Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs*

Jingjin Yu
Department of Electrical and Computer Engineering
University of Illinois, Urbana, IL 61801
jyu18@uiuc.edu

Steven M. LaValle
Department of Computer Science
University of Illinois, Urbana, IL 61801
lavalle@uiuc.edu

Abstract

In this paper, we study the structure and computational complexity of optimal multi-robot path planning problems on graphs. Our results encompass three formulations of the discrete multi-robot path planning problem, including a variant that allows synchronous rotations of robots along fully occupied, disjoint cycles on the graph. Allowing rotation of robots provides a more natural model for multi-robot path planning because robots can communicate.

Our optimality objectives are to minimize the total arrival time, the makespan (last arrival time), and the total distance. On the structure side, we show that, in general, these objectives demonstrate a pairwise Pareto optimal structure and cannot be simultaneously optimized. On the computational complexity side, we extend previous work and show that, regardless of the underlying multi-robot path planning problem, these objectives are all intractable to compute. In particular, our NP-hardness proof for the time optimal versions, based on a minimal and direct reduction from the 3-satisfiability problem, shows that these problems remain NP-hard even when there are only two groups of robots (i.e. robots within each group are interchangeable).

Introduction

Discrete multi-robot path planning problems seem to have originated from the study of Sam Loyd's 15-puzzle (Loyd 1959; Story 1879), a well known board based puzzle game. The 15-puzzle can be viewed as moving 15 robots on a 16-vertex grid graph, which readily generalizes to the multi-robot path planning problem on a N-vertex graph with $n < N$ robots. In the most basic formulation, only one pebble may move in a time step to an adjacent unoccupied vertex; we call this problem *pebble motion on graphs* or PMG.

Since robots can act autonomously and communicate, multiple robots are capable of moving in the same time step. A *parallel move* of robots is a synchronous move of a (non-self-intersecting) chain of robots as long as the first robot moves into a vertex that is unoccupied at the beginning of the time step. If multiple disjoint parallel moves per time step are allowed, we call this problem variant *multi-robot path planning on graphs with parallel moves*, or MPP_p, which were studied in (Ryan 2008; Surynek 2010), among others.

Feasible moves require unoccupied vertices in PMG and MPP_p formulations. More recently, in a variant of the problem (Yu and LaValle 2012; 2013), robots are allowed to rotate synchronously along fully occupied cycles. It was pointed out in (Yu 2012) that instances having N robots (on a N-vertex graph) can often be feasible. We call this problem *multi-robot path planning on graphs with parallel moves and rotations* or MPP_pr for short. The rotation primitive was also mentioned in a grid setting (Standley and Korf 2011). Arguably, MPP_pr provides a better model for multi-robot path planning problem than MPP_p does for two reasons: (1) when parallel moves are allowed, it is natural to include rotations, and (2) allowing rotations can only reduce the best plan's size, given some optimality criterion.

It is well known that PMG (therefore, MPP_p) is solvable in polynomial time (Kornhauser, Miller, and Spirakis 1984). Moreover, feasibility tests for PMG can be performed in linear time (Goraly and Hassin 2010). These algorithms were generalized to include MPP_pr in (Yu 2012).

Since feasible solutions can be found efficiently, one might be motivated to seek polynomial time optimal solutions to these formulations. For PMG, a distance optimal solution is NP-hard to compute (Goldreich 1984; Ratner and Warmuth 1990). Finding a plan with minimum makespan (i.e., last arrival time) for MPP_p was also shown to be NP-hard (Surynek 2010). However, not much is known about the computational complexity of optimal MPP_pr formulations or optimal MPP_p formulations other than minimum makespan. Moreover, there is a lack of understanding on the structures and relationships between different optimal multi-robot path planning formulations (e.g., whether there is a Pareto front for two different optimality criteria).

In this paper, we address these issues and systematically study three optimality objectives: minimizing the total arrival time, minimizing the makespan, and minimizing the total distance. First, we show that these objectives have a Pareto optimal structure for MPP_p and MPP_pr. That is, any pair of these three objectives cannot be simultaneous optimized for MPP_p or MPP_pr. These objectives are equivalent for the PMG problem. Continuing onto the subject of com-

*This work was supported in part by NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (Cyberphysical Systems), and MURI/ONR grant N00014-09-1-1052. We thank the anonymous reviewers for their helpful suggestions.

putational complexity, we show that computing an optimal solution for any of the three objectives is NP-hard for PMG, MPP$_{\text{p}}$, and MPP$_{\text{pr}}$. We point out that the NP-hardness results without rotations do not carry over to the case that allows rotations because rotations may introduce better optimal solutions that can be computed efficiently.

