Tractability Frontiers in Multi-Robot Coordination and Geometric Reconfiguration*

Tzvika Geft[1]**, Dan Halperin[2], and Yonatan Nakar[2]

[1] Rutgers University, New Brunswick NJ, USA
[2] Tel Aviv University, Israel

Abstract. We study the Monotone Sliding Reconfiguration (MSR) problem, in which *labeled* pairwise interior-disjoint objects in a planar workspace need to be brought *one by one* from their initial positions to given target positions, without causing collisions. That is, at each step only one object moves to its respective target, where it stays thereafter. MSR is a natural special variant of Multi-Robot Motion Planning (MRMP) and related reconfiguration problems, many of which are known to be computationally hard. A key question is identifying the minimal mitigating assumptions that enable efficient algorithms for such problems. We first show that despite the monotonicity requirement, MSR remains a computationally hard MRMP problem. We then provide additional hardness results for MSR that rule out several natural assumptions. For example, we show that MSR remains hard without obstacles in the workspace. On the positive side, we introduce a family of MSR instances that always have a solution through a novel structural assumption pertaining to the graphs underlying the start and target configuration—we require that these graphs are spannable by a forest of full binary trees (SFFBT). We use our assumption to obtain efficient MSR algorithms for unit discs and 2D grid settings. Notably, our assumption does not require separation between start/target positions, which is a standard requirement in efficient and complete MRMP algorithms. Instead, we (implicitly) require separation between *groups* of these positions, thereby pushing the boundary of efficiently solvable instances toward denser scenarios.

Keywords: Monotone rearrangement · Multi-robot motion planning · Reconfiguration · Complexity

1 Introduction

We are given a set of labeled pairwise interior-disjoint objects (e.g., robots) in a polygonal workspace that need to be brought from their initial positions to given target positions, without causing collisions. Such a setup appears in a host of geometric reconfiguration problems, such as Multi-Robot Motion Planning (MRMP),

* Work by T.G. and D.H. has been supported in part by the Israel Science Foundation (grant no 2261/23), by NSF/US-Israel-BSF (grant no. 2019754), by the Blavatnik Computer Science Research Fund, by the Israeli Smart Transportation Research Center, and by the Shlomo Shmelzer Institute for Smart Transportation at Tel Aviv University.

** Work by T.G. was carried out while the author was at Tel Aviv University.

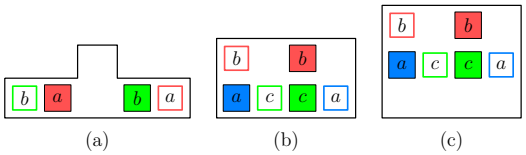

Fig. 1: MSR instances. Filled (resp. unfilled) squares are start (resp. target) positions. (a) A "No" instance, as one of the squares must make two moves. (b) An instance that can be solved by moving the squares in the order b,a,c and no other order. (c) A modification of (b) to a well-formed instance (see text).

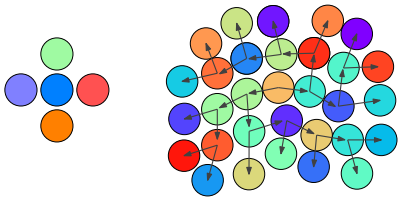

Fig. 2: Two start configurations featuring disks with little to no separation between them, which are allowed under our assumption. The arrows (right) illustrate the core concept of SFFBT (see text).

which vary by how a *move* of an object is defined and additional constraints or objectives. A *sliding move* [20] brings an object to another location in the plane using a continuous rigid motion. In the *sequential* case, only one object moves at a time, i.e., a solution is a sequence of single object moves.

Given that reconfiguration problems are often computationally hard, identifying the least restrictive assumptions that enable solving them efficiently is a major quest. Motivated by this question, we study a natural special case of sequential reconfiguration with sliding moves, which we call *Monotone Sliding Reconfiguration (MSR)*. MSR asks to find a *monotone* solution, where each object moves at most once. See Figure 1. MSR appears to be the simplest variant in the long-studied aforementioned family of problems for which no efficient algorithm is known, thus motivating us to close this gap. In turn, we perform a detailed investigation of the problem's computational complexity, which establishes its NP-hardness and rules out natural mitigating assumptions. On the positive side, we identify a key structural assumption that guarantees a solution.

Background and related work. Mobile robots' continuously increasing role in manufacturing, logistics, and many other domains requires coordinating their motion safely and efficiently, giving rise to the MRMP problem. In MRMP robots may move in parallel and make multiple moves, i.e., a robot stopping (to let another robot pass) and moving again is allowed. Already the feasibility version of MRMP (where the motions need not be optimal) is known to be intractable in various planar settings [8, 27, 28, 33, 35]. Nevertheless pinpointing exactly when and why the problem is computationally hard, as a guide towards useful tractable variants, remains a major challenge. For example, the complexity of MRMP for unit-disc robots is a longstanding open problem [8]. While for unit-square robots the problem is PSPACE-hard [33], the proof uses a polygonal workspace with holes and an unbounded complexity.[3] A similar tradeoff occurs in a classic PSPACE-hardness result [28], which uses a rectangular workspace but relies on rectangular robots of various sizes. These examples highlight the challenge in understanding the complexity of MRMP when *multiple facets*, such as robot shape

[3] The workspace's complexity depends on the number of robots and cannot be expressed as a constant.

and workspace shape, are *simultaneously* simplified. We strive towards this goal by considering MSR as a basic yet non-trivial MRMP variant, which we further restrict through additional assumptions. Related work involving two of the authors shows that minimizing the total distance traveled by the robots is APX-hard in a weakly monotone setting [4] (see below) and NP-hard for robots on a 2D grid in the monotone (and non-monotone) case [25]. Both works fall short of directly addressing the issue of monotonicity, which is at the heart of the investigation of this work.

As MRMP hardness proofs often rely on tightly packed robots, complete[4] polynomial-time MRMP algorithms [3, 5, 12, 18, 34, 36] make *separation assumptions*, such as a minimum distance between robots at their start or target positions. Another assumption for labeled unit-disc robots in a polygonal workspace [4, 32] requires each robot's start and target position to be contained in a *revolving area*, i.e., a radius 2 disc that is free from other robots' start and target positions. The setting lends itself to *weakly* monotone solutions, in which robots move one by one as in MSR, except that while one robot moves to its target, the other robots may move locally within their respective revolving areas. Separation is also implicitly required by a prevalent assumption, which is that an MRMP instance has a *well-formed environment* (WFE) [11] (see definition below).

Identifying the minimal separation that guarantees a solution that can be efficiently found is key for enabling robots to operate in crowded settings. A recent work provides tight bounds for unlabeled[5] unit discs, showing that a distance of 4 between pairs of start positions and pairs of target positions is needed [5]. Nevertheless, without such a separation a solution can still exist and the complexity of MRMP in such a case remains a challenging problem. In particular, all the aforementioned assumptions exclude having just one robot tightly surrounded by others in the start or target configuration, such as the 5 discs in Figure 2 (left). Consequently, in this work we tackle the key question of which milder assumptions suffice to efficiently solve MRMP.

