

Coordinated Motion Planning: Reconfiguring a Swarm of Labeled Robots with Bounded Stretch*

Erik D. Demaine

MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA, USA
edemaine@mit.edu

Sándor P. Fekete

Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
s.fekete@tu-bs.de

 <https://orcid.org/0000-0002-9062-4241>

Phillip Keldenich¹

Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
p.keldenich@tu-bs.de

 <https://orcid.org/0000-0002-6677-5090>

Christian Scheffer

Department of Computer Science, TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
c.scheffer@tu-bs.de

 <https://orcid.org/0000-0002-3471-2706>

Henk Meijer

Science Department, University College Roosevelt Middelburg,
Middelburg, The Netherlands
h.meijer@ucr.nl

Abstract

We present a number of breakthroughs for coordinated motion planning, in which the objective is to reconfigure a swarm of labeled convex objects by a combination of parallel, continuous, collision-free translations into a given target arrangement. Problems of this type can be traced back to the classic work of Schwartz and Sharir (1983), who gave a method for deciding the existence of a coordinated motion for a set of disks between obstacles; their approach is polynomial in the complexity of the obstacles, but exponential in the number of disks. Despite a broad range of other non-trivial results for multi-object motion planning, previous work has largely focused on *sequential* schedules, in which one robot moves at a time, with objectives such as the number of moves; attempts to minimize the overall makespan of a coordinated *parallel* motion schedule (with many robots moving simultaneously) have defied all attempts at establishing the complexity in the absence of obstacles, as well as the existence of efficient approximation methods.

We resolve these open problems by developing a framework that provides constant-factor approximation algorithms for minimizing the execution time of a coordinated, *parallel* motion plan for a swarm of robots in the absence of obstacles, provided their arrangement entails some amount of separability. In fact, our algorithm achieves *constant stretch factor*: If all robots want to move at most d units from their respective starting positions, then the total duration of the overall schedule (and hence the distance traveled by each robot) is $O(d)$. Various extensions

* This work was partially supported by the DFG Research Unit *Controlling Concurrent Change*, funding number FOR 1800, project FE407/17-2, Conflict Resolution and Optimization.

¹ Supported by the German Research Foundation under Grant No. FE 407/17-2.



© Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer,
and Christian Scheffer;

licensed under Creative Commons License CC-BY

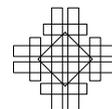
34th Symposium on Computational Geometry (SoCG 2018).

Editors: Bettina Speckmann and Csaba Tóth; Article No. 29; pp. 29:1–29:15

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



include unlabeled robots and different classes of robots. We also resolve the complexity of finding a reconfiguration plan with minimal execution time by proving that this is NP-hard, even for a grid arrangement without any stationary obstacles. On the other hand, we show that for densely packed disks that cannot be well separated, a stretch factor $\Omega(N^{1/4})$ may be required. On the positive side, we establish a stretch factor of $O(N^{1/2})$ even in this case. The intricate difficulties of computing precise optimal solutions are demonstrated by the seemingly simple case of just two disks, which is shown to be excruciatingly difficult to solve to optimality.

2012 ACM Subject Classification Theory of computation \rightarrow Computational geometry, Theory of computation \rightarrow Problems, reductions and completeness, Computer systems organization \rightarrow Robotic control

Keywords and phrases Robot swarms, coordinated motion planning, parallel motion, makespan, bounded stretch, complexity, approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2018.29

Related Version A full version of this paper can be found at <https://arxiv.org/abs/1801.01689> [6]. There is a video motivating, visualizing and demonstrating the concepts of this paper, reachable via <http://computational-geometry.org/SoCG-videos/socg18video/videos/74>, with an accompanying abstract at [4] in these proceedings, <http://dx.doi.org/10.4230/LIPIcs.SoCG.2018.74>.

Acknowledgements We thank anonymous reviewers of a preliminary version of the paper for helping to improve the overall presentation.

1 Introduction

Since the beginning of computational geometry, robot motion planning has been at the focus of algorithmic research. Planning the relocation of a geometric object among geometric obstacles leads to intricate scientific challenges, requiring the combination of deep geometric and mathematical insights with algorithmic techniques. With the broad and ongoing progress in robotics, the increasing importance of intelligent global planning with performance guarantees requires more sophisticated algorithmic reasoning, in particular when it comes to the higher-level task of coordinating the motion of many robots.

From the early days, multi-robot coordination has received attention from the algorithmic side. Even in the groundbreaking work by Schwartz and Sharir [20] from the 1980s, one of the challenges was coordinating the motion of *several* disk-shaped objects among obstacles. Their algorithms run in time polynomial in the complexity of the obstacles, but exponential in the number of disks. This illustrates the significant challenge of coordinating many individual robots. In addition, a growing number of applications focus primarily on robot interaction in the absence of obstacles, such as air traffic control or swarm robotics, where the goal is overall efficiency, rather than individual navigation.

With the challenges of multi-robot coordination being well known, there is still a huge demand for positive results with provable performance guarantees. In this paper, we provide significant progress in this direction, with a broad spectrum of results.

1.1 Our results

- For the problem of minimizing the total time for reconfiguring a system of labeled circular robots in a grid environment, we show that it is strongly NP-complete to compute an

optimal solution; see Theorem 1.

- We give an $\mathcal{O}(1)$ -approximation for the long-standing open problems of parallel motion-planning with minimum makespan in a grid setting. This result is based on establishing an *absolute* performance guarantee: We prove that for any labeled arrangement of robots, there is always an overall schedule that gets each robot to its target destination with bounded *stretch*, i.e., within a constant factor of the largest individual distance. See Theorem 3 for the base case of grid-based configurations, which is extended later on.
- For our approach, we make use of a technique to separate planar (cyclic) flows into so-called *subflows* whose thickness can be controlled by the number of subflows, see Definition 7 and Lemma 8. This is of independent interest for the area of packet routing with bounded memory: Our Theorem 4 implies that $\mathcal{O}(D)$ steps are sufficient to route any permutation of dilation D on the grid, even with a buffer size of 1, resolving an open question by Scheideler [19] dating back to 1998.
- We extend our approach to establish constant stretch for the generalization of *colored* robot classes, for which unlabeled robots are another special case; see Theorem 13.
- We extend our results to the scenario with continuous motion and arbitrary coordinates, provided the distance between a robot's start and target positions is at least one diameter; see Theorem 15. This implies that efficient multi-robot coordination is always possible under relatively mild separability conditions; this includes non-convex robots.
- For the continuous case of N unit disks and weaker separability, we establish a lower bound of $\Omega(N^{1/4})$ and an upper bound of $\mathcal{O}(\sqrt{N})$ on the achievable stretch, see Theorem 14 and Theorem 15.