Problem Formulation

Multi-robot path planning on graphs with parallel moves and rotations[1]

Let $G = (V, E)$ be a connected, undirected, simple graph with vertex set $V = \{v_i\}$ and edge set $E = \{(v_i, v_j)\}$. Let $R = \{r_1, \ldots, r_n\}$ be a set of robots that move with unit speeds along the edges of G, with initial and goal locations on G given by the injective maps $x_I, x_G : R \to V$, respectively. A *path* is a map $p_i : \mathbb{Z}^+ \to V$. A path p_i is *feasible* for a robot r_i if it satisfies the following properties: (1) $p_i(0) = x_I(r_i)$, (2) for each i, there exists a smallest $t_i \in \mathbb{Z}^+$ such that $p_i(t_i) = x_G(r_i)$, (3) for any $t \geq t_i$, $p_i(t) \equiv x_G(r_i)$, and (4) for any $0 \leq t < t_i$, $(p_i(t), p_i(t+1)) \in E$ or $p_i(t) = p_i(t+1)$ (if $p_i(t) = p_i(t+1)$, robot r_i stays at vertex $p_i(t)$ between the time steps t and $t+1$). We say that two paths p_i, p_j are in *collision* if there exists $k \in \mathbb{Z}^+$ such that $p_i(t) = p_j(t)$ or $(p_i(t), p_i(t+1)) = (p_j(t+1), p_j(t))$.

Problem (MPP$_{\text{pr}}$). Given (G, R, x_I, x_G), find a set of paths $P = \{p_1, \ldots, p_n\}$ such that p_i's are feasible paths for respective robots r_i's and no two paths p_i, p_j are in collision.

Synchronized rotations of robots along fully occupied cycles distinguishe MPP$_{\text{pr}}$ from the majority of previously studied multi-robot path planning problems. In an MPP$_{\text{pr}}$ instance, even when the number of robots equals the number of vertices, robots may still be able to move. A simple feasible example here is n robots on an n-cycle, with each robot having the left (assuming an orientation of the cycle in the plane) adjacent vertex as its goal.

Optimality

We examine three common objectives in optimal multi-robot path planning: minimizing the makespan (last arrival time), minimizing the total arrival time, and minimizing the total distance. Formally, let $P = \{p_1, \ldots, p_n\}$ be an arbitrary solution to a fixed MPP$_{\text{pr}}$ instance. For a path $p_i \in P$, $len(p_i)$ denotes the length of the path p_i, which is incremented by one each time when the robot r_i passes an edge. A robot, following p_i, may visit the same edge multiple times. Recall that t_i denotes the arrival time of robot r_i.

Objective 1 (Minimum Total Arrival Time). Compute a path set P that minimizes $\sum_{i=1}^n t_i$.

Objective 2 (Minimum Makespan). Compute a path set P that minimizes $\max_{1 \leq i \leq n} t_i$.

Objective 3 (Minimum Total Distance). Compute a path set P that minimizes $\sum_{i=1}^n len(p_i)$.

[1]We only provide a full description of MPP$_{\text{pr}}$ here. For complete formulations of PMG and MPP$_{\text{p}}$, see (Kornhauser, Miller, and Spirakis 1984; Surynek 2010).

For a PMG problem with a single unoccupied vertex, these objectives are all equivalent because only one robot can move in each time step. Therefore, the NP-hardness result from (Goldreich 1984) implies the following.

Lemma 1. *Computing a minimum total arrival time, minimum makespan, or minimum total distance solution for a PMG problem is NP-hard.*

The decision versions of the opitmal MPP$_{\text{pr}}$ problems are defined as follows.

MTATMPP (Minimum Total Arrival Time MPP$_{\text{pr}}$)
INSTANCE: An instance of MPP$_{\text{pr}}$, and $k \in \mathbb{Z}$.
QUESTION: Is there a solution path set P with a total arrival time no more than k?

M3PP (Minimum Makespan MPP$_{\text{pr}}$)
INSTANCE: An instance of MPP$_{\text{pr}}$, and $k \in \mathbb{Z}$.
QUESTION: Is there a solution path set P with a makespan no more than k?

MTDMPP (Minimum Total Distance MPP$_{\text{pr}}$)
INSTANCE: An instance of MPP$_{\text{pr}}$, and $k \in \mathbb{Z}$.
QUESTION: Is there a solution path set P with a total path distance no more than k?