Related to MRMP is sequential reconfiguration in the plane, which has been investigated under various models and settings [1, 6, 10, 20, 26]. Abellanas et al. [1] study the *translation model*, in which a move is a single translation along a fixed direction, and provide lower and upper bounds on the required number of moves to reconfigure n discs in various settings. They also give an $O(n^2)$-time algorithm for the special case of MSR under the translation model. Dumitrescu and Jiang [20] show that minimizing the number of moves in the reconfiguration of (labeled or unlabeled) unit discs is NP-hard for either the translation or sliding model. We use *Reconfiguration with Minimum Number of Moves (RMNM)* to refer to the labeled case of their problem in our model. MSR is a special case of the latter requiring the least possible number of moves. The discrete version of RMNM, where objects are pebbles on a graph, was shown to be NP-hard on the 2D grid [10] and W[1]-hard on general graphs [13]. For more related work see references within [19, 26].

[4] An algorithm is *complete* if, in finite time, it returns a solution or reports that none exists.

[5] In the *unlabeled* case any robot can go to any target, provided all the targets are eventually occupied.

Similar problems arise as *rearrangement* problems in robotics, e.g., when a robotic arm needs to rearrange products on a shelf, where the combinatorial problem nature results in various challenges [22,23,24]. Minimizing the number of moves shortens the time needed to perform such tasks, e.g., through fewer pick and place operations. In this vein, MSR for unit discs has been recently tackled empirically, where the problem was shown to be challenging already for 10-30 discs [39], thus further motivating the study of MSR.

Contribution

Negative results. First, we provide a simple construction that establishes the NP-hardness of MSR. We then present stronger hardness results through additional restrictions along two facets. In the first facet, we examine two previously studied tractable MSR variants and show that an immediate generalization of each of them becomes intractable: In a *well-formed environment (WFE)* [11] every robot has a path from its start to its target position that is *endpoint-free*, which is a path that does not intersect any other robot's start or target position. The assumption is strong in the sense that under it the robots may move to their targets one by one in *any* order. We show that for the slightly milder *nearly well-formed environment (nearly WFE)*, where only one robot does not have an endpoint-free path, MSR is NP-hard. Next, for MSR under the translation model [1] we show that if only one designated robot is allowed a sliding move instead of a single translation, then the problem becomes NP-hard as well.

The second facet of our results tackles a key element of previous MRMP and reconfiguration hardness proofs–obstacles in the workspace (Section 2.2). We show that MSR remains hard without obstacles in conjunction with another assumption: We require that the start and target configurations are *line separable*, i.e., there exists a vertical line such that all start positions are left of it and all target positions are right of it. Line separability prohibits robots from emulating obstacles by staying in place (which would give a trivial negative statement). Here we distinguish between a *bounded* workspace (in which the robots are confined) and an *unbounded* workspace (allowing robots to move anywhere in the plane). We consider the negative result for the latter case to be our strongest negative result as this setting is significantly less constrained and hence requires the most intricate construction. Theorem 1 summarizes the results.

These results point to a scheduling problem called Pivot Scheduling [2] as the underlying source of the intractability of all studied variants. We therefore turn towards a structural assumption that avoids the hardness of Pivot Scheduling.

Positive results. For the discrete version of MSR, we define a family of pebble motion graphs (PMGs) that always have a solution through a novel structural assumption. We show that if a PMG is *spannable by a forest of full binary trees (SFFBT)*, or *spannable* for short, (see Section 3 for details) then the MSR instance has a solution that can be efficiently found given the forest. Next, we turn to the problem of recognizing spannable PMGs, i.e., deciding whether a discrete MSR instance belongs in our family. Although we show that this problem is NP-hard for general graphs, we obtain positive results for geometric MSR instances. Specifically, for unit-disc

robots we show that we can efficiently compute a corresponding PMG and decide whether it is spannable. We obtain this result by showing that in this case spannable PMGs have constant treewidth, which allows applying Courcelle's theorem [14] (a brief background is given in Section 4). We also prove that a special variant of MSR for grid-aligned unit squares in which the start and target configurations induce a set of simple thin polyominoes (defined in Section 4.4) always has a spannable PMG.

Notably, through our structural assumption we show that a solution can be guaranteed and efficiently found under a relaxation of standard separation assumptions used in complete and efficient MRMP algorithms. In our setting, we replace separation between individual (non-overlapping) start/target positions with a requirement for separation between "groups" of such positions. A potential consequence is enabling more space-efficient object reconfiguration [26] and deployment of robot teams. This work can also be seen as a step towards formal workspace design [31], where guaranteeing solutions is crucial, in higher density.

Notation and terminology. In MSR we are given a set R of n robots (or objects), which are either all axis-parallel squares or all unit discs. The robots operate in a planar, possibly polygonal, workspace, where each $r \in R$ has a start and target position, denoted by $s(r)$ and $t(r)$, respectively, which we also call *endpoints*. A *solution* to MSR is a *monotone motion plan*, which is a sequence of moves in which each robot $r \in R$ moves *once* from $s(r)$ to $t(r)$ along a collision-free path $\pi(r)$ while all other robots are stationary. If such a solution exists for a MSR instance we call it *feasible* or *solvable*. We distinguish between two variants with regard to the robots' uniformity: In MSR($\square$) R and the obstacles consist of grid-aligned unit squares, which is equivalent to the discrete case of pebbles moving on the 2D grid or grid-aligned unit discs. In MSR($\square$,$\square$) R consists of *nearly identical squares*, i.e., each square has a side length of 1 or $1+\varepsilon$, for an arbitrarily small $\varepsilon > 0$.

2 Hardness results

In this section, we establish the following hardness results:

Theorem 1. *MSR is NP-hard in each of the following variants:*

(i) Nearly WFE variant of MSR($\square$,$\square$),
(ii) MSR($\square$) where all the squares except one must move via a single axis-parallel translation,

and the following obstacle-free line-separable variants:

(iii) MSR($\square$) for a $3 \times \ell$ rectangular workspace,
(iv) MSR($\square$,$\square$) for an unbounded workspace.

We first establish the hardness of MSR (without additional assumptions) using a reduction from a variant of Pivot Scheduling [2], which obtains a MSR instance M. Then, for each MSR variant in Theorem 1, we show how to convert M to an instance M' of the variant such that M is solvable if and only if M' is solvable. The details for Theorem 1(i) and (ii) appear in Appendix A.1.[6]

[6] The appendix is attached as supplementary material.

