

Unlabeled Multi-Robot Motion Planning with Tighter Separation Bounds

Bahareh Banyassady ✉ 

Zuse Institute Berlin, Germany

Mark de Berg ✉ 

TU Eindhoven, The Netherlands

Karl Bringmann ✉

Universität des Saarlandes, Saarbrücken, Germany

Max Planck Institute for Informatics, Saarbrücken, Germany

Kevin Buchin ✉ 


TU Dortmund, Germany

Henning Fernau ✉ 

Universität Trier, Germany

Dan Halperin ✉ 

Tel Aviv University, Israel

Irina Kostitsyna ✉ 

TU Eindhoven, The Netherlands

Yoshio Okamoto ✉ 

The University of Electro-Communications, Tokyo, Japan

Stijn Slot ✉

Adyen, Amsterdam, The Netherlands

Abstract

We consider the unlabeled motion-planning problem of m unit-disc robots moving in a simple polygonal workspace of n edges. The goal is to find a motion plan that moves the robots to a given set of m target positions. For the unlabeled variant, it does not matter which robot reaches which target position as long as all target positions are occupied in the end.

If the workspace has narrow passages such that the robots cannot fit through them, then the free configuration space, representing all possible unobstructed positions of the robots, will consist of multiple connected components. Even if in each component of the free space the number of targets matches the number of start positions, the motion-planning problem does not always have a solution when the robots and their targets are positioned very densely. In this paper, we prove tight bounds on how much separation between start and target positions is necessary to always guarantee a solution. Moreover, we describe an algorithm that always finds a solution in time $O(n \log n + mn + m^2)$ if the separation bounds are met. Specifically, we prove that the following separation is sufficient: any two start positions are at least distance 4 apart, any two target positions are at least distance 4 apart, and any pair of a start and a target positions is at least distance 3 apart. We further show that when the free space consists of a single connected component, the separation between start and target positions is not necessary.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases motion planning, computational geometry, simple polygon

Digital Object Identifier 10.4230/LIPIcs.SoCG.2022.12

Related Version *Full Version*: <https://arxiv.org/abs/2205.07777>

Funding *Mark de Berg*: Supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003.



© Bahareh Banyassady, Mark de Berg, Karl Bringmann, Kevin Buchin, Henning Fernau, Dan Halperin, Irina Kostitsyna, Yoshio Okamoto, and Stijn Slot; licensed under Creative Commons License CC-BY 4.0

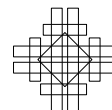
38th International Symposium on Computational Geometry (SoCG 2022).

Editors: Xavier Goaoc and Michael Kerber; Article No. 12; pp. 12:1–12:16



Leibniz International Proceedings in Informatics

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



Dan Halperin: Supported in part by the Israel Science Foundation (grant no. 1736/19), by NSF/US-Israel-BSF (grant no. 2019754), by the Israel Ministry of Science and Technology (grant no. 103129), by the Blavatnik Computer Science Research Fund, and by the Yandex Machine Learning Initiative for Machine Learning at Tel Aviv University.

Yoshio Okamoto: JSPS KAKENHI Grant Numbers JP20H05795 and JP20K11670.

Acknowledgements This research was initiated at the Lorentz-Center Workshop on Fixed-Parameter Computational Geometry, 2018. We thank Gerhard Woeginger for discussions during the workshop.

1 Introduction

Multi-robot systems are already playing a central role in manufacturing, warehouse logistics, inspection of large structures (e.g., bridges), monitoring of natural resources, and in the future they are expected to expand to other domains such as space exploration, search-and-rescue tasks and more. One of the key ingredients necessary for endowing multi-robot systems with autonomy is the ability to plan collision-free motion paths for their constituent robots towards desired target positions.

In the basic multi-robot motion-planning (MRMP) problem several robots are operating in a common environment. We are given a set of start positions and a set of desired target positions for these robots, and we wish to compute motions that will bring the robots to the targets while avoiding collisions with obstacles and the other robots. We distinguish between two variants of MRMP, *labeled* and *unlabeled*, depending on whether each robot has to reach a specific target. In *labeled* robot motion planning, each robot has a designated target position. In contrast, in the *unlabeled* variant, which we study here, each robot only needs to reach *some* target position; it does not matter which robot reaches which target as long as at the end each target position is occupied by a robot.

MRMP is an extension of the extensively studied *single* robot motion-planning problem (see, e.g., [3, 6, 13]). The multi-robot case is considerably harder [7, 8, 23], since the dimension of the *configuration space* grows with the number of robots in the system. The configuration space of a robot system is a parametric representation of all the possible configurations of the system, which are determined by specifying a real value for each independent parameter (degree of freedom) of the system.