The Pareto Optimal Structure

In this section, we show that in general, it is impossible to simultaneously optimize multiple objectives for MPP$_{\text{p}}$ and MPP$_{\text{pr}}$. This is true for every pair from Objectives 1-3. Since the incompatibility proof for Objectives 2 and 3 was given in (Yu and LaValle 2013), we show Pareto optimal structures for the other two pairing of the three objectives. For each pair, we provide an infinite family of instances on which the two objectives are optimized by different solutions.

Proposition 2. *For MPP$_{p}$ and MPP$_{pr}$, optimality cannot always be simultaneously achieved for minimum makespan and minimum total arrival time.*

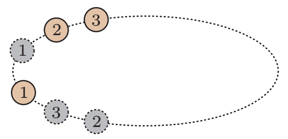

Figure 1: An instance in which the graph is a single cycle. Discs with solid borders are the start locations of robots 1-3 (as numbered) and discs with dotted borders are the goal locations of robots 1-3.

PROOF. In Fig. 1, the start and goal vertices of robots 1-3 are as marked. Let the distance between the consecutive numbered discs on the left side of the oval be one each and let the distance of the right path (between robot 3's vertex and 2's goal) be $x \geq 1$. Clearly, optimal solutions require that all robots move in the same (clockwise or counterclockwise) direction until they reach their goals. If all robots move in the clockwise direction, the cost vector for makespan and total arrival time is $(x+1, 2x+3)$. The cost vector is $(x+4, x+12)$ if the robots move in the counterclockwise direction. Thus, a clockwise move always yields the solution with minimum makespan. However, when $x > 9$, the solution corresponding to counterclockwise movements has a smaller total arrival time. □

Proposition 3. *For MPP$_p$ and MPP$_{pr}$, optimality cannot always be simultaneously achieved for minimum total arrival time and minimum total distance.*

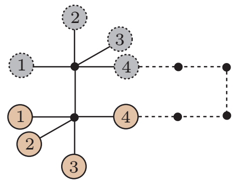

Figure 2: The start locations of robots 1-4 are marked with discs having solid borders (as numbered). Their goals are the numbered discs with dotted borders.

PROOF. In Fig. 2, the start and goal locations of robots 1-4 are as marked. The distance between any adjacent pair of nodes (discs, black dots) is one. The solution with minimum total arrival time sends robots 1-3 through solid paths on the left and robot 4 through the dotted path on the right. This yields a total arrival time of $3 + 4 + 5 + 4 = 16$ and a total distance of $3 + 3 + 3 + 4 = 13$. On the other hand, the solution with minimum total distance sends all robots from the left path, which yields a total arrival time of 18 and a total distance of 12. By extending the lengths of the two vertical edges in the middle, we get an infinite family of examples. □

Intractability of MTATMPP and M3PP

Unlike finding feasible solutions, solving optimal versions of MPP$_{pr}$ appears to be intractable in general. In this section, we provide evidence to this claim by showing that **MTATMPP** and **M3PP** are NP-hard. We give a minimal and direct reduction from **3SAT** (Garey and Johnson 1979) that works for both problems.

Theorem 4. **MTATMPP** *is NP-hard.*

PROOF We reduce **3SAT** to **MTATMPP**. Let (X, C) be an arbitrary instance of **3SAT** with $|X| = n$ variables $x_1, \ldots, x_n$ and $|C| = m$ clauses $c_1, \ldots, c_m$, in which $c_j = y_j^1 \vee y_j^2 \vee y_j^3$. Without loss of generality, we may assume that the set of all literals, y_j^k's, contain both unnegated and negated form of each variable x_i.

From the **3SAT** instance, an **MTATMPP** instance is constructed as follows. For each variable x_i, two paths of length $m + 2$ each, jointed at the end, are added (e.g. the four horizontal strips in the middle of Fig. 3). At the left end of the joined path, vertex v_{x_i}, sits a robot r_{x_i}, with its goal vertex, v'_{x_i}, at the right end. The robot can travel along either of the two paths to reach its goal in $m + 2$ steps. Call these two paths the i-th upper and lower paths.

Then, for each clause $c_j = y_j^1 \vee y_j^2 \vee y_j^3$, add a robot r_{c_j}, sitting at a vertex v_{c_j}. The vertex v_{c_j} is connected to three paths associated with the three variables corresponding to c_j's three literals. If a literal is the unnegated (resp., negated) form of variable x_i, then v_{c_j} is connected to the i-th upper (resp., lower) path at a vertex of distance j from v_{x_i}. For example, if $c_1 = x_1 \vee \neg x_3 \vee x_4$, then v_{c_1} is connected to the first upper, third lower, and fourth upper paths, all at vertices of distance 1 from the left end of the "strips" (see Fig. 3).