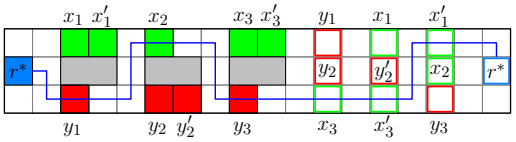

Fig. 3: The MSR instance M for the Pivot Scheduling instance with the before constraints $X_1 = \{x_1, x_1'\}, Y_1 = \{y_1\}, X_2 = \{x_2\}, Y_2 = \{y_2, y_2'\}, X_3 = \{x_3, x_3'\}, Y_3 = \{y_3\}$ and after constraints $\mathcal{C} = \{\{y_1, y_2, x_3\}, \{x_1, y_2', x_3'\}, \{x_1', x_2, y_3\}\}$. Obstacles appear in gray. The start and target positions are the filled and unfilled colored squares, respectively. The start and target positions of the $R(X_i)$ (resp. $R(Y_i)$) robots are green (resp. red). Labels distinguish between robots having the same color. The path $P_{B,A}$ (blue) is shown for the solution $B = \{y_1, x_2, y_3\}, A = \{x_1, x_1', y_2, y_2', x_3, x_3'\}$.

2.1 Establishing the hardness of MSR

Let us define the base problem for our reduction.

Pivot Scheduling. Let V be a set of jobs. Let $\mathcal{X} = \{X_1, ..., X_n\}$ and $\mathcal{Y} = \{Y_1, ..., Y_n\}$ be $2n$ pairwise disjoint subsets of V, i.e., $X_i, Y_i \subseteq V$ for each i. Let $\mathcal{C} = \{C_1, ..., C_m\}$, where $C_j \in V^3$ for each j, and the C_j's are pairwise disjoint. The problem is to determine whether V can be partitioned into a before-set B and an after-set A (i.e., $A \cap B = \emptyset$ and $A \cup B = V$) under the following constraints. *Before constraints:* For each $i \in [n]$, either $X_i \subseteq B$ or $Y_i \subseteq B$. *After constraints:* For each $j \in [m]$, we have $C_j \cap A \neq \emptyset$.

One can think of each constraint as being imposed by an implicit *pivot job* where the jobs of B are to be executed before this pivot job and the jobs of A are to be executed after it. For an example instance and a solution see Figure 3. A similar version of the problem is known to be NP-hard [2]. Our formulation differs slightly for convenience. We prove its hardness using a reduction from a special 3SAT variant [16] (see Appendix A.3).

Lemma 1. *Pivot Scheduling is NP-hard even when* $|X_i| = 3$, $|Y_i| = 1$ *for all* i.

A unified approach for proving the hardness of MSR variants. The common elements of our proofs are as follows. We represent each job by a robot and additionally use a distinguished *pivot robot* r^* (or multiple such robots) that emulates the implicit pivot job in Pivot Scheduling. The robot r^* starts at the left side of the workspace and then has to proceed rightward through *gadgets* that represent Pivot Scheduling constraints. First, r^* proceeds through *before-constraint* gadgets. For such a gadget, which we denote by B_i, r^* must choose between one of two paths to traverse the gadget. Each path is initially blocked by robots, which have to move before r^*. Robots on one path correspond to X_i while those on the other path correspond to Y_i. After proceeding through all before-constraint gadgets, r^* has to proceed through after-constraint gadgets. Each after-constraint gadget, denoted by A_j, can be traversed using one of three paths, each passing through a target of a robot representing a job in C_j. Therefore, for r^* to traverse the gadget, one of the paths has to have an unoccupied target, i.e., A_j enforces that one of the jobs in C_j appears in the after-set A.

Hardness of MSR. Given a Pivot Scheduling instance $\mathcal{P} = (V, \mathcal{C})$, we construct a MSR instance $M := M(\mathcal{P})$ that is feasible if and only if $\mathcal{P}$ is feasible. The workspace

of M is a three-row high rectangular portion of the unit grid; see Figure 3. All the start and target positions, as well as obstacles, are grid cells. There is a robot in M for each job in V and a distinguished pivot robot r^*. For a set of jobs $V' \subseteq V$, we denote by $R(V')$ the robots that correspond to the jobs in V'. All the non-pivot robots' start positions are located at before-constraint gadgets. For each pair X_i,Y_i in $\mathcal{P}$ the gadget B_i has obstacle cells in its middle row, which form a top (resp. bottom) path initially containing the robots $R(X_i)$ (resp. $R(Y_i)$). The B_i's are separated from each other by columns of empty cells.

The non-pivot robots' target positions are located in after-constraint gadgets, each corresponding to an after constraint C_j. The gadget consists of a column of three cells, which are the targets of $R(C_j)$. The after-constraint gadgets are separated from each other by columns of empty cells. Finally, r^* must traverse the workspace from left to right passing through all gadgets. The order of gadgets of the same type is arbitrary but determines the order of the start positions: The left to right order of start positions within each before-constraint gadget is set to match the left to right order of the corresponding target positions. We refer to this ordering as the *start-target ordering correspondence*.

Theorem 2. *MSR is NP-complete.*

Proof. Given a motion plan for M, let R_1 and R_2 denote the robots that move before and after r^* in the motion plan. We have a correspondence between valid job partitions (B,A) and (R_1,R_2) pairs, which describe motion plans for M. Given a valid motion plan, the corresponding partition of V satisfies all the constraints: Each before-constraint is satisfied since either all the robots in $R(X_i)$ or all the robots in $R(Y_i)$ must move so that r^* can traverse B_i to its target. Similarly, each after-constraint is satisfied since one of the targets of each A_j must be unoccupied by a robot for r^* to traverse the gadget.

For the other direction, we show that if (B,A) is a valid job partition, the robots can move in the order $R(B)$,r^*,$R(A)$. We first define the path for r^*, which we refer to as the **partition path** and denote by $P_{B,A}$. We take $P_{B,A}$ to be the shortest collision-free path from $s(r^*)$ to $t(r^*)$ that passes through start positions of the robots in $R(B)$ and targets of the robots in $R(A)$ (and no other endpoints). $P_{B,A}$ exists in M since (B,A) is valid, meaning that each gadget contains the start/target positions that enable $P_{B,A}$ to traverse it.

The robots in $R(B)$, which are initially located on $P_{B,A}$, move in the right to left order of their start positions as follows. When a robot $r \in R(B)$ moves, it moves along $P_{B,A}$ until it reaches the after-gadget that contains its target position, at which point the robot strays off $P_{B,A}$ to reach its target. Due to the order in which $R(B)$ move, when r moves any starting position through which it passes is unoccupied. Any target that r passes through is also unoccupied, since the robots of $R(A)$ have not moved at this stage. Therefore, r's motion is collision-free. Next, r^* moves to its target using $P_{B,A}$. At this point, all the start and target positions that lie on $P_{B,A}$ are unoccupied, so this move is collision-free. Finally, the robots of $R(A)$ move in the right to left order of the target positions, similarly to robots of $R(B)$. Due to the order in which $R(A)$ move, when some $r \in R(A)$