We also highlight the geometric difficulty of computing optimal trajectories even in seemingly simple cases; due to limited space, this can be found in the full version of the paper [6].

1.2 Related work

Multi-object motion planning problems have received a tremendous amount of attention from a wide spectrum of areas. Due to limited space, we focus on algorithmic work with an emphasis on geometry; see the full version of the paper [6] for a more comprehensive overview.

In the presence of obstacles, Aronov et al. [3] demonstrate that for up to three robots, a path can be constructed efficiently, if one exists. Ramanathan and Alagar [17] and Schwartz and Sharir [20] consider the case of several disk-shaped objects between polygonal obstacles. Both give algorithms for deciding reachability of a given target configuration. The algorithms run in time polynomial in the complexity of the obstacles, but exponential in the number of disks. Hopcroft et al. [11] and Hopcroft and Wilfong [12] prove that it is PSPACE-complete to decide reachability of a given target configuration, even when restricted to rectangular objects in a rectangular region. This was later strengthened by Hearn and Demaine [9, 10] to rectangles of size 1×2 and 2×1 . Moreover, this problem is similar to the well-known *Rush-Hour Problem*, which was shown to be PSPACE-complete by Flake and Baum [8]. For moving disks, Spirakis and Yap [24] prove strong NP-hardness of the same problem for disks of varying size. Bereg et al. [5] and Abellanas et al. [1] consider minimizing the *number* of moves of a set of disks into a target arrangement without obstacles. These bounds were later improved by Dumitrescu and Jiang [7], who prove that the problem remains NP-hard for congruent disks even when the motion is restricted to sliding. Yu [27] provides an expected constant-factor approximation for the optimal makespan in the grid case.

On the practical side, there is a wide range of approaches for solving multi-object motion planning problems, both optimally and heuristically; for instances of limited size, SAT solvers and IP-based methods are used for discretized versions, while for larger instances, previous work resorts to heuristic solutions. Refer to the full version of the paper [6] for an overview.

In both discrete and geometric variants of the problem, the objects can be *labeled*, *colored* or *unlabeled*. In the *colored* case, the objects are partitioned into k groups and each target position can only be covered by an object with the right color. This case was recently considered by Solovey and Halperin [21], who present and evaluate a practical sampling-based algorithm. In the *unlabeled* case, the objects are indistinguishable and each target position can be covered by any object. This scenario was first considered by Kloder and Hutchinson [13], who presented a practical sampling-based algorithm. Turpin et al. [25] prove that it is possible to find a solution in polynomial time, if one exists. This solution is optimal with respect to the longest distance traveled by any one robot. However, their results only hold for disk-shaped robots under additional restrictive assumptions on the free space. For unit disks and simple polygons, Adler et al. [2] provide a polynomial-time algorithm under the additional assumption that the start and target positions have some minimal distance from each other. Under similar separability assumptions, Solovey et al. [22] provide a polynomial-time algorithm that produces a set of paths that is no longer than $\text{OPT} + 4N$, where N is the number of robots. However, they do not consider the makespan, but only the total path length. On the negative side, Solovey and Halperin [23] prove that the unlabeled multiple-object motion planning problem is PSPACE-hard, even when restricted to unit square objects in a polygonal environment.

The problem of finding constructive algorithmic solutions for the problem of coordinated, parallel motion planning in the absence of obstacles (with the objective of minimizing the makespan of the overall schedule) was explicitly posed as a long-standing open problem by Overmars [16] at the 2006 Dagstuhl meeting on Robot Navigation, but can be traced back much further. It is also related to open problems from the field of routing, where it is well-known that for any given family of simple paths one can find a way to route packets along the paths such that the total number of steps required is $\mathcal{O}(C + D)$, where C is the congestion and D is the dilation of the given family of paths. However, algorithms in this context typically require that each node can store a constant number of packets. Scheideler [19] raises the question whether routing in $\mathcal{O}(C + D)$ steps is still possible if only one packet can be stored at each node. We answer a variant of this question in the case of Grid Permutation Routing. By our Theorem 4, $\mathcal{O}(D)$ steps are sufficient to route any permutation of dilation D on the grid, even with a buffer size of 1.

On the other hand, on grid graphs, the problem resembles the generalization of the 15-puzzle, for which Wilson [26] and Kornhauser et al. [14] give an efficient algorithm that decides reachability of a target configuration and provide both lower and upper bounds on the number of moves required. Ratner and Warmuth [18] prove finding a shortest solution for this puzzle remains NP-hard.

During the review period of our work, Yu [28] has independently proposed a similar approach that also achieves a constant-factor approximation in the case of a rectangular grid.

2 Preliminaries

In the grid setting of Section 3 we consider an $n_1 \times n_2$ -grid $G = (V, E)$, which is dual to an $n_1 \times n_2$ -rectangle P in which the considered robots are arranged. A *configuration* of P is a mapping $C : V \rightarrow \{1, \dots, N, \perp\}$, which is injective w.r.t. the labels $\{1, \dots, N\}$ of the

$N \leq |P|$ robots to be moved, where \perp denotes the empty square. The inverse image of a robot's label ℓ is $C^{-1}(\ell)$. In the following, we consider a *start configuration* C_s and *target configuration* C_t ; for $i \in \{1, \dots, N\}$, we call $C_s^{-1}(i)$ and $C_t^{-1}(i)$ the *start* and *target position* of the robot i . Given the (minimum) Manhattan distance between each robot's start/target positions for each robot, we denote by d the maximum such distance over all robots.