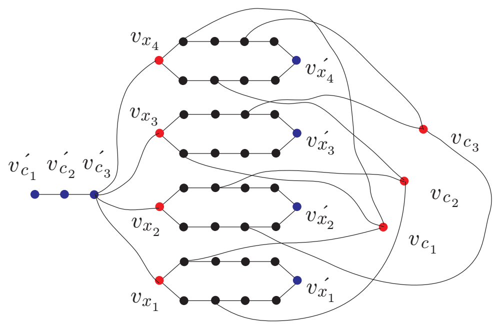

Figure 3: An MPP$_{pr}$ instance constructed from the **3SAT** instance $(\{x_1, x_2, x_3, x_4\}, \{x_1 \vee \neg x_3 \vee x_4, \neg x_1 \vee x_2 \vee \neg x_4, \neg x_2 \vee x_3 \vee x_4\})$. The red vertices are the start vertices and the blues one the goals.

After the clause structures are created, the goals for the r_{c_j}'s are added. For this purpose, a path of length m is added (e.g. the leftmost path with blue vertices in Fig. 3), with the left vertex being the goal for r_{c_1} and the right vertex the goal for r_{c_m}. The goal vertex for r_{c_m}, v'_{c_m}, is connected to all v_{x_i}'s, the start vertices of robots r_{x_i}'s. Having constructed an MPP$_{pr}$ instance, setting $k = (n + m)(m + 2)$ fully describes an instance of **MTATMPP**. Fig. 3 gives the complete graph for the **MTATMPP** instance constructed from the **3SAT** instance $(\{x_1, x_2, x_3, x_4\}, \{x_1 \vee \neg x_3 \vee x_4, \neg x_1 \vee x_2 \vee \neg x_4, \neg x_2 \vee x_3 \vee x_4\})$.

If the **3SAT** instance is satisfiable, let $\widetilde{x}_1, \ldots, \widetilde{x}_n$ be an assignment of the truth values to the variables. For each variable x_i, if $\widetilde{x}_i$ is true (resp., false), then let robot r_{x_i} take the lower (resp., upper) path on its strip. The upper (resp., lower) path is then free to use for transporting the robots corresponding to the clauses, r_{c_j}'s. All $m + n$ robots can start moving at time step zero and arrive at their desired goals at time step $m + 2$. The total time is then $(m + n)(m + 2)$.

On the other hand, if the MPP$_{pr}$ instance have a solution with total arrival time $(n+m)(m+2)$, then every robot must start moving at time step zero, follow a shortest path, and never stop until it reaches its goal. This forces every robot r_{x_i} to take either the upper or lower path on its own strip, which prevents any robot r_{c_j} from using the same path in the opposite direction. If robot r_{x_i} uses the upper (resp., lower) path, let $\widetilde{x}_i = true$ (resp., $false$). The resulting assignment $\widetilde{x}_1, \ldots, \widetilde{x}_n$ satisfies the **3SAT** instance. □

Corollary 5. **M3PP** *is NP-hard.*

PROOF. In the proof of Theorem 5, after the MPP$_{pr}$ instance is created, setting $k = m + 2$ as the minimum makespan produces a **M3PP** instance from the **3SAT** instance. The rest of the proof remains essentially the same. □

In our many-one reduction, it is clear that rotations of robots along cycles do not contribute to better paths. Therefore, the reduction works for time optimal MPP$_p$ problem as well. In particular, our proof greatly simplifies the NP-hard proof of minimum makespan MPP$_\text{p}$ from (Surynek 2010).

Corollary 6. *Finding a minimum total arrival time or a minimum makespan solution for MPP$_p$ is NP-hard.*

The reduction illustrates one reason that makes finding time optimal solutions hard: When multiple robots want to travel in opposite directions on a few shared paths, it is critical that the right paths are picked if time optimality is sought. Moreover, our proof shows an even stronger intractability result: Computing a time optimal solution is NP-hard even when there are only two groups of robots (i.e., the robots within each group are interchangeable).

Theorem 7. $\mathbf{MTATMPP}$ *and* $\mathbf{M3PP}$ *remain NP-hard, even when there are only two groups of robots.*

PROOF. In the reduction from $\mathbf{3SAT}$, let the variable robots belong to one group and the clause robots belong to another group. □

Intractability of MTDMPP

Unfortunately, the simple structure from Fig. 3 is not as useful in proving the NP-hardness of $\mathbf{MTDMPP}$ because there is no need for the robots to synchronize their movements unless they are forced to. It is possible, however, to force such a synchronization, as shown in (Ratner and Warmuth 1990), in which $\mathbf{2/2/4\ \ SAT}$ is reduced to the distance optimal (n^2-1)-puzzle. $\mathbf{2/2/4\ \ SAT}$ is a specialized version of the boolean satisfiability problem.