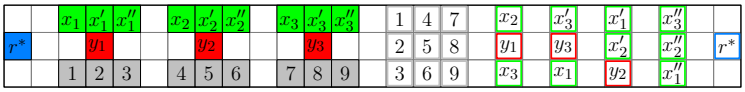

Fig. 4: The instance M_3' obtained from the instance $M(\mathcal{P})$ where $\mathcal{P}$ is a Pivot Scheduling instance with the before constraints $X_1 = \{x_1, x_1', x_1''\}, Y_1 = \{y_1\}, X_2 = \{x_2, x_2', x_2''\}, Y_2 = \{y_2\}, X_3 = \{x_3, x_3', x_3''\}, Y_3 = \{y_3\}$ and after constraints $\mathcal{C} = \{\{x_2, y_1, x_3\}, \{x_3', y_3, x_1\}, \{x_1', x_2', y_2\}, \{x_3'', x_2'', x_1''\}\}$.

moves any target position through which it passes is unoccupied. The start-target ordering correspondence ensures that any start position that r passes through is also unoccupied. Therefore, r's motion is collision-free and so overall we described a valid motion plan. Lastly, NP membership follows from standard techniques for translational motion planning in the plane [17, Chapter 13]. □

2.2 Hardness without obstacles

In this section we prove the hardness of MSR when there are no obstacles in the workspace. We assume here that M (the instance in Theorem 2) has $|R(X_i)| = 3$ and $|R(Y_i)| = 1$ in each B_i, which is valid due to Lemma 1. Omitted details and proofs appear in the full version of this section in Appendix A.2.

Bounded workspace. We show that MSR(□) remains hard in a rectangular workspace without obstacles. The key idea is to modify the before gadgets in M so that obstacles are now *obstacle robots*: Each B_i occupies a 3×3 subgrid where the top row contains $R(X_i)$, the middle cell in the middle row contains the single robot in $R(Y_i)$, and the bottom row contains three obstacle robots. The targets of the three obstacle robots are placed in a single column in one of newly inserted columns; see Figure 4. Notice that r^* may only traverse each B_i by passing through the start positions of $R(X_i)$ or $R(Y_i)$, as in M; the only other option is for r^* to pass through all three start positions of obstacle robots, though if these robots move before r^*, they will form an untraversable barrier.

Unbounded workspace. We now show that MSR is NP-hard for nearly identical squares in an unbounded workspace with no obstacles using an instance M_4'; refer to Figure 5. Similarly to the previous reduction here we also emulate obstacles with robots, which now emulate boundaries in the workspace, and are called *boundary robots*. We use *left (resp. right) boundary* to refer to the boundary that partially encloses the start (resp. target) positions. For this reduction we use multiple pivot robots. All the pivot robots and right boundary robots have a $1+\varepsilon$ side length; the rest of the robots are unit squares. The exact number of robots and overall dimensions of M_4' are made more precise in Appendix A.2.

Left boundary. The part of M_4' with start positions is partially enclosed by the *left boundary*, which forms a *square room* (large red square in Figure 5), where the pivot robots initially reside. The square room is connected to a narrow corridor, the *left corridor*, on its right, leading to another corridor formed by the right boundary, the *right corridor*. The left boundary, a two-layer thick wall of unit squares, marks the start positions of boundary robots (gray). These unit squares are grid-aligned except for a section where the left corridor's height narrows. The left corridor initially has

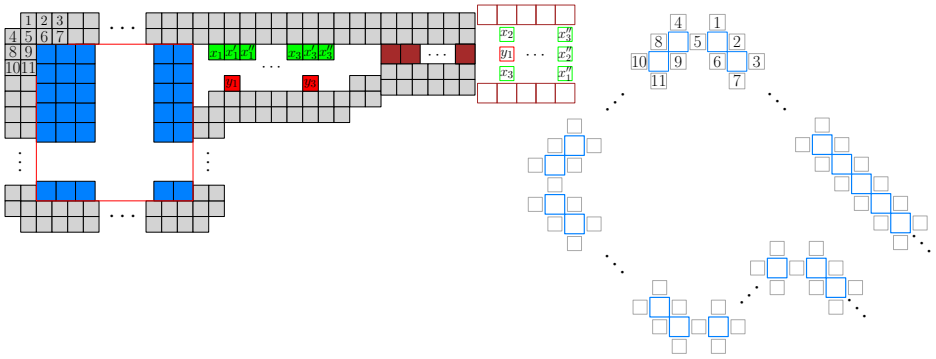

Fig. 5: An example of M'_4 for $\varepsilon = 1/4$ in the format of Figure 3 except that gray and brown colors denote left and right boundary robots, respectively. For decluttering, we omit most robot labels; see text.

a height of 3, to allow placing before-constraint gadgets in it. Each gadget is 3×3 portion of the grid, with the top row containing $R(X_i)$'s and the bottom containing $R(Y_i)$. After the before-gadgets, the corridor narrows to a height of $1+\varepsilon$. This narrower portion contains the start positions of right boundary robots (brown).

Mapping the left boundary to target positions. We describe the target configuration of the left boundary robots using a bijective transformation of their start positions. The transformation takes the centers of the start positions and then rigidly rotates them by 45° clockwise. We then uniformly scale the rotated points so that they are farther apart by a factor of $(2+\varepsilon)/\sqrt{2}$. Target positions are then placed so their centers lie on the resulting points. The resulting configuration is placed in the unbounded area of the workspace. The labels in Figure 5 illustrate the mapping of the top-left portion of the boundary to corresponding target positions. Notice that any 2×2 block of start positions corresponds to a "diamond" of four target squares. We use this to enforce that the left boundary is in place when the pivot robots move by placing the target of each pivot robot inside a diamond.

The right corridor has a height of $3(1+\varepsilon)$ and contains appropriately spaced after gadgets. The corridor's right opening leads to the unbounded area of the workspace, while its left opening is placed $\varepsilon/2$ units away from the opening of the left corridor. This ensures that pivot robots can only reach the unbounded area of the workspace through the right corridor.

Overall, M'_4 emulates M since the first pivot robot that moves must move through all gadgets, similarly to r^* in M. This holds since if the first pivot robot to move tries to escape the left boundary, then that would cause a target of some pivot robot to become unreachable.

3 A monotone solution in pebble motion graphs

To state our assumption for MSR instances we define a monotone version of a Pebble Motion Graph (PMG) problem [30]. A PMG is a simple graph $G=(V,E)$, also called a *motion graph*, where V consists of start vertices (S), target vertices (T), with $|S|=|T|$, and non-endpoint vertices (Z). Each robot r is represented by a pebble that is

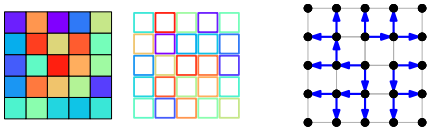

Fig. 6: MSR(□) instance (left; same format as Figure 1, cells of Z not drawn) alongside (right) $G[S]$ and a spanning forest (blue arrows) showing that S is SFFBT. As $G[T]$ has an analogous forest, the complete instance is spannable.