A configuration $C_1 : V \rightarrow \{1, \dots, N, \perp\}$ can be *transformed within one single transformation step* into another configuration $C_2 : V \rightarrow \{1, \dots, N, \perp\}$, denoted $C_1 \rightarrow C_2$, if $C_1^{-1}(\ell) = C_2^{-1}(\ell)$ or $(C_1^{-1}(\ell), C_2^{-1}(\ell)) \in E$ holds for all $\ell \in \{1, \dots, N\}$, i.e., if each robot does not move or moves to one of the at most four adjacent squares. Furthermore, two robots cannot exchange their squares in one transformation step, i.e., for all occupied squares $v \neq w \in V$, we require that $C_2(v) = C_1(w)$ implies $C_2(w) \neq C_1(v)$. For $M \in \mathbb{N}$, a *schedule* is a sequence $C_1 \rightarrow \dots \rightarrow C_M$ of transformations. The number of steps in a schedule is called its *makespan*. Given a start configuration C_s and a target configuration C_t , the *optimal makespan* is the minimum number of steps in a schedule starting with C_s and ending with C_t . Let $n > 1$. Note that for the 2×2 -, $1 \times n$ - and $n \times 1$ -rectangles, there are pairs of start and target configurations where no such sequence exists. For all other rectangles, such configurations do not exist; we provide an $\mathcal{O}(1)$ -approximation of the makespan in Section 3.

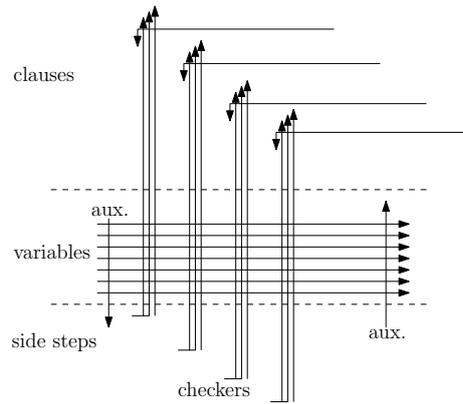
For the continuous setting of Section 5, we consider N robots $R := \{1, \dots, N\} \subseteq \mathbb{N}$. The Euclidean distance between two points $p, q \in \mathbb{R}^2$ is $|pq| := \|p - q\|_2$. Every robot r has a *start* and *target position* $s_r, t_r \in \mathbb{R}^2$ with $|s_i s_j|, |t_i t_j| \geq 2$ for all $i \neq j$. In the following, $d := \max_{r \in R} |s_r t_r|$ is the maximum distance a robot has to cover. A *trajectory* of a robot r is a curve $m_r : [0, T_r] \rightarrow \mathbb{R}^2$, where $T_r \in \mathbb{R}^+$ denotes the *travel time* of r . This curve m_r does not have to be totally differentiable, but must be totally left- and right-differentiable. Intuitively, at any point in time, a robot has a unique *past* and *future direction* that are not necessarily identical. This allows the robot to make sharp turns, but does not allow jumps. We bound the speed of the robot by 1, i.e., for each point in time, both left and right derivative of m_r have Euclidean length at most 1. Let $m_i : [0, T_i] \rightarrow \mathbb{R}^2$ and $m_j : [0, T_j] \rightarrow \mathbb{R}^2$ be two trajectories; w.l.o.g., all travel times are equal to the maximum travel time T_{\max} by extending m_r with $m_r(t) = m_r(T_r)$ for all $T_r < t \leq T_{\max}$. The trajectories m_i and m_j are *compatible* if the corresponding robots do not intersect at any time, i.e., if $|m_i(t) m_j(t)| \geq 2$ holds for all $t \in [0, T_i]$. A *trajectory set* of R is a set of compatible trajectories $\{m_1, \dots, m_N\}$, one for each robot. The (*continuous*) *makespan* of a trajectory set $\{m_1, \dots, m_N\}$ is defined as $\max_{r \in R} T_r$. A trajectory set $\{m_1, \dots, m_N\}$ *realizes* a pair of start and target configurations $\mathcal{S} := (\{s_1, \dots, s_N\}, \{t_1, \dots, t_N\})$ if $m_r(0) = s_r$ and $m_r(T_r) = t_r$ hold for all $r \in R$. We are searching for a trajectory set $\{m_1, \dots, m_N\}$ realizing \mathcal{S} with minimal makespan.

3 Labeled grid permutation

Let $n_1 \geq n_2 \geq 2$, $n_1 \geq 3$ and let P be an $n_1 \times n_2$ -rectangle. In this section, we show that computing the optimal makespan of arbitrarily chosen start and target configurations C_s and C_t of k robots in P is strongly NP-complete. This is followed by a $\mathcal{O}(1)$ -approximation for the makespan.

► **Theorem 1.** *The minimum makespan parallel motion planning problem on a grid is strongly NP-hard.*

We prove hardness using a reduction from MONOTONE 3-SAT. Intuitively speaking, given a formula, we construct a parallel motion planning instance with a *variable robot* for each variable in the formula. To encode a truth assignment, each variable robot is forced to move



■ **Figure 1** A sketch of the parallel motion planning instance resulting from the reduction.

on one of two paths. This is done by employing two groups of auxiliary robots that have to move towards their goal in a straight line in order to realize the given makespan. These auxiliary robots form *moving obstacles* whose position is known at any point in time.

The variable robots cross paths with *checker robots*, one for each literal of the formula, forcing the checker to wait for one time step if the assignment does not satisfy the literal. The checker robots then cross paths with *clause robots*; each clause robot has to move to its goal without delay and can only do so if at least one of the checkers did not wait. In order to ensure that the checkers meet with the clauses at the right time, further auxiliary robots force the checkers to perform a sequence of side steps in the beginning. Figure 1 gives a rough overview of the construction; full details of the proof are given in the full version of the paper [6].

In the proof of NP-completeness, we use a pair of start and target configurations in which the corresponding grids are not fully occupied. However, for our constant-factor approximation, we assume in Theorem 3 that the grid is fully occupied. This assumption is without loss of generality; our approximation algorithm works for any grid population, see Theorem 4.

Our constant-factor approximation is based on an algorithm that computes a schedule with a makespan upper-bounded by $\mathcal{O}(n_1 + n_2)$ described by Lemma 2. Based on Lemma 2, we give a constant factor approximation of the makespan, see Theorem 3. Finally, we embed the algorithm of Theorem 3 into a more general approach to ensure simultaneously a polynomial running time w.r.t. the number N of input robots and a constant approximation factor, see Theorem 4.