$\mathbf{2/2/4\ \ SAT}$
INSTANCE: An instance of the boolean satisfiability problem with m boolean variables and m clauses. Each clause has exactly four literals and each variable appear four times in the clauses, twice negated and twice unnegated.
QUESTION: Does the instance have a satisfiable assignment?

$\mathbf{2/2/4\ \ SAT}$ is NP-hard and has the property that given a satisfying assignment, each clause has exactly two true literals and two false literals (Ratner and Warmuth 1990). Once rotation is allowed, the proof from (Ratner and Warmuth 1990) (or (Goldreich 1984)) no longer works because its synchronization scheme depends on the fact that only robots near the only unoccupied vertex may move.

Proof outline. To show that $\mathbf{MTDMPP}$ is NP-hard, we adapt the construction from (Ratner and Warmuth 1990) with some significant changes. To reduce proof complexity, we will build an MPP$_\text{pr}$ instance such that all vertices are occupied by robots. The essential idea behind the main construct of the reduction (Fig. 6, to be introduced in detail shortly) is to force the robots to go through a predetermined "path" along the construct. If the robots are to deviate from this path, significant extra distance cost will be incurred. On the other hand, the construct ensures that the robots, following the predetermined "path", can reach the desired goals if and only if the associated $\mathbf{2/2/4\ \ SAT}$ instance is solvable.

To build a new scheme for synchronizing robots' movements in an optimal solution, we need several gadgets. The first gadget (see e.g., Fig. 4) allows distance optimal transportation of three robots. In the structure, there are $2\ell+2$

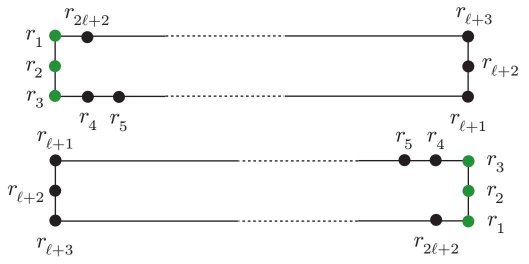

Figure 4: A gadget for optimally transporting three robots in the middle path. [top] Initial configuration of robots. [bottom] The final configuration.

robots and r_1, r_2, r_3 are the robots to be transported. The starts and goals for these three robots may be temporary; the starts and goals for all other robots are final. We call such a gadget a *forward path*. In a forward path, each robot must move at least a distance of ℓ to reach its goal. The gadget can only be joined to other structures at the two short sides in such a way that all shortest paths between any robot $r_i \in \{r_4, \ldots, r_{2\ell+2}\}$ and its goal are within the forward path. Furthermore, any path connecting r_i and its goal without using a long side of the forward path must have a distance at least 2ℓ. It is clear that the optimal total distance for all robots, including r_1-r_3, is $2\ell^2+2\ell$.

Proposition 8. *Transporting multiple groups (one group must reach the right end before another group can be transported) of robots through a forward path incurs an extra distance of $\Omega(\ell)$ for robots $r_4, \ldots, r_{2\ell+2}$.*

PROOF SKETCH. Each group of robots to be transported must use a long side of a forward path and pushes all other robots on the long side through with them. It can be checked (simple but tedious case analysis and counting are involved) that intermediate configurations between transporting different groups of robots will require robots $r_4, \ldots, r_{2\ell+r}$ to deviate from optimal paths by at least $\Theta(\ell)$ in total. □

Proposition 9. *Transporting a single group of more than three robots through a forward path incurs an extra distance of $\Omega(\ell)$ for robots $r_4, \ldots, r_{2\ell+2}$.*

PROOF SKETCH. When more than three robots are in a forward path at the same time, some robot(s) in $r_4, \ldots, r_{2\ell+4}$ cannot stay on the forward path and must travel extra distance. This induces an extra cost of at least $\Theta(\ell)$. □

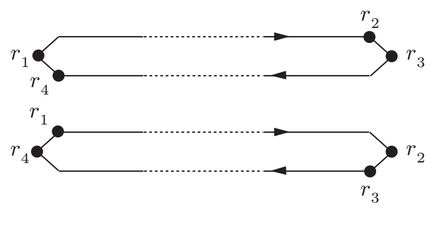

Figure 5: A gadget for synchronizing the movements of robots. [top] Initial configuration. [bottom] Final configuration.