The system we study consists of unit-disc robots moving in the plane; see below for a more formal problem statement. Not only is this a reasonably faithful representation of existing robot systems (e.g., in logistics), but it already encapsulates the essential hardships of MRMP, as MRMP for planar systems with simply-shaped robots are known to be hard [8, 20]. Surprisingly, when we assume some minimum spacing between the start and target positions, the problem for robots moving in a simple polygon always has a solution, and the solution can be found in polynomial time, as shown by Adler et al. [1]. The *separation*, the minimum distance between the start and target positions, thus plays a key role in the difficulty of the problem. However the separation bounds assumed by Adler et al. are not proven to be tight, so the question remains for what separation bounds the problem is always solvable. In this paper we determine the minimal separation needed to ensure that the motion-planning problem has a solution, improving on the bounds obtained by Adler et al. We also describe an algorithm that plans such motions efficiently, relying on the new bounds that we obtain.

Related work. The multi-robot motion planning problem has received much attention over the years. Already in 1983, the problem was described in a paper on the *Piano Mover's problem* by Schwartz and Sharir [15]. Later that year, an algorithm for the case of two or

three disc robots moving in a polygonal environment with n polygon vertices was described, running in $O(n^3)$ and $O(n^{13})$ time respectively [16]. This was later improved by Yap [28] to $O(n^2)$ and $O(n^3)$ for two and three robots respectively, using the *retraction method*. A general approach using *cell decomposition* was later developed in 1991 by Sharir and Sifrony [17] that could deal with a variety of robot pairs in $O(n^2)$ time.

Unfortunately, when the number of robots increases beyond a fixed constant, the problem becomes hard. In 1984, a *labeled* case of the multi-robot motion planning with disc robots and a simple polygonal workspace was shown to be strongly NP-hard [23]. This is a somewhat weaker result than the PSPACE-hardness for many other motion planning problems. For rectangular robots in a rectangular workspace, however, the problem was shown to be PSPACE-hard [8]. This result has later been refined to show that for PSPACE-completeness it is sufficient to have only 1×2 or 2×1 robots in a rectangular workspace [7].

The hardness results for the general problem, as well as the often complex algorithms that solve the problem exactly [6], led to the development of more practical solutions, which often trade completeness of the solution for simplicity and speed, and can successfully cope with motion-planning problems with many degrees of freedom. Most notable among the practical solutions are *sampling-based* (SB) techniques. These include the celebrated Probabilistic Roadmaps (PRM) [9], the Rapidly Exploring Random Trees (RRT) [12], and their numerous variants [3, 5, 13]. The probabilistic roadmaps can be widely applied to explore the high-dimensional configuration space, such as settings with a large number of robots or robots with many degrees of freedom. However, in experiments by Sanchez and Latombe [14] already for 6 robots with a total of 36 degrees of freedom the algorithm requires prohibitively long time to find a solution. Svestka and Overmars [25] suggested an SB algorithm specially tailored to many robots. Their solution still requires exorbitantly large roadmaps and is restricted to a small number of robots. Solovey et al. [21] devised a more economical approach, dRRT (for discrete RRT), which is capable of coping with a larger number of robots, and which was extended to produce asymptotically optimal solutions [18], namely converging to optimal (e.g., shortest overall distance) solution as the number of samples tends to infinity.

Regarding separability bounds, Solomon and Halperin [19] studied the labeled version of the unit-disc problem among polygonal obstacles in the plane, and showed that a solution always exists under a more relaxed separation: each start or target position has an *aura*, namely it resides inside a not-necessarily-concentric disc of radius 2, and the auras of two positions (each being start or target) may overlap, as long as the aura of one robot does not intersect the other robot. They do not however make the distinction between monochromatic and bichromatic separation, and impose the same conditions for all auras.

With respect to unlabeled motion planning, the problem was first considered by Kloder and Hutchinson [10] in 2006. In their paper they provide a sampling-based algorithm which is able to solve the problem. In 2016, Solovey and Halperin [20] have shown that for unit square robots the problem is PSPACE-hard using a reduction from *non-deterministic constraint logic* (NCL) [7]. This PSPACE-hardness result also extends to the labeled variant for unit square robots. Just recently, the unlabeled variant for two classes of disc robots with different radii was also shown to be PSPACE-hard [2], with a similar reduction from NCL. In the reduction the authors use robots of radius $\frac{1}{2}$ and 1. In contrast, the earlier NP-hardness result for disc robots by Spirakis and Yap [23] required discs of many sizes with large differences in radii.