► **Lemma 2.** *For a pair of start and target configurations C_s and C_t of an $n_1 \times n_2$ -rectangle, we can compute in polynomial time w.r.t. n_1 and n_2 a sequence of $\mathcal{O}(n_1 + n_2)$ steps transforming C_s into C_t .*

The high-level idea of the algorithm of Lemma 2 is the following. We apply a sorting algorithm called ROTATESORT [15] that computes a corresponding permutation of an $n_1 \times n_2$ (orthogonal) grid within $\mathcal{O}(n_1 + n_2)$ *parallel steps*. Each parallel step is made up of a set of pairwise disjoint *swaps*, each of which causes two neighbouring robots to exchange their positions. Because in our model direct swaps are not allowed, we simulate one parallel step by a sequence of $\mathcal{O}(1)$ transformation steps. This still results in a sequence of $\mathcal{O}(n_1 + n_2)$ transformation steps. A detailed description of the algorithm used in the proof of Lemma 2

is given in the full version of the paper [6]. An alternative to our Lemma 2 was recently proposed by Yu [27], who uses a routine called SPLITANDGROUP for achieving a makespan that is linear in the diameter of the rectangular environment.

Based on the algorithm of Lemma 2, we can give a constant-factor approximation algorithm.

► **Theorem 3.** *There is an algorithm with running time $\mathcal{O}(dn_1n_2)$ that, given an arbitrary pair of start and target configurations of an $n_1 \times n_2$ -rectangle with maximum distance d between any start and target position, computes a schedule of makespan $\mathcal{O}(d)$, i.e., an approximation algorithm with constant stretch.*

For the algorithm of Theorem 3, Lemma 2 is repeatedly applied to rectangles of side length $\mathcal{O}(d)$, resulting in $\mathcal{O}(d)$ transformation steps in total. Because d is a lower bound on the makespan, this yields an $\mathcal{O}(1)$ -approximation of the makespan.

At a high level, the algorithm of Theorem 3 first computes the maximal Manhattan distance d between a robot's start and target position. Then we partition P into a set T of pairwise disjoint rectangular *tiles*, where each tile $t \in T$ is an $n'_1 \times n'_2$ -rectangle for $n'_1, n'_2 \leq 24d$. We then use an algorithm based on flows to compute a sequence of $\mathcal{O}(d)$ transformation steps, ensuring that all robots are in their target tile. Once all robots are in the correct tile, we use Lemma 2 simultaneously on all tiles to move each robot to the correct position within its target tile. The details of the algorithm of Theorem 3 are given further down in this section.

The above mentioned tiling construction ensures that each square of P belongs to one unambiguously defined tile and each robot has a *start* and *target tile*.

Based on the approach of Theorem 3 we give a $\mathcal{O}(1)$ -approximation algorithm for the makespan with a running time polynomial w.r.t. the number N of robots to be moved.

► **Theorem 4.** *There is an algorithm with running time $\mathcal{O}(N^5)$ that, given an arbitrary pair of start and target configurations of a rectangle P with N robots to be moved and maximum distance d between any start and target position, computes a schedule of makespan $\mathcal{O}(d)$, i.e., an approximation algorithm with constant stretch.*

Intuitively speaking, the approach of Theorem 4 distinguishes two cases.

(1) Both $\lfloor \frac{n_2}{4} \rfloor$ and the maximum distance d between the robots' start and target positions, are lower-bounded by the number N of input robots.

(2) $N > \lfloor \frac{n_2}{4} \rfloor$ or $N > d$.

In case (1), the grid is populated sparsely enough such that the robots' trajectories in northern, eastern, southern, and western direction can be done sequentially by four individual transformation sequences.

In order to ensure that each robot has locally enough space, we consider a preprocessed start configuration C_o in which the robots have odd coordinates. We ensure that C_s can be transformed into C_o within $\mathcal{O}(d)$ steps. Analogously, we ensure that the outcome of the northern, eastern, southern, and western trajectories is a configuration C_e with even coordinates, such that C_e can be transformed into C_t within $\mathcal{O}(d)$ transformation steps. The choice of the divisor 4 in the criteria " $N \leq \lceil \frac{n_1}{4} \rceil$ " has the following technical reasoning: In the first case of the proof of Theorem 4, we assume w.l.o.g. that n_1 and n_2 are even. If this is not the case, we move all robots from the last line into the second-to-last line and from the last column into the second-to-last column. This may double the largest x -coordinate of a robot and the largest y -coordinate of a robot, e.g., in the case that all positions in the last line and all positions in that last column are occupied by robots. In a next step,

we transform the start configuration into a configuration C_o in which all robots' x - and y -coordinates are odd. This may cause another doubling of the largest x -coordinate and the largest y -coordinate which may result fourfold increase of the largest x -coordinate and a fourfold increase of the largest y -coordinate. The assumption $N \leq \lceil \frac{n_2}{4} \rceil$ ensures that both dimensions of the rectangular environment are large enough because $n_1 \geq n_2$.

In the second case, we apply the approach of Theorem 3 as a subroutine to a union of smallest rectangles that contain the robots' start and target configurations.

The full detailed version of the proof of Theorem 4 can be found in the full version of the paper [6].

In the rest of Section 3, we give the proof of Theorem 3, i.e. we give an algorithm that computes a schedule with makespan linear in the maximum distance between robots' start and target positions. The remainder of the proof of Theorem 3 is structured as follows. In Section 3.1 we give an outline of our *flow algorithm* that ensures that each robot reaches its target tile in $\mathcal{O}(d)$ transformation steps. Section 3.2 gives the full intuition of this algorithm and its subroutines. (For full details, we refer to the full version of the paper [6]).

3.1 Outline of the approximation algorithm of Theorem 3

We model the trajectories of robots between tiles as a flow f_T , using the weighted directed graph $G_T = (T, E_T, f_T)$, which is dual to the tiling T defined in the previous section. In G_T , we have an edge $(v, w) \in E_T$ if there is at least one robot that has to move from v into w . Furthermore, we define the weight $f_T((v, w))$ of an edge as the integer number of robots that move from v to w . As P is fully occupied, f_T is a *circulation*, i.e., a flow with no sources or sinks, in which flow conservation has to hold at all vertices. Because the side lengths of the tiles are greater than d , G_T is a grid graph with additional diagonal edges and thus has degree at most 8.