The second gadget given in Fig. 5 consists of a single cycle (formed by two paths joined at the ends) that will be connected to other gadgets at the two end vertices on the left and right. The length of the cycle is 8ℓ. Denote such a gadget a *backward path*. The function of a backward path is to push r_1 into the cycle and r_2 out of the cycle as a synchronization mechanism. Every robot in the middle of the cycle have its goal one vertex to its left or right as indicated by the arrows. An optimal solution for a backward path is to rotate all robots in the direction of the arrows, which yields a total distance of 8ℓ. Before the rotation, r_1 may come from elsewhere to its start location and after the rotation, r_2 may move to elsewhere. All other start and goal locations are final. The optimal cost for transporting all robots including r_1, r_2 is 8ℓ. It is clear that if r_1, r_2 are not moved at the same time, then an extra cost of at least 4ℓ is incurred. Note that moving a robot through a backward path incurs $\Omega(\ell^2)$ cost to the robots on the path.

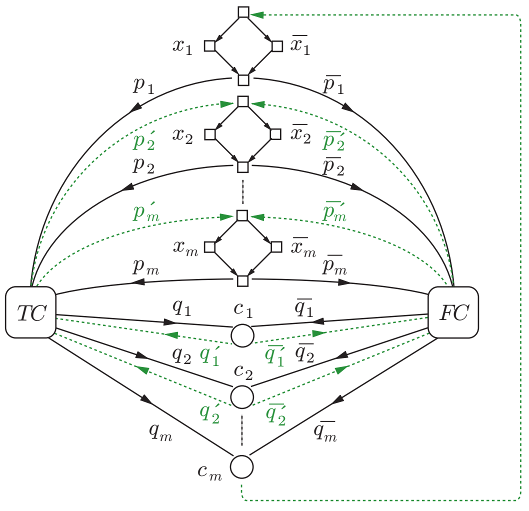

Figure 6: Reduction of $\mathbf{2/2/4}$ **SAT** to **MTDMPP**.

The third and the main construct (similar to that used in (Ratner and Warmuth 1990)) is given in Fig. 6, constructed from a $\mathbf{2/2/4}$ **SAT** instance with m variables. In the construct, each solid edge $(p_i, q_j, \overline{p_i}, \overline{q_j})$ represents a forward path and each dotted edge $(p_i', q_i', \overline{p_i'}, \overline{q_i'})$ a backward path.

On the top half (above the squares marked TC and FC) there are m diamond structures. We call these the *variable diamonds*. The details of a variable diamond is given in Fig. 7. The start locations for robots a_i-f_i and $x_{i1}, x_{i2}, \overline{x_{i1}}, \overline{x_{i2}}$ (the robots representing the unnegated and negated literals) are given in the figure. The goal locations of b_i, c_i are start locations of x_{i1}, x_{i2}, respectively. Same goes for e_i, f_i. The literals will be moved out of the variable diamond. The goals of a_i, d_i are in the next variable diamond (the goals of a_{i-1}, d_{i-1} are marked as dotted circles in Fig. 7).

The squares on the sides, TC and FC, each contains a strip of $3m$ vertices and robots. TC has the structure given in Fig. 8; the structure of FC is similar. On the bottom half of Fig. 6 there are m *clause nodes*, the structure of the j-

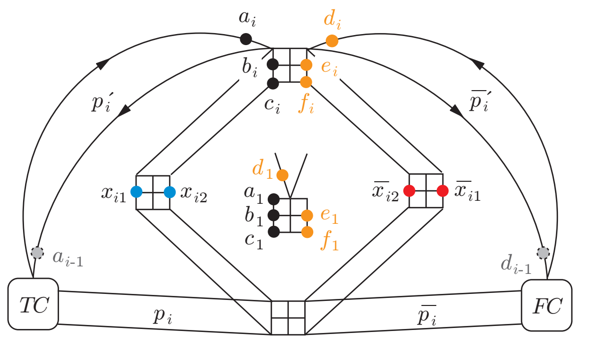

Figure 7: The structure of a variable diamond. The top of the variable diamond for x_1 is slightly different and is shown in the bottom right corner.

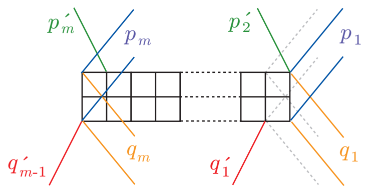

Figure 8: The gadget for temporarily hosting the true literals.

th node is given in Fig. 9. These clause nodes host the goal locations for the $4m$ literals (these are y_{j1}-y_{j4}). The start locations of g_j, h_j and goal locations of g_{j-1}, h_{j-1} are as marked. The goal location of g_{m-1} will be given shortly; the goal of h_{m-1} is an arbitrary unused location in the last (m-th) clause node.