Fortunately, an efficient (polynomial-time) algorithm can still exist when some additional assumptions are made on the problem. Turpin, Michael, and Kumar [26] consider a variant of the unlabeled motion-planning problem where the collection of free positions surrounding every start or target position is star-shaped. This allows them to create an efficient algorithm

for which the path-length is minimized. In the paper by Adler et al. [1], an $O(n \log n + mn + m^2)$ algorithm is given for the unlabeled variant, assuming the workspace is a simple polygon and the start and target positions are *well-separated*, which is defined as minimum distance of four between any start or target position. Their algorithm is based on creating a motion graph on the start and target positions and then treating this as an *unlabeled pebble game*, which can be solved in $O(S^2)$ where S is the number of pebbles [11]. Furthermore, in the paper by Adler et al. [1] the separation bound $4\sqrt{2} - 2$ (≈ 3.646) is shown to be sometimes necessary for the problem to always have a solution. When the workspace contains obstacles, Solovey et al. [22] describe an approximation algorithm which is guaranteed to find a solution when one exists, assuming also that the start and target positions are *well-separated* and a minimum distance of $\sqrt{5}$ between a start or target position and an obstacle.

Finally, we mention that multi-pebble motion on graphs, already brought up above, is part of a large body of work on motion planning in discrete domains, sometimes called multi-agent path finding (MAPF), and often adapted to solving continuous problems; see [24] for a review, and [4, 27, 29, 30] for a sample of recent results.

Contributions. We distinguish between two types of separability bounds: *monochromatic*, denoted by μ , the separation between two start positions or between two target positions, and *bichromatic*, denoted by β , between a start and a target position (see Figure 1).

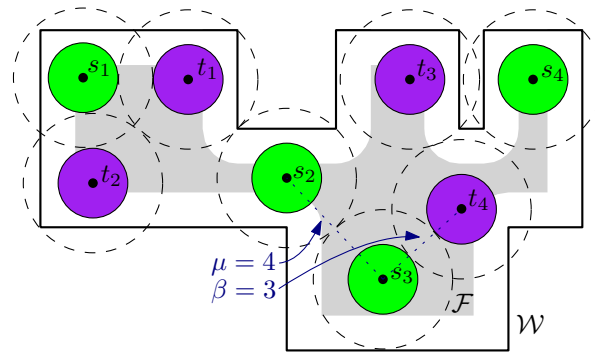
After introducing necessary definitions and notation in Section 2, we begin with a lower bound construction for the monochromatic and bichromatic separation in Section 3. We prove that for $\mu = 4 - \varepsilon$ or for $\beta = 3 - \varepsilon$ (for arbitrarily small positive ε) the solution to the unlabeled multi-robot motion-planning problem in a simple polygon may not always exist.

We devote the remainder of the paper to showing a matching upper bound. We prove that the unlabeled MRMP problem for unit-disc robots in a *simple* polygon is always solvable for monochromatic separation $\mu = 4$ and bichromatic separation $\beta = 3$, assuming that the number of start and target positions match in each free space component. For the case of a single free space component, we show an even stronger result that the problem is always solvable for $\mu = 4$ and $\beta = 0$.

Specifically, in Section 4 we devise an efficient algorithm for MRMP for $\mu = 4$ and $\beta = 2$ in the case of a single free space component, and then extend it to also work for $\mu = 4$ and $\beta = 0$. In Section 5 we extend the algorithm to the case of a free space with multiple components and $\mu = 4$ and $\beta = 3$. Our algorithm runs in $O(n \log n + mn + m^2)$ time, where n is the size of the polygon, and m is the number of robots.

Our results improve upon the results by Adler et al. [1], who describe an algorithm with the same running time that always solves the problem assuming separation of $\mu = \beta = 4$. Similarly to their approach, we restrict the robots to move one at a time on a *motion graph* that has the start and target positions as vertices. Separation of $\mu = \beta = 4$ ensures that the connectivity of the motion graph never changes. However, in our case, the lower bichromatic separation results in a dynamic motion graph: existence of some edges may depend on whether specific nodes are occupied by the robots. Furthermore, the lower bichromatic separation in the case of multiple free space components leads to more intricate dependencies between the components. Nonetheless, we show that there is always an order in which we can process the components, and devise a schedule for the robots to reach their targets.

Due to space restrictions, some proofs are omitted and can be found in the full version of this paper.



■ **Figure 1** Basic definitions. The workspace \mathcal{W} is the rectilinear polygon, the free space \mathcal{F} is the inner gray area. The aura of a start or target position is shown as a dashed circle of radius two (for unit-disc robots). The monochromatic separation $\mu = 4$, the bichromatic separation $\beta = 3$.

2 Definitions and notation

We consider the problem of m indistinguishable unit-disc robots moving in a simple polygonal workspace $\mathcal{W} \subset \mathbb{R}^2$ with n edges. The *obstacle space* \mathcal{O} is the complement of the workspace, that is, $\mathcal{O} = \mathbb{R}^2 \setminus \mathcal{W}$. We refer to points $x \in \mathcal{W}$ as *positions*, and we say that a robot is at position x when its center is positioned at point $x \in \mathcal{W}$.

