the graph at time step 8 in case the group of robots is divided, all the robots must go either into $V^{\mathcal{L}}$ or into $V^{\mathcal{R}}$ at time step 1. The upper part of the graph can be prolonged as necessary to obtain an optimal solution of the required makespan. It is not difficult to generalize the instance from Figure 2 for an arbitrary group of robots R^G. ■

Again observe that the **size** of the conjugation instance is polynomial in $|R^G|$. At this point, it is easy to polynomially **reduce** an instance of the *Boolean satisfiability problem* (*SAT*) (Cook, 1971) to the decision version of the optimal multi-robot path planning using the above constructions.

Theorem 1 *(NP-hardness).* Let F be an instance of SAT in conjunctive normal form (*CNF*). It is possible to construct an instance of the problem of multi-robot path planning Σ/η in polynomial time such that there exists a solution of Σ of the makespan η if and only if F is satisfiable. ■

Sketch of proof. Observe that F can be transformed to an equisatisfiable $F_=$ in which each Boolean variable has the same number of positive and negative occurrences in polynomial time. Then each occurrence of a literal in $F_=$ is associated with a vertex and each variable in $F_=$ is associated with a **conjugation** subgraph where the size of the group of robots is equal to the number of positive (=negative) occurrences of the variable in $F_=$. Vertices corresponding to negative and positive literals of the same variable are matched to vertices $V^{\mathcal{L}}$ and $V^{\mathcal{R}}$ of the corresponding conjugation subgraph respectively. The height of conjugation subgraphs is set in a way to have the same makespan η of optimal solutions in all these subgraphs.

Literals that correspond to vertices through which a robot goes at time step 1 are set to $FALSE$; otherwise they are set to $TRUE$. The construction guarantees that this assignment is correctly defined (**Boolean consistency** is preserved; it cannot happen that two complementary literals of the same variable are assigned the same truth value).

A satisfied clause must have at least one unoccupied vertex at time step 1. To enforce this **clause satisfaction** the technique of **vertex set locking** is applied on the set of clause vertices. That is, some additional robots are forced to enter at least one vertex of each clause at time step 1. ■

Since there exists a solution of the makespan of $\mathcal{O}(|V|^3)$ for any **solvable** instance of the problem of pebble motion on a graph $G = (V, E)$ (Kornhauser et al., 1984), there is also such a small solution for any solvable instance of multi-robot path planning. Hence, the decision version of the optimization variant of multi-robot path planning is in NP. Altogether, the studied problem is NP-complete.

References

Cook, S. A., 1971. *The Complexity of Theorem Proving Procedures.* Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, pp. 151-158, ACM Press.

Kornhauser, D., Miller, G. L., Spirakis, P. G., 1984. *Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications.* Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984), pp. 241-250, IEEE Press.

Ratner, D., Warmuth, M. K., 1986. *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), pp. 168-172, Morgan Kaufmann Publishers.

Ryan, M. R. K., 2008. *Exploiting Subgraph Structure in Multi-Robot in Multi-Robot Path Planning.* Journal of Artificial Intelligence Research (JAIR), Volume 31, pp. 497-542, AAAI Press.

Surynek, P., 2009. *An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning.* Proceedings of the 21st International Conference on Tools with Artificial Intelligence (ICTAI 2009), pp. 151-158, IEEE Press.