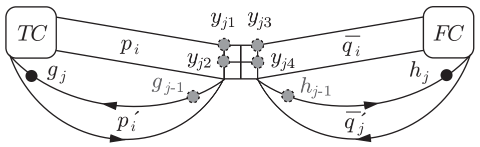

Figure 9: The structure of the clause node j.

Finally, the backward path connecting the last clause node and the first variable diamond is given in Fig. 10, which specifies the start location of d_1 and goal location of g_{m-1}. So far the start and goal locations of almost all robots are specified, with the exception of some robots in the 3×3 grids, TC, FC, and near the ends of backward paths. The goals for these robots can be set arbitrarily as long as they remain local with respect to their start location (i.e. within a constant distance) and consistent.

So far, a full MPP$_{\text{pr}}$ problem has been constructed from the $\mathbf{2/2/4}$ **SAT** instance. Recall that we require a forward path to be joined to the rest of the graph such that for an arbitrary robot $r_i \in \{r_4, \ldots, r_{2\ell+2}\}$ on the forward path, a path connecting r_i and its goal must have a distance of 2ℓ or more if it does not pass through the forward path itself. It can be checked that this is satisfied by the MPP$_{\text{pr}}$ instance. We set $\ell = m^4$.

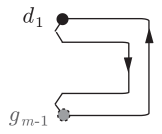

Figure 10: The backward path connecting the last clause node and the first variable diamond.

Lemma 10. *If an instance of* $\mathbf{2/2/4}$ $\mathbf{SAT}$ *is satisfiable, then the corresponding MPP$_{pr}$ problem has a solution with a total distance of* $16m^9 + 48m^5 - 24m^4 + O(m^2)$.

PROOF. Suppose that the $\mathbf{2/2/4}$ $\mathbf{SAT}$ instance is satisfiable. Let $\widetilde{x}_1, \ldots, \widetilde{x}_m$ be a satisfying assignment to the variables $x_1, \ldots, x_m$. The paths for taking the robots to their goals are described below.

The first moves take a_1 to TC. If $\widetilde{x}_1$ is true, a_1, b_1, c_1 can be transported through the top left forward path in the first variable diamond. If $\widetilde{x}_1$ is false, using a constant number of moves (see Proposition 5 in (Yu and LaValle 2013)), a_1 can be exchanged with the robot at the top right corner of the top 3×3 grid of the first variable diamond (i.e., on top of e_1, f_1). Such *local* rearrangements will be assumed from now on without explicitly stating so. Then a_1, e_1, f_1 will take the top right forward path. Without loss of generality, assume that the right path is taken. Once a_1, e_1, f_1 get to the right 3×3 grid in the first variable diamond, e_1, f_1 stay and $a_1, \overline{x_{11}}, \overline{x_{12}}$ can be moved to the bottom 3×3 grid of the variable diamond. They can then be moved to TC using the left forward path.

Once a_1 is in TC, it can be used to free a_2 in p_2'. Inductively, all the $2m$ literals that are set to true can be collected to TC along with a_m. These literals can then be distributed to the clause nodes, two at a time (including $a_m, g_1, \ldots, g_{m-1}$, three robots will actually be transported at a time). Since $\widetilde{x}_1, \ldots, \widetilde{x}_m$ is a satisfying assignment, TC contains the robots such that two of which have goals in each clause node. Once g_{m-1} gets to the last clause node, it can then free d_1 on the top, and the right half of the paths can be "traversed" so that all robots can reach their desired goals.

There are $8m$ forward paths and $4m - 3$ backward paths. These induce a total distance cost of $8m(2\ell^2 + 2\ell) + (4m - 3)(8\ell) = 16m^9 + 48m^5 - 24m^4$, plus some local rearrangements. These local rearrangements can be performed with a total distance cost of $O(m^2)$ (again, see Proposition 5 in (Yu and LaValle 2013)). □

Lemma 11. *If the MPP$_{pr}$ problem reduced from an* $\mathbf{2/2/4}$ $\mathbf{SAT}$ *instance has a solution with a total distance of* $16m^9 + 48m^5 - 24m^4 + O(m^2)$*, the* $\mathbf{2/2/4}$ $\mathbf{SAT}$ *instance is satisfiable.*

PROOF. Through straightforward counting, it can be checked that the least amount of distance connecting the start and goal locations of the robots is $16m^9 + 48m^5 - 24m^4 + O(m^2)$. For such a total distance to be achievable, the forward and backward paths must be followed in a pattern similar to that from the proof of Lemma 10 because if not, an extra cost of $\Omega(\ell) = \Omega(m^4)$ is incurred by Propositions 8, 9 and properties of backward paths. This means that a forward path can only be used to transport a single group of no more than three robots and robots on a backward path can only move once (excluding robots at the two end vertices). Otherwise, the $\Omega(m^4)$ extra cost will take the total distance cost to $16m^9 + 48m^5 - 24m^4 + \Omega(m^4)$, which is strictly larger than $16m^9 + 48m^5 - 24m^4 + \Omega(m^4)$ for large enough m.