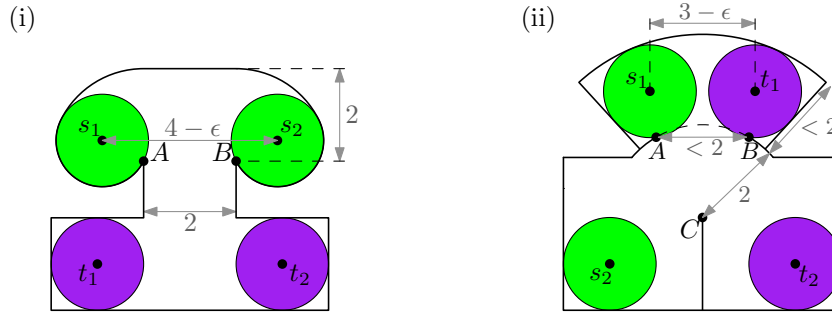
For given $x \in \mathbb{R}^2$ and $r \in \mathbb{R}_+$, we define $\mathcal{D}_r(x)$ to be the open disc of radius r centered at x . The unit-disc robots are defined to be open sets. Thus, a robot collides with the obstacle space \mathcal{O} if and only if its center is at a distance strictly less than 1 from \mathcal{O} . We can now define the *free space* \mathcal{F} to be all positions where a unit-disc robot does not collide with obstacle space, or, more formally, $\mathcal{F} = \{x \in \mathbb{R}^2 \mid \mathcal{D}_1(x) \cap \mathcal{O} = \emptyset\}$. The free space is therefore a closed set. We refer to the connected components of \mathcal{F} as *free space components*.

As the robots are defined to be open sets, two robots collide if the distance between their positions is strictly less than 2. In other words, if a robot occupies a position x then no other robot can be at a position $y \in \mathcal{D}_2(x)$; we call $\mathcal{D}_2(x)$ the *aura* of the robot at position x . In our figures the auras are indicated by dashed circles (see Figure 1).

Unlabeled multi-robot motion-planning problem. Given a set S of m start positions and a set T of m target positions, where $S, T \subset \mathcal{F}$, the goal is to plan a *collision-free* motion for m robots from S to T , such that by the end of the motion every target position in T is occupied by some robot. Since the robots are indistinguishable (i.e., unlabeled), it does not matter which robot ends up at which target position. More formally, we wish to find continuous paths $\pi_i: [0, 1] \rightarrow \mathcal{F}$, for $1 \leq i \leq m$, such that $\pi_i(0) = s_i$ and $\{\pi_i(1) \mid 1 \leq i \leq m\} = T$. Furthermore, we require that, at any moment in time $\tau \in [0, 1]$, for all robots i , no other robot j is in the aura of robot i , $\pi_j(\tau) \notin \mathcal{D}_2(\pi_i(\tau))$. In our figures we indicate start positions by green unit discs centered at points in S , and target positions by purple unit discs centered at points in T .

For a subset $Q \subset \mathcal{F}$ of the free space, we use $S(Q) = S \cap Q$ to denote the set of start positions that reside in Q , and similarly $T(Q) = T \cap Q$ to denote the set of target positions in Q . We define the *charge* $q(Q)$ as the difference between the number of start and target positions in Q , $q(Q) = |S(Q)| - |T(Q)|$. For each free space component F_i , we require that $q(F_i) = 0$, i.e., there needs to be an equal number of start and target positions.

Finally, we state below a few useful properties proven by Adler et al. [1].



■ **Figure 2** (i) An instance for $\mu = 4 - \epsilon$ with one free space component. The robots are blocking each other from entering the corridor. (ii) An instance for $\beta < 3 - \epsilon$. The distance $|AB| < 2$ is too small for a robot to pass through, thus there are two free space components. The robot in the top component is blocking the one in the bottom component.

► **Lemma 1** ([1]). *Each component F_i of the free space is simply connected.*

► **Lemma 2** ([1]). *For any $x \in \mathcal{F}$, let F_i be the connected component of the free space containing x . Then the set $\mathcal{D}_2(x) \cap F_i$ is connected.*

3 Tighter separation bounds

In this section we explore the separation between the start and target positions that is necessary for the problem to always have a solution. We show that, without a certain amount of monochromatic separation (μ) and bichromatic separation (β), there are instances of the problem that cannot be solved, thus certain separation is necessary for the problem to always have a solution. We first prove that a separation of $\mu = 4$ is necessary. This bound is tight and it improves a previous lower bound of $\mu = 4\sqrt{2} - 2 (\approx 3.646)$ [1]. We then show that $\beta = 3$ is also necessary.