The maximum edge value of f_T is $\Theta(d^2)$, but only $\mathcal{O}(d)$ robots can possibly leave a tile within a single transformation step. Therefore, we decompose the flow f_T of robots into a *partition* consisting of $\mathcal{O}(d)$ *subflows*, where each individual robot's motion is modeled by exactly one subflow and each edge in the subflow has value at most d . Thus we are able to *realize* each subflow in a single transformation step by placing the corresponding robots adjacent to the boundaries of its corresponding tiles before we realize the subflow. To facilitate the decomposition into subflows, we first preprocess G_T . In total, the algorithm consists of the following subroutines, elaborated in detail in Section 3.2.

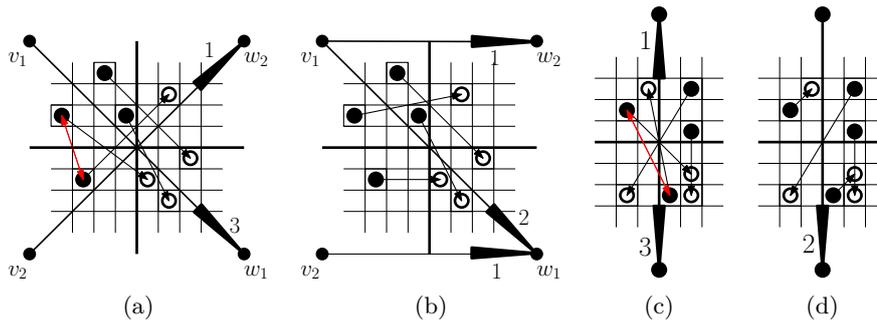
- **Step 1:** Compute d , the tiling T and the corresponding flow G_T .
- **Step 2:** Preprocess G_T in order to remove intersecting and bidirectional edges.
- **Step 3:** Compute a partition into $\mathcal{O}(d)$ d -subflows.
- **Step 4:** Realize the $\mathcal{O}(d)$ subflows using $\mathcal{O}(d)$ transformation steps.
- **Step 5:** Simultaneously apply Lemma 2 to all tiles, moving each robot to its target position.

3.2 Details of the approximation algorithm of Theorem 3

In this section we only give more detailed descriptions of Steps 1-4 because Step 5 is a trivial application of Lemma 2 to all tiles in parallel.

3.2.1 Step 1: Compute d , the tiling T , and the corresponding flow G_T

The maximal distance between robots' start and target positions can be computed in a straightforward manner.



■ **Figure 2** Illustration of the preprocessing (step (1): before and after removing crossing edges (a)+(b) and step (2): before and after removing bidirectional edges (c)+(d)). The red arrows indicate how robots change their positions during the preprocessing steps.

For the tiling, we assume that the rectangle P is axis aligned and that its bottom-left corner is $(0, 0)$. We consider $k_v := \lfloor \frac{n_1}{12d} \rfloor$ vertical lines $\ell_1^v, \dots, \ell_{k_v}^v$ with x -coordinate modulo $12d$ equal to 0. Analogously, we consider $k_h := \lfloor \frac{n_2}{12d} \rfloor$ horizontal lines $\ell_1^h, \dots, \ell_{k_h}^h$ with y -coordinate modulo $12d$ to 0. Finally, we consider the tiling of P that is induced by the arrangement induced by $\ell_1^v, \dots, \ell_{k_v-1}^v, \ell_1^h, \dots, \ell_{k_h-1}^h$ and the boundary of P . This implies that the side length of a tile is upper-bounded by $24d - 1$.

Finally, computing the flow G_T is straightforward by considering the tiling T and the robots' start and target positions.

3.2.2 Step 2: Ensuring planarity and unidirectionality

After initialization, we preprocess G_T , removing edge intersections and bidirectional edges by transforming the start configuration C_s into an intermediate start configuration C'_s , obtaining a planar flow without bidirectional edges. This transformation consists of two steps: (1) ensuring planarity and (2) ensuring unidirectionality.

Step (1): We observe that edge crossings only occur between two diagonal edges with adjacent source tiles, as illustrated in Figure 2(a)+(b). To remove a crossing, it suffices to eliminate one of the diagonal edges by exchange robots between the source tiles. To eliminate all crossings, each robot is moved at most once, because after moving, the robot does no longer participate in a diagonal edge. Thus, all necessary exchanges can be done in $\mathcal{O}(d)$ steps by Lemma 2, covering the tiling T by constantly many layers, similar to the proof of Lemma 2.

Step (2): We delete a bidirectional edge $(v, w), (w, v)$ by moving $\min\{f_T((v, w)), f_T((w, v))\}$ robots with target tile w from v to w and vice versa which achieves that $\min\{f_T((v, w)), f_T((w, v))\}$ robots achieve their target tile w and $\min\{f_T((v, w)), f_T((w, v))\}$ robots achieve their target tile v , thus eliminating the edge with lower flow value. This process is depicted in Figure 2(c)+(d). Like step (1), this can be done in $\mathcal{O}(d)$ parallel steps by Lemma 2. As we do not add any edges, we maintain planarity during step (2). Observe that during the preprocessing, we do not destroy the grid structure of G_T .

Step (1) and step (2) maintain the flow property of f_T without any other manipulations to the flow f_T , because both preprocessing steps can be represented by local circulations.

3.2.3 Step 3: Computing a flow partition

After preprocessing, we partition the flow G_T into d -subflows.

► **Definition 5.** A *subflow* of G_T is a circulation $G'_T = (T, E', f'_T)$, such that $E' \subseteq E_T$, and $0 \leq f'_T(e) \leq f_T(e)$ for all $e \in E'$. If $f'_T(e) \leq z$ for all $e \in E'$ and some $z \in \mathbb{N}$, we call G'_T a z -flow.

The flow partition relies on an upper bound on the maximal edge weight in G_T . By construction, tiles have side length at most $24d$; therefore, each tile consists of at most $576d^2$ unit squares. This yields the following upper bound; a tighter constant factor can be achieved using a more sophisticated argument.

► **Observation 6.** We have $f_T(e) \leq 576d^2$ for all $e \in E_T$.

► **Definition 7.** A (z, ℓ) -partition of G_T is a set of ℓ z -subflows $\{G_1 = (V_1, E_1, f_1), \dots, G_\ell = (V_\ell, E_\ell, f_\ell)\}$ of G_T , such that G_1, \dots, G_ℓ sum up to G_T .

► **Lemma 8.** We can compute a $(d, \mathcal{O}(d))$ -partition of G_T in polynomial time.