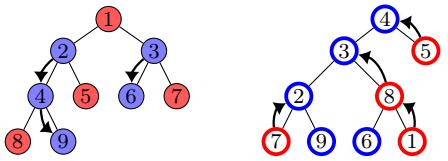

Fig. 7: Illustration of Theorem 3 for a forest with a tree for start positions (left) and a tree for targets (right), showing a partition of the robots into B (blue) and A (red). The B robots move in the order 9,6,4,3,2 to the target tree (via Z, not shown) using paths indicated by the arrows.

initially located at a distinct start vertex $u \in S$ and needs to be moved to its respective target $v \in T$. Such a move is allowed if there is a path from u to v whose vertices (except u) do not contain a pebble. Note that while the location of a pebble changes after each move, G remains the same. A solution to the PMG problem brings all the pebbles to their respective targets using one move at a time, with at most $|S|$ moves. We now define our assumption that guarantees a solution to the PMG problem.

Definition 1. *We say that the subset of vertices $V' \subseteq V$ in G is* spannable by a forest of full binary trees *(*SFFBT *or* spannable *for short) if there exists a spanning forest of $G[V']$, the subgraph of G induced by V', where each tree $\mathcal{T}$ in the forest satisfies the following: $\mathcal{T}$ is full binary tree (i.e., every node has either 0 or 2 children) and every leaf of $\mathcal{T}$ is adjacent to some vertex $v \in Z$.*

Definition 2. *We say that G is* spannable by a forest of full binary trees *(*SFFBT *or* spannable *for short) if (i) S,T,Z are pairwise disjoint, (ii) the set Z of non-endpoint vertices is connected, and (iii) each of S and T is spannable by a forest of full binary trees.*

Figure 6 and Figure 2 (right) illustrate the definitions.

3.1 Solving an instance with a spannable motion graph

Solving the instances in our MSR hardness proofs boils down to choosing when the pivot robot r^* moves with respect to other robots. This choice is induced by a Pivot Scheduling problem where either X_i or Y_i contain multiple elements. We now show that spannable PMGs have a solution because they induce a Pivot Scheduling variant with singleton X_i's and Y_i's. We show that the latter always has a solution.

Lemma 2. *Let $\mathcal{P}$ be a Pivot Scheduling instance where $|X_i|=|Y_i|=1$ for all i and $|C_j|>1$ for all j.*[7] *Then $\mathcal{P}$ is feasible and can be solved in linear time.*

[7] For this special case of Pivot Scheduling we write each before-constraint as $B_i=\{j_1,j_2\}$ instead of $X_i=\{j_1\}$, $Y_i=\{j_2\}$.

The proof is via a reduction of $\mathcal{P}$ to a SAT formula that is always satisfiable; see Appendix A.3.

Theorem 3. *A PMG instance in which G is SFFBT has a solution. If the corresponding spanning forest of G is provided, then an ordering of the pebble robots in which they can be moved to their targets one by one can be found in $O(n)$ time, for n pebbles.*

Proof. Given such a motion graph G along with the corresponding forest, we first show how to construct a Pivot Scheduling instance $\mathcal{P}$ that satisfies the requirements of Lemma 2 (and is thus feasible). Then we apply Lemma 2 to get a solution to $\mathcal{P}$, which we use to obtain a monotone motion plan.

We define $\mathcal{P}$ as follows. For each robot we have a corresponding job in $\mathcal{P}$. For simplicity, we use the same notation to refer to a robot, its corresponding job, and nodes corresponding to the robot's two endpoints in S and T. For each internal node u in the forest, let u_ℓ and u_r be its left and right children. For each internal node u in the forest spanning S, we have a before-constraint with $B_u = \{u_\ell, u_r\}$, and for each internal node u' in the forest spanning T, we have an after-constraint $A_{u'} = \{u'_\ell, u'_r\}$. Note that there are no before-constraints (resp. after-constraints) on the roots of the trees of S (resp. T) in the forest. It is straightforward to verify that $\mathcal{P}$ is solvable by Lemma 2.

We now show how to use the solution to $\mathcal{P}$ to construct a monotone motion plan. Let (B,A) be a solution to $\mathcal{P}$. We first move the robots corresponding to B one by one in increasing order of the height of their start nodes, i.e., in a bottom-up order. Since each parent node u in B imposes a before constraint on two of its children, at least one child appears before u in the ordering. Therefore, such an ordering allows us to move each robot $u \in B$ out of the tree in which it starts to the connected set of vertices Z. Namely, u can reach Z by repeatedly moving from its present node to a child node corresponding to a robot of B (which would be empty by the chosen order), until reaching a leaf, which is adjacent to a node of Z, by definition. See Figure 7.

We now claim that there exists a robot-free path π by which u can continue its motion from Z to its target position. The path π exists since each target position $v \in T$ corresponds either to a leaf in the forest (in which case it is directly reachable from Z) or to an after-constraint A_v. In the latter case, at least one of the children in A_v does not belong to B, and, by induction on height, v is again reachable from Z, regardless of which robots in B have already moved. More precisely, π can be traced in reverse starting from the target of u by repeatedly proceeding to a child node that is a target of a robot of A (which is empty at this point since the robots of A has not yet moved), until reaching a leaf.

It remains to move all the robots corresponding to A. We apply a symmetric argument to the one we have applied for the B robots, by using the after-constraints: Let us assume for a moment that all the robots are at their target positions. We can then move the robots of A one by one to their start positions using the same argument as above, where we switch the roles of A and B. Following the resulting order and corresponding paths in reverse will move the robots of A to their targets in the original problem. The claim follows. □

3.2 Hardness of deciding whether a motion graph is spannable

Deciding whether a given motion graph G is spannable amounts to deciding whether a subset of its vertices $V' \subseteq V \setminus Z$ is spannable. (We must decide this once for $V' = S$ and once for $V' = T$. The other requirements in Definition 2 are trivial to verify, so we assume they hold.) To study the latter, it is convenient to consider the following problem.

Leaf-Constrained Spannability (LCS) problem. Given (H,L), where $H = (V_H,E_H)$ is a connected simple graph and $L \subseteq V_H$ is a subset of its vertices, determine whether H has a spanning forest consisting of full binary trees such that every leaf in the spanning forest is in L. We refer to a "yes" instance of LCS as *spannable*.

Determining whether G is spannable equates to solving an induced set of LCS instances, as formalized by Lemma 3, which follows directly from our definitions.

Lemma 3. *Let $H_1,...,H_q$ be the connected components of $G[V']$. For each H_i, let L_i denote the subset of vertices of H_i that are adjacent to some vertex $v \in Z$ in G. Then V' is spannable if and only if for each i the LCS instance (H_i,L_i) is spannable.*

We call a vertex v a *boundary* vertex if $v \in L$; otherwise, it is an *inner* vertex. The *depth* of a vertex v is the shortest path distance from v to the nearest boundary vertex. The *depth* of an LCS instance is the maximum depth of a vertex in V_H. We prove that for general graphs LCS is NP-complete using a reduction from Hamiltonian Path. Fortunately, this hardness result does not apply to graphs induced by geometric instances, as we demonstrate in the sequel using the concept of depth.