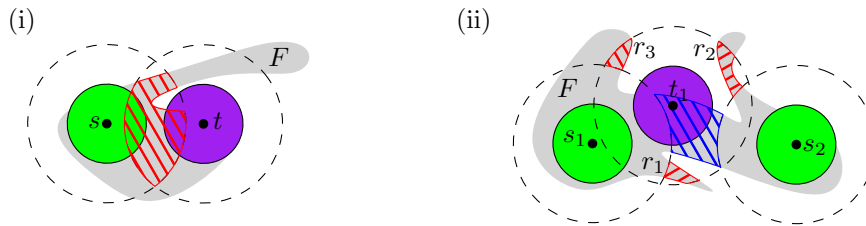
The following lemma is proven using the construction in Figure 2 (i).

► **Lemma 3.** *For $\mu < 4$ a solution does not always exist, even if the free space consists of a single connected component containing two start and two target positions.*

Thus, for a solution to always exist, a monochromatic separation of $\mu = 4$ is necessary. Since the problem for $\mu = \beta = 4$ always has a solution, the monochromatic separation is tight. Hence, we aim to reduce the bichromatic separation β . The proof of the following lemma uses the construction in Figure 2 (ii).

► **Lemma 4.** *For $\beta < 3$ a solution does not always exist, even if there are only two free space components, each containing one start and one target position.*

The lower bound construction for $\beta < 3$ has two free space components with one robot in each. A robot in the top free space component is blocking the motion of a robot in the bottom component, no matter which position it is in. Thus, the lower bound is not applicable if the free space has only one component. Indeed, as we show in the next section, in this case no bichromatic separation is necessary.



■ **Figure 3** (i) When $\beta < 4$, a robot cannot cross the intersection of the auras of s and t (in red) if either s or t is occupied. (ii) $\mathcal{D}_2^-(t_1)$ consists of multiple connected components (remote in red and non-remote in blue). Remote components r_1 and r_2 are blocking areas associated with blocker t_1 .

4 A single free space component

In this section we consider the multi-robot motion-planning problem for the case where the free space consists of a single component F . Initially, for simplicity, we assume $\mu = 4$ and $\beta = 2$. That is, no start/target position can be inside the aura of another start/target position. We later modify the algorithm to handle the case with no bichromatic separation.

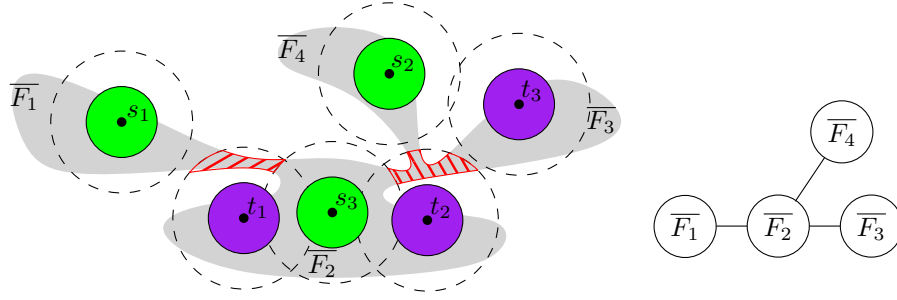
The algorithm by Adler et al. [1] uses the separation assumption $\mu = \beta = 4$, and cannot be applied if $\beta < 4$. Their algorithm greedily moves the robots to the target positions, and may not always be able to find a solution in our case. Indeed, a pair of a start and a target positions whose auras intersect can possibly block the path for robots who need to go through the intersection of these auras (see Figure 3(i)). Therefore, in our algorithm we need to handle such blocking positions.

4.1 Preliminaries

Remote components. Let $A(S) = \bigcup_{s \in S} \mathcal{D}_2(s)$ be the union of all auras of the start positions S . For a target position $t \in T$, let $\mathcal{D}_2^-(t) = (\mathcal{D}_2(t) \cap F) \setminus A(S)$ be the portion of F within the aura of t minus the auras of the start positions in S (see Figure 3(ii)). Note that even though, by Lemma 2, $\mathcal{D}_2(x) \cap F$ is always connected for any $x \in F$, the region $\mathcal{D}_2^-(t)$ may consist of multiple connected components (split by the auras of start positions). One of these components contains t (shown in blue in the figure). A component of $\mathcal{D}_2^-(t)$ that does not contain t is called a *remote component* of t (shown in red in the figure). Let R be the set of remote components for all target positions in T .