Proof sketch. In a slight abuse of notation, throughout this proof, the elements in sets of cycles are not necessarily unique. A $(d, \mathcal{O}(d))$ -partition can be constructed using the following steps.

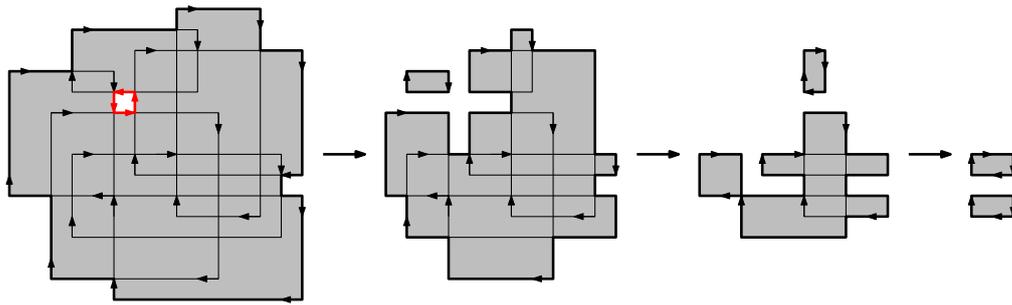
- We start by computing a $(1, h)$ -partition \mathbb{C}_\circ of G_T consisting of $h \leq n_1 n_2$ cycles. This is possible because G_T is a circulation. If a cycle C intersects itself, we subdivide C into smaller cycles that are intersection-free. Furthermore, h is clearly upper bounded by the number of robots $n_1 n_2$, because every robot can contribute only 1 to the sum of all edges in G_T . As the cycles do not self-intersect, we can partition the cycles \mathbb{C}_\circ by their orientation, obtaining the set \mathbb{C}_\circ of clockwise and the set \mathbb{C}_\circ of counterclockwise cycles.
- We use \mathbb{C}_\circ and \mathbb{C}_\circ to compute a $(1, h')$ -partition $\mathbb{C}_\circ^1 \cup \mathbb{C}_\circ^2 \cup \mathbb{C}_\circ^1 \cup \mathbb{C}_\circ^2$ with $h' \leq n_1 n_2$, such that two cycles from the same subset \mathbb{C}_\circ^1 , \mathbb{C}_\circ^2 , \mathbb{C}_\circ^1 , or \mathbb{C}_\circ^2 share a common orientation. Furthermore, we guarantee that two cycles from the same subset are either edge-disjoint or one lies nested in the other. A partition such as this can be constructed by applying a recursive peeling algorithm to \mathbb{C}_\circ and \mathbb{C}_\circ as depicted in Figure 3, yielding a decomposition of the flow induced by \mathbb{C}_\circ into two cycle sets \mathbb{C}_\circ^1 and \mathbb{C}_\circ^2 , where \mathbb{C}_\circ^1 consists of clockwise cycles and \mathbb{C}_\circ^2 consists of counterclockwise cycles, and a similar partition of \mathbb{C}_\circ , see the appendix for details.
- Afterwards, we partition each set \mathbb{C}_\circ^1 , \mathbb{C}_\circ^2 , \mathbb{C}_\circ^1 , and \mathbb{C}_\circ^2 into $\mathcal{O}(d)$ subsets, each inducing a d -subflow of G_T , see the appendix for details. ◀

3.2.4 A subroutine of Step 4: Realizing a single subflow

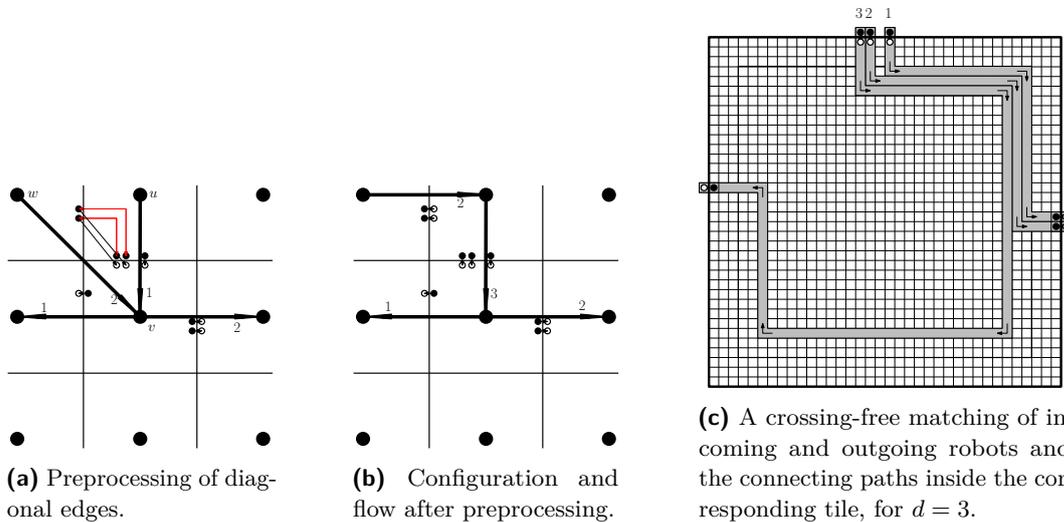
In this section, we present a procedure for *realizing* a single d -subflow G'_T of G_T .

► **Definition 9.** A schedule $t := C_1 \rightarrow \dots \rightarrow C_{k+1}$ *realizes* a subflow $G'_T = (T, E', f'_T)$ if, for each pair v, w of tiles, the number of robots moved by t from their start tile v to their target tile w is $f'_T((v, w))$, where we let $f'_T((v, w)) = 0$ if $(v, w) \notin E'$.

► **Lemma 10.** Let $G'_T = (T, E'_T, f'_T)$ be a planar unidirectional d -subflow. There is a polynomial-time algorithm that computes a schedule $C_1 \rightarrow \dots \rightarrow C_{k+1}$ realizing G'_T for a constant $k \in \mathcal{O}(1)$.