Theorem 4 (Appendix A.3). *LCS is NP-complete even for a depth of 1.*

4 Application in geometric settings

We now apply our positive result for PMGs to geometric and non-discrete instances. We begin with MSR for unit-disk robots operating in the plane. Let M be such an instance with S and T denoting the start and target positions, respectively, where a *position* is specified by the location of a robot's center. Our approach consists of constructing a motion graph $\hat{G} \coloneqq G(M)$ such that a solution for $\hat{G}$ can be translated to a solution for M. We take the vertices of $\hat{G}$ to be $V = S \cup T \cup \{z\}$, i.e., we consider each start/target position as a vertex and use an additional position z as an empty vertex. Hereafter we will typically not distinguish between a position and its corresponding vertex in $\hat{G}$. Our goal is to associate each edge $e = (u,v)$ in $\hat{G}$ with a corresponding path $\pi(e)$ from u to v in M such that a motion of a pebble robot along e always corresponds to a collision-free motion along $\pi(e)$. Therefore, we add edge e to $\hat{G}$ if and only if path $\pi(e)$ exists in M such that a robot moving along $\pi(e)$ will not collide with a robot located at any position $w \in (S \cup T) \setminus \{u,v\}$. We will show how to compute efficiently such edges, which are determined uniquely based on V. Then, we will show that we can efficiently determine whether $\hat{G}$ is spannable by exploiting the properties of its induced LCS instances.

Since our main goal is to show that spannability can be efficiently decided for PMGs based on non-discrete inputs, we do not tighten running times and assume no obstacles, which can be handled efficiently but are omitted for simplicity.

4.1 Preliminaries and background

For a point $p \in \mathbb{R}^2$, let $\mathcal{D}_r(p)$ denote the open disk of radius r centered at point p. Let $\mathcal{E} = \bigcup_{x \in S \cup T} \mathcal{D}_1(x)$ be the union of all unit disks positioned at endpoints. Let $\mathcal{Z} = \{x \in \mathbb{R}^2 : \mathcal{D}_1(x) \cap \mathcal{E} = \varnothing\}$ denote the *endpoint-free space*, i.e., all positions where a unit-disk robot is guaranteed not to intersect another robot located at an endpoint. To ensure conditions (i) and (ii) of Definition 2 we require the following in M: (i) unit disks placed at $S \cup T$ are pairwise interior-disjoint and (ii) $\mathcal{Z}$ is a single connected component. We can easily test these conditions as part of the construction of $\hat{G}$ (to be described) and report if they do not hold.

Arrangements. An arrangement of circles is a subdivision of the plane into vertices (intersection points of the circles), edges (maximal arcs of a circle not intersected by any other circle), and faces (maximal portions of the plane that are not intersected by any circle) induced by the given circles. For ease of exposition, we assume general position, i.e., no more than two circles intersect at any given point and there are no two tangent circles.

Courcelle's theorem. To show that LCS can be efficiently decided for geometric inputs we require a basic understanding of treewidth and Courcelle's theorem [14]. In brief, *treewidth* measures how "tree-like" a graph is using a natural number. Courcelle's theorem states that any graph property that can be expressed in Monadic Second-Order logic (MSO) is decidable in linear time on graphs with a fixed treewidth. See [15] for a more comprehensive background.

4.2 Constructing the motion graph

To construct $\hat{G}$ we subdivide the plane into maximal path-connected regions, where a robot positioned anywhere in a region intersects the same subset of endpoints. This is done by computing the arrangement $\mathcal{A} \coloneqq \mathcal{A}(\mathcal{C})$ where $\mathcal{C}$ is the set of radius 2 circles centered at $S \cup T$; see Figure 8. Notice that the circles of $\mathcal{C}$ that contain a point p correspond to the endpoints that a robot centered at p intersects with. Also, since $\mathcal{Z}$ is a single connected component, $\mathcal{Z}$ corresponds to the single unbounded face of $\mathcal{A}$. Let us fix z as an arbitrary position in $\mathcal{Z}$.

We now construct $\hat{G}$. Since we fix the vertices to be $V = S \uplus T \uplus \{z\}$, the main task is determining which edges to add. For $v \in V$, let $f(v)$ denote the face of $\mathcal{A}$ containing v. We must ensure that the motion of a pebble across an edge (u,v) in $\hat{G}$ corresponds to a collision-free motion of a unit disc between the two positions in M. Hence, for $u,v \in S \cup T$, we will add (u,v) to $\hat{G}$ if and only if a path π exists from u to v that does not intersect any disk $\mathcal{D}_2(w)$ for $w \in (S \cup T) \setminus \{u,v\}$, i.e., a robot can move along π without colliding with other robots located at positions $V \setminus \{u,v\}$. It suffices to consider the case where $\pi \subset \mathcal{D}_2(u) \cup \mathcal{D}_2(v)$.[8] In this case, π passes through exactly 3 faces, which are $f(u)$, $f(v)$, and a face contained in $\mathcal{D}_2(u) \cap \mathcal{D}_2(v)$ but not in any other circle. The other type of edge, (u,z), is added to $\hat{G}$ if and only if the faces $f(u)$ and $f(z)$ are adjacent, i.e., they share a common boundary.

[8] The only other option is when π passes through $\mathcal{Z}$. In this case the edges (u,z) and (v,z) would be present in $\hat{G}$, which would make u and v boundary vertices in any LCS instance induced by $\hat{G}$. Since any spannable LCS instance remains spannable if we remove edges between two boundary vertices, we can safely ignore (u,v).

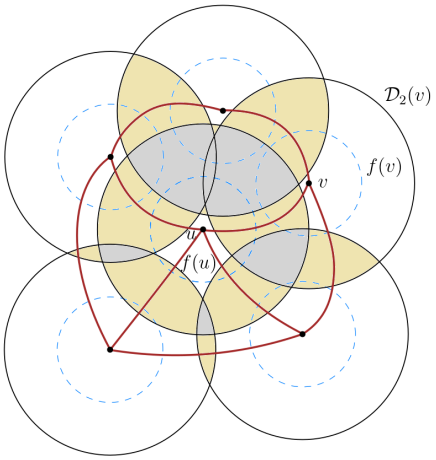

Fig. 8: Partial view of $\mathcal{A}$ showing robots at their endpoints (blue dashed circles), corresponding circles of $\mathcal{C}$ (large black circles), and edges of $\hat{G}$ (thick brown curves) between shown endpoints. Faces of $\mathcal{A}$ contained in 2 (resp. more than 2) disks are shaded in beige (resp. gray).

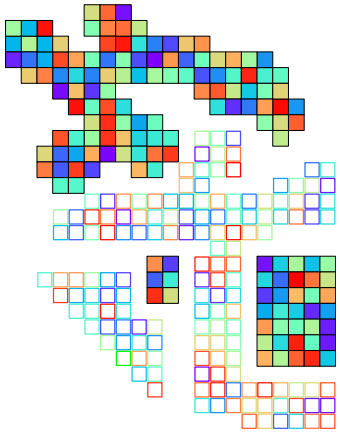

Fig. 9: MSR(□) instance, which is SFFBT, composed of 5 polyominoes satisfying conditions (i)-(iii) in Section 4.4 except for the 5×7 polyomino (right), which violates (ii), but is nevertheless spannable. Filled (resp. unfilled) cells are start (resp. target) positions. The rest of the grid cells (not drawn) are empty.