Blockers and blocking areas. Consider the example in Figure 3(ii). If t_1 is occupied, its remote components r_1 , r_2 , and r_3 cannot be crossed by a moving robot. Crossing the remote component r_3 can be avoided by moving along its boundary. However remote components r_1 and r_2 pose a problem, as they cut the free space, and thus crossing them cannot be avoided. We call such remote components *blocking areas*.

For a target position $t \in T$, a *blocking area* is a remote component of t that partitions F into multiple components. If t is associated with at least one blocking area, we refer to t as a *blocker*. A blocker might have multiple associated blocking areas (as in Figure 3(ii)). Let $B \subseteq R$ be the set of blocking areas for all target positions in T .



■ **Figure 4** An example with two blockers t_1 and t_2 , and their associated blocking areas shown in red. The corresponding residual components graph H is illustrated on the right side. Note that there is no edge between \overline{F}_4 and \overline{F}_3 , since the blocker t_2 is not located in either of them.

For a blocking area $b \in B$ associated with position t , let the *blocking path* be any path $\pi \subset \mathcal{D}_2(t)$ connecting b to t . By Lemma 2, π exists, and by definition of the blocking area, π crosses the aura of at least one start position. We further show in the following lemma that this path does not intersect any other blocking area.

▶ **Lemma 5.** *For a blocking area $b_x \in B$ and its associated blocker x , there exists some blocking path π such that $\pi \subset \mathcal{D}_2(x)$ and π does not intersect a blocking area b_y of any other blocker y .*

Residual components. Let $\overline{F} = F \setminus \bigcup R$ be the portion of the free space F that does not intersect any remote component in R . By definition, a blocking area partitions F into multiple connected components. Since some remote components are blocking areas, \overline{F} may consist of multiple connected components. We refer to the connected components of \overline{F} as *residual components*. Next, let $F^* = \overline{F} \setminus A(S)$ be the portion of the free space F that does not intersect either the aura of a start position or a remote component of a target position.

▶ **Lemma 6.** *Given m starting and target positions in a polygonal workspace of size n , the subsets \overline{F} and F^* of the free space, and the remote components R , all have complexity $O(m+n)$ and can be computed in $O((m+n)\log(m+n))$ time.*

Residual components graph. We define the *residual components graph* as $H = (V^H, E^H)$ where V^H contains one vertex for each residual component of \overline{F} (see Figure 4). There is an edge between two vertices $v_1, v_2 \in V^H$ if their respective residual components are separated by a single blocking area $b \in B$ and its associated blocker t resides in the respective residual component of either v_1 or v_2 . Although a single blocking area in B can divide \overline{F} into more than two connected components, such a blocking area does not create a cycle in H . This is due to the definition of an edge in H which requires the associated blocker to be in one of the two components. Next lemma follows directly from Lemma 5.

▶ **Lemma 7.** *Any blocking area $b \in B$ shares a boundary with the residual component containing its associated blocker t .*

Next we show that H is a tree by construction.

▶ **Lemma 8.** *The residual components graph H is a tree.*

The general idea of our algorithm is to use the residual components graph H to help us split the problem into smaller subproblems. Using the graph H , we will iteratively choose a leaf residual component \bar{F}_i with a non-positive charge (recall that the charge of a component is the number of start positions minus the number of the target positions), and solve the subproblem restricted to that component using its motion graph, which we define shortly. If afterwards \bar{F}_i will require more robots, they will be moved from a neighboring residual component, ensuring that the blocking area is free for the robots to pass.

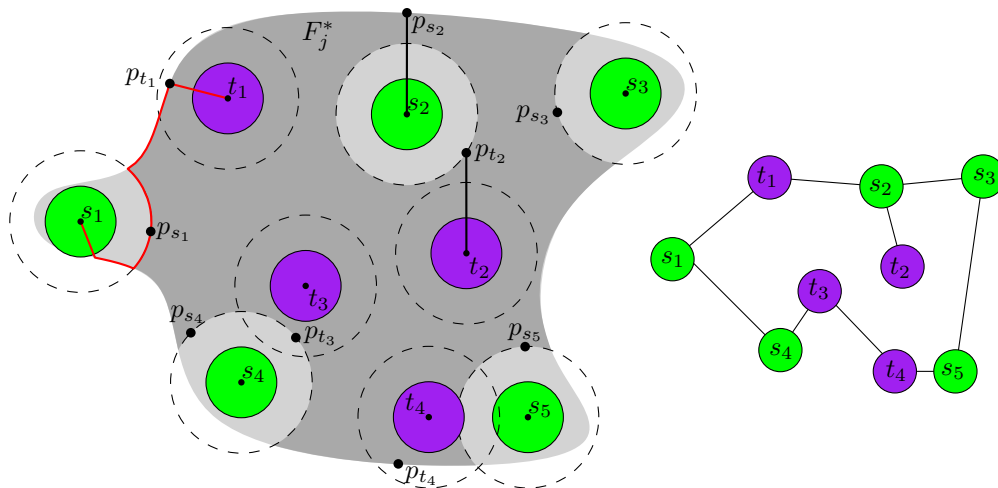
4.2 The motion graph

We now introduce the motion graph, which captures *adjacencies* between the start/target positions. Similarly to [1], the underlying idea of our algorithm is to always have the robots positioned on start or target positions and, using the motion graph, to move one robot at a time between these positions until all target positions are occupied.