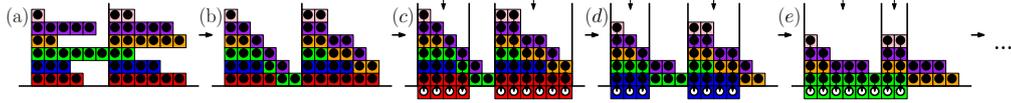
■ **Figure 3** Recursive peeling of the area bounded by the cycles from \mathbb{C}_\circ , resulting in clockwise cycles (thick black cycles). Cycles constituting the boundary of *holes* are counterclockwise (thick red cycles). Note that an edge e vanishes when $f_T(e)$ cycles containing that edge are removed by the peeling algorithm described above.



■ **Figure 4** Procedure for computing transformation steps that realize a d -subflow. Figures (a) and (b) illustrate how we preprocess G'_T such that E'_T consists of horizontal and vertical edges only. Figure (c) illustrates the main approach. White disks illustrate start positions and black disks illustrate target positions.

Proof sketch. We give a high-level description of the proof and refer to the full version of the paper [6] for details.

Our algorithm uses $k = \mathcal{O}(d)$ preprocessing steps $C_1 \rightarrow \dots \rightarrow C_k$, as depicted in Figure 4(a)+(b), and one final realization step $C_k \rightarrow C_{k+1}$, shown in Figure 4(c), pushing the robots from their start tiles into their target tiles. The preprocessing eliminates diagonal edges and places the moving robots next to the border of their target tiles. For the final realization step we compute a pairwise disjoint matching between incoming and outgoing robots, such that each pair is connected by a tunnel inside the corresponding tile in which these tunnels do not intersect, see Figure 4(a). The final realization step is given via the robots' motion induced by pushing each robot into the interior of the tile and by pushing this one-step motion through the corresponding tunnel into the direction of the corresponding outgoing robot. ◀



■ **Figure 5** Stacking robots in lines induced by flows of the edges of the subflows to be realized.

3.2.5 Step 4: Realizing all subflows

Next we extend the idea of Lemma 10 to $\ell \leq d$ subflows instead of one and demonstrate how this can be leveraged to move all robots to their target tile using $\mathcal{O}(d)$ transformation steps.

► **Lemma 11.** *Let $\mathcal{S} := \langle G_1 = (V_1, E_1, f_1), \dots, G_\ell = (V_\ell, E_\ell, f_\ell) \rangle$ be a sequence of $\ell \leq d$ unidirectional planar d -subflows of G_T . There is a polynomial-time algorithm computing $\mathcal{O}(d) + \ell$ transformation steps $C_1 \rightarrow \dots \rightarrow C_{k+\ell}$ realizing \mathcal{S} .*

Proof sketch. We give a high level description of the proof and refer for details to the full version of the paper [6].

Let t be an arbitrary tile. Similar to the approach of Lemma 10, we first apply a preprocessing step guaranteeing that the robots to be moved into or out of t are in the right position close to the boundary of t , see Figure 5. Thereafter we move the robots into their target tiles, using ℓ applications of the algorithm from Lemma 10 without the preprocessing phase. In particular, we realize a sequence of ℓ d -subflows by applying ℓ times the single realization step of Lemma 10. ◀

► **Lemma 12.** *There is a polynomial-time algorithm computing $\mathcal{O}(d)$ transformation steps moving all robots into their target tiles.*

Proof. By Lemma 8, we can compute a (d, cd) -partition of G_T for $c \in \mathcal{O}(1)$. We group the corresponding d -subflows into $\frac{cd}{d} = c$ sequences, each consisting of at most d d -subflows. We realize each sequence by applying Lemma 11, using $\mathcal{O}(d)$ transformation steps for each sequence. This leads to $\mathcal{O}(cd) = \mathcal{O}(d)$ steps for realizing all sequences of d -subflows. ◀

For the proof of Theorem 3, we still need to analyze the time complexity of our approach, for which we refer to the full version of the paper [6].

4 Variants on labeling

A different version is the unlabeled variant, in which all robots are the same. A generalization of both this and the labeled version arises when robots belong to one of k color classes, with robots from the same color class being identical.

We formalize this problem variant by using a coloring $c : \{1, \dots, n_1 n_2\} \rightarrow \{1, \dots, k\}$ for grouping the robots. By populating unoccupied cells with robots carrying color $k + 1$, we may assume that each unit square in the environment P is occupied. The robots *draw an image* $I = (I^1, \dots, I^k)$, where I^i is the set of cells occupied by a robot with color i . We say that two images I_s and I_t are *compatible* if in I_s and I_t the number of cells colored with color i are equal for each color $i = 1, \dots, k$. By moving the robots, we want to transform a start image I_s into a compatible target image I_t , minimizing the makespan.

► **Theorem 13.** *There is an algorithm with running time $\mathcal{O}(k(N)^{1.5} \log(N) + N^5)$ for computing, given start and target images I_s, I_t with maximum distance d between start and target positions, an $\mathcal{O}(1)$ -approximation of the optimal makespan M and a corresponding schedule.*

The basic idea is to transform the given unlabeled problem setting into a labeled problem setting by solving a geometric bottleneck matching problem, see the full version of the paper [6] for details.

5 Continuous motion

The continuous case considers N unit disks that have to move into a target configuration; the velocity of each robot is bounded by 1, and we want to minimize the makespan. For arrangements of disks that are not well separated, we show that constant stretch is *impossible*.

► **Theorem 14.** *There is an instance with optimal makespan $M \in \Omega(N^{1/4})$.*

The basic proof idea is as follows. Let $\{m_1, \dots, m_N\}$ be an arbitrary trajectory set with makespan M . We show that there must be a point in time $t \in [0, M]$ where the area of $\text{Conv}(m_1(t), \dots, m_N(t))$ is lower-bounded by $cN + \Omega(N^{3/4})$, where cN is the area of the convex hull $\text{Conv}(m_1(0), \dots, m_N(0))$ of $m_1(0), \dots, m_N(0)$. Assume $M \in o(N^{1/4})$ and consider the area of $\text{Conv}(m_1(t'), \dots, m_N(t'))$ at some point $t' \in [0, M]$. This area is at most $cN + \mathcal{O}(\sqrt{N}) \cdot o(N^{1/4})$ which is a contradiction. Proof details are given in the full version of the paper [6].

Conversely, we give a non-trivial upper bound on the stretch, as follows.