Following our observations, a straightforward traversal of $\mathcal{A}$ obtains the edges of $\hat{G}$, for which we can also obtain the corresponding paths in M. We can construct $\mathcal{A}$ using a standard sweep-line algorithm in $O(n\log n)$ time, and traverse it in the same time [21]. Notice that $\hat{G}$ is planar and has a complexity of $O(n)$ since the dual graph[9] of $\mathcal{A}$ is planar and $\hat{G}$ can be obtained from the dual graph by deleting vertices and contracting edges.

4.3 Efficiently deciding whether the motion graph is spannable

Let $(\hat{H},L)$ be an LCS instance induced by $\hat{G}$ (cf. Lemma 3). We now show that we can solve $(\hat{H},L)$ in linear time.

Lemma 4. *If $(\hat{H},L)$ is spannable then its depth is at most 9.*

Proof. Let d be the depth of $(\hat{H},L)$ and let $v \in V_{\hat{H}}$ be a vertex with such a depth. Notice that the forest spanning $\hat{H}$ contains a subtree $\mathcal{T}$ rooted at v, which is a perfect binary tree of height d, i.e., $\mathcal{T}$ has $2^{d+1}-1$ nodes. This is true since the depth of a node in the forest differs from the depth of its parent by at most 1 and each tree in the forest is full. Let us now bound the area available for the start/target positions of the unit-disk robots corresponding to the nodes of $\mathcal{T}$. For any $(u,v) \in E_{\hat{H}}$, which is also an edge of $\hat{G}$, the distance between positions u and

[9] In the dual graph of $\mathcal{A}$ each node represents a face, and an edge connects two nodes if their corresponding faces are adjacent in $\mathcal{A}$.

v is at most 4 as $\mathcal{D}_2(u)$ and $\mathcal{D}_2(v)$ must intersect for (u,v) to exist. Consequently, any robot in $\mathcal{T}$ is positioned at most $4d$ units from v. That is, all robots in $\mathcal{T}$ must be located inside $\mathcal{D}_{4d}(v)$. The total area of $2^{d+1}-1$ unit disks exceeds the area of $\mathcal{D}_{4d}(v)$ for $d>9$, which concludes the argument. □

By Lemma 4 we easily rule out $(\hat{H},L)$ as not spannable if its depth is greater than 9 using a breadth-first search from L. Otherwise, $\hat{H}$ has a constant depth, which, together with its planarity, means that $\hat{H}$ has a constant treewidth [7]. We therefore apply Courcelle's theorem in the latter case to show that we can efficiently decide spannability. To do so, it suffices to express spannability as an MSO formula that is true if and only if $(\hat{H},L)$ is spannable. We may construct such a formula having a constant size using standard predicates.

Lemma 5 (Appendix A.3). *Let (H,L) be an LCS instance where H is planar and has a constant depth. Then we can solve (H,L) in linear time.*

Putting everything together, we obtain the following.

Theorem 5 (Appendix A.3). *Let M be an MSR instance for n unit disks in the plane. We can construct the motion graph $\hat{G}$ for M, decide whether $\hat{G}$ is spannable, and if so output a solution for M, in $O(n^2)$ time.*

4.4 A restricted case that always has a monotone solution

We now identify a restricted variant of MSR(□) that always has a solution. Let G be a PMG of an MSR(□) instance in which Z is connected. We require the following for each connected component P of S or T in G, which is commonly called a *polyomino*:[10] (i) P is simple, i.e., it has no holes, (ii) each square in P is either a boundary square (i.e., a square with an edge on the perimeter) or is adjacent to a boundary square, and (iii) each boundary square of P is adjacent to a square of Z. We say that such a PMG G *is composed of thin polyominoes*; see Figure 9 for an example.

Theorem 6 (Appendix A.3). *A PMG G that is composed of thin polyominoes is spannable and thus solvable.*

We prove Theorem 6 through a careful analysis showing that each polyomino P contains a *good square*, which is an inner square s that has two adjacent boundary squares in P whose sole adjacent inner square is s; see Figure 12(a) in Appendix A.3. We then recursively get an appropriate spanning tree for P by removing such a square.

We conclude by observing that spannable MSR(□) instances give rise to various shapes; see Figure 9. In particular, the family gives rise to configurations where a robot is tightly surrounded by one or two "layers" of other robots, such as a 7×5 polyomino or a $4\times k$ polyomino, for all k.

[10] We do not distinguish between a vertex of G and its corresponding grid square.

5 Conclusion

Through a detailed investigation of MSR we identify a new structural assumption, SFFBT, guaranteeing a solution, which we show can be efficiently decided on geometric MSR inputs. SFFBT relaxes existing separation assumptions guaranteeing a solution in complete efficient MRMP algorithms. We thereby echo calls for establishing such milder conditions [3,5] and give insights into the challenging complexity landscape of MRMP in denser scenarios. Roughly speaking, we allow a start/target position to be closely surrounded by other such positions, provided that it is not too deeply packed among them. SFFBT also generalizes the notion of well-formed environments [11], where each endpoint can be seen as a singleton tree in Definition 2.

We remark that a solution to MSR can apply to settings where parallel motion is allowed (e.g., MRMP). One could reduce the reconfiguration time of a monotone solution by repeatedly replanning the trajectory of small subsets of the robots while treating the rest as moving obstacles, akin to prioritized planning [11].

Additional results. Our negative and positive results apply to two related problems for which MSR is a special case. On the negative side, we establish the NP-hardness of optimally decoupling an MRMP instance into sequential plans [38] (we recall its definition in Appendix A.4) and strengthen the NP-hardness of Reconfiguration with Minimum Number of Moves [20]. Specifically, while the proof of [20] relies heavily on obstacles, we present hardness without obstacles. Also, it may be verified that all of our reductions have a linear blow-up. Consequently, they establish a $2^{\Omega(n)}$ running time lower bound, for n objects, conditioned on the Exponential Time Hypothesis (ETH) [29], which is a first for the latter problem (the blow-up in [20] is at least quadratic). In this regard, our running time lower bound matches the upper bound, up to a polynomial factor, of the dynamic programming-based MSR algorithm in [39], establishing its optimality. On the positive side, our results also apply to the two related problems since a monotone solution is necessarily optimal for both problems, as is readily verifiable.

Future work and open problems. First, our result for efficiently deciding whether unit disc MSR instances are spannable can be similarly applied to other uniform shapes (e.g., unit squares). We believe that SFFBT can be generalized in a few other ways. One could allow multiple components of Z, whereby a robot would traverse many trees to reach its target. Another generalization is trees containing both start and target positions.