Recall that for now we assume that the free space consists of one connected component F . For a free space F with start positions S and target positions T , we define the *motion graph* $G = (V^G, E^G)$, where $V^G = S \cup T$. The edges E^G in G are of two types: *guaranteed* or *blockable*, which we formally define below. Guaranteed edges correspond to so called *guaranteed paths*, where a path in the free space F between $u, v \in S \cup T$ is said to be *guaranteed* if it does not intersect the aura of any position other than u and v .

Unlike guaranteed, blockable edges correspond to paths in F that must cross blocking areas. Our algorithm requires the motion graph G to be connected. However, as $\beta < 4$, without blockable edges the motion graph may be disconnected. Introducing blockable edges ensures that G is connected.

Guaranteed edges. First, we define the guaranteed edges in E^G and show how to construct corresponding guaranteed paths. Recall that we define the set F^* to be the free space minus the auras of the start positions and the remote components, $F^* = F \setminus (\bigcup_{s \in S} \mathcal{D}_2(s) \cup \bigcup R)$.



■ **Figure 5** An illustration of generating the guaranteed edges in a single component F_j^* . Component F_j^* is shown in dark grey; Λ_j is $\langle p_{s_1}, p_{t_1}, p_{s_2}, p_{s_3}, p_{s_5}, p_{t_4}, p_{t_3}, p_{s_4} \rangle$. A path between a pair of adjacent positions is shown in red. The motion graph is shown on the right.

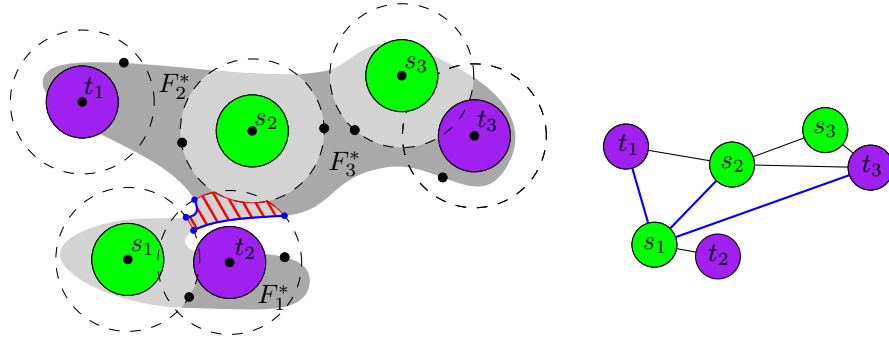
12:10 Unlabeled Multi-Robot Motion Planning with Tighter Separation Bounds

Consider a connected component $F_j^* \subset F^*$. Note that F_j^* may not be simply-connected, as it may contain holes due to subtracted auras of start positions. Abusing the notation, by ∂F_j^* we refer to the outer boundary of F_j^* . For ∂F_j^* , we create an ordered circular list Λ_j of points along ∂F_j^* as follows (see Figure 5).

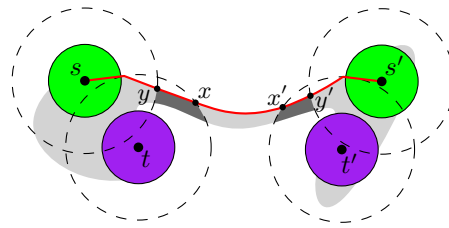
- (i) For each target position $t \in T \cap F_j^*$ whose aura intersects ∂F_j^* , we pick a set of representative points P_t such that P_t contains one point on each connected component of $\partial F_j^* \cap \mathcal{D}_2(t)$. The points P_t are stored in Λ_j based on their ordering along ∂F_j^* .
- (ii) For each position x which is (1) either a target position in F_j^* whose aura does not intersect ∂F_j^* , or (2) a start position corresponding to a hole in F_j^* , we shoot a ray vertically upwards until it hits either ∂F_j^* or the aura of another position y . In the former case the first intersection point p_x is added to Λ_j as a representative point of x . In the latter case a guaranteed edge is added to E^G connecting x and y .
- (iii) Now, consider a start position s whose aura shares a boundary with ∂F_j^* . Note that $\partial F_j^* \cap \mathcal{D}_2(s)$ is connected. If we can pick a representative point p_s on $\partial F_j^* \cap \mathcal{D}_2(s)$ such that there exists an unobstructed path in F connecting s to p_s , then we insert p_s to Λ_j based on its ordering along ∂F_j^* . Otherwise, if for every choice of $p_s \in \partial F_j^* \cap \mathcal{D}_2(s)$ any path connecting s to p_s crosses an aura of some target position t , then we add a guaranteed edge to E^G connecting s and t (for every such target position t). Observe, that by the definition of remote components, if a path connecting s to p_s crosses the aura of t , it must cross it through the non-remote component of t . Thus, there must exist a guaranteed path connecting s and t .