Suppose that a feasible solution path set with a total distance $16m^9 + 48m^5 - 24m^4 + O(m^2)$ exists. At the beginning, no backward path can be taken due to the synchronization/locking mechanism. For example, the backward path connecting the last clause node and the first variable diamond cannot be used because g_{m-1} is not in the last clause node. This suggests that the only possible first move (without incurring an $\Omega(m^4)$ extra cost) is to move three robots, a_1 with c_1, d_1 or e_1, f_1, along the top left or top right forward path in the first variable diamond (since a_1-f_1 must all travel down and no forward path can transport more than three a time or multiple groups, each forward path must take three robots). Following this argument, $2m$ of the $4m$ literal robots must go to TC and the other $2m$ must go to FC. The robots going to FC must have one pair of literals (either unnegated or negated, but not both) per variable. These robots then must end in the clause nodes with each clause node getting two literals from TC and two from FC. Setting the literals corresponding to the literal robots passing through TC yields a good assignment. □

Lemmas 10 and 11 prove to the following theorem.

Theorem 12. $\mathbf{MTDMPP}$ *is NP-hard, even if all vertices are occupied by robots.*

PROOF. After the MPP$_{\text{pr}}$ instance is constructed, set $k = 16m^9 + 48m^5 - 24m^4 + m^3$ proves the claim for large enough m. □

General Intractability of Optimal Multi-robot Path Planning Problems on Graphs

We conclude this paper with the following main result.

Theorem 13. *Computing a minimum total arrival time, a minimum makespan, or a minimum total distance solution is NP-complete for PMG, MPP$_p$, and MPP$_{pr}$.*

PROOF. Lemma 1 covers the PMG part. Theorem 4, Corollary 5, and Theorem 12 cover MPP$_{\text{pr}}$. It is also clear that Theorem 4 and Corollary 5 generalizes to MPP$_{\text{p}}$ without modification because the optimal paths do not use synchronous rotations of robots. We are left to show that computing a distance optimal solution for MPP$_{\text{p}}$ is NP-hard. This is again covered by the distance optimal result from (Ratner and Warmuth 1990) because parallel moves do not shorten the total distance traveled.

These problems are NP-complete because PMG, MPP$_{\text{p}}$, and MPP$_{\text{pr}}$ are in NP (Kornhauser, Miller, and Spirakis 1984; Yu 2012). □

References

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

Goldreich, O. 1984. Finding the shortest move-sequence in the graph-generalized 15-puzzle is np-hard. Laboratory for Computer Science, Massachusetts Institute of Technology, unpublished manuscript.

Goraly, G., and Hassin, R. 2010. Multi-color pebble motion on graph. *Algorithmica* 58:610–636.

Kornhauser, D.; Miller, G.; and Spirakis, P. 1984. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS '84)*, 241–250.

Loyd, S. 1959. *Mathematical Puzzles of Sam Loyd*. New York: Dover.

Ratner, D., and Warmuth, M. 1990. The $(n^2 - 1)$-puzzle and related relocation problems. *Journal of Symbolic Computation* 10:111–137.

Ryan, M. R. K. 2008. Exploiting subgraph structure in multi-robot path planning. *Journal of Artificial Intelligence Research* 31:497–542.

Standley, T., and Korf, R. 2011. Complete algorithms for cooperative pathfinding problems. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 668–673.

Story, E. W. 1879. Note on the '15' puzzle. *American Journal of Mathematics* 2:399–404.

Surynek, P. 2010. An optimization variant of multi-robot path planning is intractable. In *The Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 1261–1263.

Yu, J., and LaValle, S. M. 2012. Multi-agent path planning and network flow. In *The Tenth International Workshop on Algorithmic Foundations of Robotics*.

Yu, J., and LaValle, S. M. 2013. Planning optimal paths for multiple robots on graphs. In *Proceedings IEEE International Conference on Robotics & Automation*. to appear.

Yu, J. 2012. Diameters of permutation groups on graphs and linear time feasibility test of pebble motion problems. *arXiv:1205.5263*.