A more direct efficient LCS algorithm for geometric inputs would be more practical than applying Courcelle's theorem. One alternative is a standard dynamic programming approach for bounded treewidth graphs [15]. Better yet, can we characterize spannable graphs such as, e.g., those arising in MSR($\square$)? A necessary condition is that there are fewer inner than boundary squares, however it is not sufficient.

For the cases where we show hardness for MSR($\square$,$\square$), what is the complexity for MSR($\square$)? Specifically for the latter, consider n^2 robots where each of the start and target configurations lies on a $n \times n$ portion of the 2D grid, and the configurations are separated by an empty column (generalizing Figure 6). Are such instances always solvable?

References

1. Manuel Abellanas, Sergey Bereg, Ferran Hurtado, Alfredo García Olaverri, David Rappaport, and Javier Tejel. Moving coins. *Comput. Geom.*, 34(1):35–48, 2006.
2. Mikkel Abrahamsen, Tzvika Geft, Dan Halperin, and Barak Ugav. Coordination of Multiple Robots along Given Paths with Bounded Junction Complexity. In *AAMAS*, pages 932–940. ACM, 2023.
3. Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Trans Autom. Sci. Eng.*, 12(4):1309–1317, 2015.
4. Pankaj K. Agarwal, Tzvika Geft, Dan Halperin, and Erin Taylor. Multi-robot motion planning for unit discs with revolving areas. *Comput. Geom.*, 114:102019, 2023.
5. Bahareh Banyassady, Mark de Berg, Karl Bringmann, Kevin Buchin, Henning Fernau, Dan Halperin, Irina Kostitsyna, Yoshio Okamoto, and Stijn Slot. Unlabeled multi-robot motion planning with tighter separation bounds. In *SoCG*, volume 224 of *LIPIcs*, pages 12:1–12:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
6. Sergey Bereg, Adrian Dumitrescu, and János Pach. Sliding disks in the plane. *Int. J. Comput. Geom. Appl.*, 18(5):373–387, 2008.
7. Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998.
8. Thomas Brocken, G. Wessel van der Heijden, Irina Kostitsyna, Lloyd E. Lo-Wong, and Remco J. A. Surtel. Multi-robot motion planning of k-colored discs is PSPACE-hard. In *FUN*, volume 157 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
9. Stephen J. Buckley. Fast motion planning for multiple moving robots. In *ICRA*, pages 322–326. IEEE Computer Society, 1989.
10. Gruia Călinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. *SIAM J. Discret. Math.*, 22(1):124–138, 2008.
11. Michal Cáp, Peter Novák, Alexander Kleiner, and Martin Selecký. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE Trans Autom. Sci. Eng.*, 12(3):835–849, 2015.
12. Rupesh Chinta, Shuai D. Han, and Jingjin Yu. Coordinating the motion of labeled discs with optimality guarantees under extreme density. In *WAFR*, volume 14 of *Springer Proceedings in Advanced Robotics*, pages 817–834. Springer, 2018.
13. Alexandre Cooper, Stephanie Maaz, Amer E. Mouawad, and Naomi Nishimura. Parameterized complexity of reconfiguration of atoms. In *WALCOM*, volume 13174 of *Lecture Notes in Computer Science*, pages 263–274. Springer, 2022.
14. Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.
15. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer, 2015.
16. Andreas Darmann and Janosch Döcker. On simplified NP-complete variants of monotone 3-SAT. *Discret. Appl. Math.*, 292:45–58, 2021.
17. Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications, 3rd Edition*. Springer, 2008. URL: https://www.worldcat.org/oclc/227584184.
18. Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.*, 48(6):1727–1762, 2019.

19. Adrian Dumitrescu. Mover problems. In János Pach, editor, *Thirty Essays in Geometric Graph Theory*, pages 185–211. Springer, Berlin-Heidelberg, 2013.
20. Adrian Dumitrescu and Minghui Jiang. On reconfiguration of disks in the plane and related problems. *Comput. Geom.*, 46(3):191–202, 2013.
21. Efi Fogel, Dan Halperin, and Ron Wein. *CGAL Arrangements and Their Applications - A Step-by-Step Guide*, volume 7 of *Geometry and Computing*. Springer, 2012. doi:10.1007/978-3-642-17283-0.
22. Kai Gao, Si Wei Feng, Baichuan Huang, and Jingjin Yu. Minimizing running buffers for tabletop object rearrangement: Complexity, fast algorithms, and applications. *Int. J. Robotics Res.*, 42(10):755–776, 2023.
23. Kai Gao, Darren Lau, Baichuan Huang, Kostas E. Bekris, and Jingjin Yu. Fast high-quality tabletop rearrangement in bounded workspace. In *ICRA*, pages 1961–1967. IEEE, 2022.
24. Kai Gao and Jingjin Yu. On the utility of buffers in pick-n-swap based lattice rearrangement. In *ICRA*, pages 5786–5792. IEEE, 2023.
25. Tzvika Geft and Dan Halperin. Refined hardness of distance-optimal multi-agent path finding. In *AAMAS*, pages 481–488. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022.
26. Dan Halperin, Marc J. van Kreveld, Golan Miglioli-Levy, and Micha Sharir. Space-aware reconfiguration. *Discret. Comput. Geom.*, 69(4):1157–1194, 2023.
27. Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theor. Comput. Sci.*, 343(1-2):72–96, 2005.
28. John E. Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". *The International Journal of Robotics Research*, 3(4):76–88, 1984.
29. Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.
30. Daniel Kornhauser, Gary L. Miller, and Paul G. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *FOCS*, pages 241–250. IEEE Computer Society, 1984.
31. Oren Salzman and Roni Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *AAMAS*, pages 1711–1715. International Foundation for Autonomous Agents and Multiagent Systems, 2020.
32. Israela Solomon and Dan Halperin. Motion planning for multiple unit-ball robots in $\mathbb{R}^d$. In *Workshop on the Algorithmic Foundations of Robotics, WAFR*, pages 799–816, 2018.
33. Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *Int. J. Robotics Res.*, 35(14):1750–1759, 2016.
34. Kiril Solovey, Jingjin Yu, Or Zamir, and Dan Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems*, 2015.
35. Paul G. Spirakis and Chee-Keng Yap. Strong NP-hardness of moving many discs. *Inf. Process. Lett.*, 19(1):55–59, 1984.
36. Sarah Y. Tang and Vijay Kumar. A complete algorithm for generating safe trajectories for multi-robot teams. In *ISRR (2)*, volume 3 of *Springer Proceedings in Advanced Robotics*, pages 599–616. Springer, 2015.
37. Craig A. Tovey. A simplified NP-complete satisfiability problem. *Discret. Appl. Math.*, 8(1):85–89, 1984.

38. Jur van den Berg, Jack Snoeyink, Ming C. Lin, and Dinesh Manocha. Centralized path planning for multiple robots: Optimal decoupling into sequential plans. In *Robotics: Science and Systems*. The MIT Press, 2009.
39. Rui Wang, Kai Gao, Daniel Nakhimovich, Jingjin Yu, and Kostas E. Bekris. Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search. In *ICRA*, pages 6621–6627. IEEE, 2021.