► **Theorem 15.** *There is an algorithm that computes a trajectory set with continuous makespan of $\mathcal{O}(d + \sqrt{N})$. If $d \in \Omega(1)$, this implies a $\mathcal{O}(\sqrt{N})$ -approximation algorithm.*

The approach of Theorem 15 applies an underlying grid with mesh size $2\sqrt{2}$. Our algorithm (1) moves the robots to vertices of the grid, (2) applies our $\mathcal{O}(1)$ -approximation for the discrete case, and (3) moves the robots from the vertices of the grid to their targets. For a detailed description of the Algorithm of Theorem 15 see the full version of the paper [6].

6 Conclusion

We have presented progress on a number of algorithmic problems of parallel motion planning, also shedding light on a wide range of interesting open problems described in the following.

The first set of problems consider complexity. The labeled problem of Section 3 is known to be NP-complete for planar graphs. It is natural to conjecture that the geometric version is also hard. It seems tougher to characterize the family of optimal trajectories: As shown above, their nature is unclear, so membership in NP is doubtful.

A second set of questions considers the relationship between stretch factor and disk separability in the continuous setting. We believe that the upper bound of $\mathcal{O}(\sqrt{N})$ on the worst-case stretch factor for dense arrangements is tight. What is the critical separability of disks for which constant stretch can be achieved? How does the stretch factor increase as a function of N below this threshold? For *sparse* arrangements of disks, simple greedy, straight-line trajectories between the origins and destinations of disks encounter only isolated conflicts, resulting in small stretch factors close to 1, i.e., $1 + o(1)$. What is the relationship between (local) density and the achievable stretch factor along the whole density spectrum?

Finally, practical motion planning requires a better handle on characterizing and computing optimal solutions for specific instances, along with lower bounds, possibly based on numerical methods and tools. Moreover, there is a wide range of additional objectives and requirements, such as accounting for acceleration or deceleration of disks, turn cost, or multi-stop tour planning. All these are left for future work.

References

- 1 M. Abellanas, S. Bereg, F. Hurtado, A. G. Olaverri, D. Rappaport, and J. Tejel. Moving coins. *Computational Geometry: Theory and Applications*, 34(1):35–48, 2006.
- 2 Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Transactions on Automation Science and Engineering*, 12(4):1309–1317, 2015.
- 3 B. Aronov, M. de Berg, A. F. van der Stappen, P. Švestka, and J. Vleugels. Motion planning for multiple robots. *Discrete & Computational Geometry*, 22(4):505–525, 1999.
- 4 Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Lillian Lin, and Christian Scheffer. Coordinated motion planning: The video. In Csaba Tóth and Bettina Speckmann, editors, *34th International Symposium on Computational Geometry (SoCG 2018, these proceedings)*, volume 99 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:6, 2018. Video available via <https://youtu.be/0OrG72sX5gk>.
- 5 S. Bereg, A. Dumitrescu, and J. Pach. Sliding disks in the plane. *International Journal of Computational Geometry & Applications*, 18(5):373–387, 2008.
- 6 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. *Computing Research Repository (CoRR)*, 1801:1–32, 2018. Available at <https://arxiv.org/abs/1801.01689>.
- 7 A. Dumitrescu and M. Jiang. On reconfiguration of disks in the plane and related problems. *Computational Geometry: Theory and Applications*, 46:191–202, 2013.
- 8 G. W. Flake and E. B. Baum. Rush Hour is PSPACE-complete, or “Why you should generously tip parking lot attendants”. *Theoretical Computer Science*, 270(1):895–911, 2002.
- 9 R. A. Hearn and E. D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1):72–96, 2005.
- 10 R. A. Hearn and E. D. Demaine. *Games, puzzles, and computation*. CRC Press, 2009.
- 11 J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the warehouseman’s problem. *International Journal of Robotics Research*, 3(4):76–88, 1984.
- 12 J. E. Hopcroft and G. T. Wilfong. Reducing multiple object motion planning to graph searching. *SIAM Journal on Computing*, 15(3):768–785, 1986.
- 13 S. Kloder and S. Hutchinson. Path planning for permutation-invariant multi-robot formations. In *IEEE Transactions on Robotics*, volume 22, pages 650–665. IEEE, 2006.
- 14 D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science (SFCS)*, pages 241–250, 1984. doi:10.1109/SFCS.1984.715921.
- 15 J. M. Marberg and E. Gafni. Sorting in constant number of row and column phases on a mesh. *Algorithmica*, 3:561–572, 1988. doi:10.1007/BF01762132.
- 16 Mark Overmars. Contributed open problem. In Sándor P. Fekete, Rudolf Fleischer, Rolf Klein, and Alejandro Lopez-Ortiz, editors, *Robot Navigation, Dagstuhl Seminar 06421*, 2006. URL: <http://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=06421>.
- 17 G. Ramanathan and V. Alagar. Algorithmic motion planning in robotics: Coordinated motion of several disks amidst polygonal obstacles. In *Proceedings of the Second IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 514–522, 1985.
- 18 D. Ratner and M. K. Warmuth. Finding a shortest solution for the $N \times N$ extension of the 15-puzzle is intractable. In *Proceedings of the Fifth AAAI Conference on Artificial Intelligence*, pages 168–172, 1986.

- 19 Christian Scheideler. *Universal routing strategies for interconnection networks*, volume 1390 of *Lecture Notes in Computer Science*. Springer, 1998.
- 20 Jacob T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *International Journal of Robotics Research*, 2(3):46–75, 1983.
- 21 K. Solovey and D. Halperin. k -color multi-robot motion planning. *International Journal of Robotics Research*, 33(1):82–97, 2014.
- 22 K. Solovey, J. Yu, O. Zamir, and D. Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems (RSS)*, 2015.
- 23 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *International Journal of Robotics Research*, 35(14):1750–1759, 2016.
- 24 P. Spirakis and C. K. Yap. Strong NP-hardness of moving many discs. *Information Processing Letters*, 19(1):55–59, 1984.
- 25 M. Turpin, N. Michael, and V. Kumar. Trajectory planning and assignment in multirobot systems. In *Algorithmic Foundations of Robotics X*, pages 175–190. Springer, 2013.
- 26 R. M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16(1):86–96, 1974.
- 27 J. Yu. Constant factor optimal multi-robot path planning in well-connected environments. arXiv. URL: <https://arxiv.org/abs/1706.07255>.
- 28 J. Yu. Constant factor time optimal multi-robot routing on high-dimensional grids in mostly sub-quadratic time. arXiv. URL: <https://arxiv.org/abs/1801.10465>.