Now that Λ_j is generated, we add a guaranteed edge to the motion graph between any two nodes in G whose representative points are consecutive in Λ_j . If multiple edges between two vertices and self-loops are generated, we remove them in a post-processing step. We repeat this procedure for every connected component $F_j^* \subset F^*$.

Blockable edges. For any blocking area $b \in B$ and its associated blocker t , each section of ∂b is either shared with (1) the boundary of the aura of t , (2) with the boundary of the aura of some start position in S , or (3) with ∂F . See Figure 6 for an illustration. We call a section of ∂b which is shared with ∂F a *free boundary* segment of b . For any free boundary segment of b with endpoints x and y , we assign a set of *incident* positions in $S \cup T$ to x and to y (see below for details). We then add a blockable edge to the motion graph between every pair of incident positions of x and y respectively. Consider an endpoint x of a free boundary segment of b . The set of incident positions of x is defined as follows.



■ **Figure 6** An illustration of a blocking area (in red) with its free boundary (in blue). On the right, the motion graph is shown, with the guaranteed edges (in black) and blockable edges (in blue).



■ **Figure 7** The special case when a blockable edge is added across two blocking areas.

- (i) If x is also an endpoint of a section of ∂b that is shared with $\partial D_2(s)$ for $s \in S$, then s is the only incident position for x .
- (ii) If x is also an endpoint of a section of ∂b that is shared with $\partial D_2(t)$, then x lies on the boundary of a component of F^* . Let that component be F_j^* . Based on the position of x on ∂F_j^* , using Λ_j , we find the predecessor and the successor points of x in Λ_j . By construction of Λ_j , these points are representative of some positions in $S \cup T$. We select those positions as the incident positions for x . The special case that Λ_j is empty, is handled separately and is explained next.

For the special case when Λ_j of F_j^* is empty, if b is the only blocking area incident to ∂F_j^* , then F_j^* does not contain any position in $S \cup T$, and can be ignored. Otherwise, if F_j^* is adjacent to another blocking area b' (of some blocker t'), then from x we follow ∂F_j^* until we reach $\partial b'$ at x' , which must be the endpoint of a free boundary segment of b' (see Figure 7). Let y' be the other endpoint of that free boundary segment. We now select the incident positions of y' as the incident positions of x , i.e., we add a blockable edge between the incident positions of y and those of y' . This results in blockable edges associated with two blocking areas b and b' .

Translating motion graph edges to free space paths. Consider a component $F_j^* \subset F^*$, and let Λ_j be the circular list of representative positions constructed for F_j^* . Let u and v be two positions whose representative points are adjacent in Λ_j . By definition, $(u, v) \in E^G$ is a guaranteed edge, and we claim that there exists a guaranteed path between u and v in F . We construct such a path π_{uv} in the following way.

- (i) First, π_{uv} connects u to its representative point p_u by either following an unobstructed path from u to p_u within u 's aura, or by following the vertical ray used to generate p_u outside of u 's aura.
- (ii) Next, π_{uv} connects p_u to the representative point p_v of v by following ∂F_j^* .
- (iii) Finally, π_{uv} connects p_v to v similarly to (i).

Now consider the case when a guaranteed edge (u, v) is constructed without adding the representative points to Λ_j . If (u, v) is constructed according to the case (ii) of the definition of the guaranteed edges, and without loss of generality the vertical ray emanates from u , then the guaranteed path π_{uv} consists of the vertical segment up_u and an unobstructed path connecting the representative point p_u to the node v within the aura of v . If (u, v) is constructed in case (iii), and without loss of generality $u \in S$ and $v \in T$, then the guaranteed path π_{uv} consists of a path from u until the first intersection with the aura of v and an unobstructed path to v within its aura.

The paths for blockable edges are constructed in the following way. Each endpoint of a free boundary segment is incident to at most two representative points in some list Λ_i . Thus, each free boundary segment (x, y) of every blocking area b contributes up to four edges to the motion graph. Consider a blockable edge (u_x, v_y) between an incident position u_x of x and an incident position v_y of y . The corresponding path consists of three parts.

- (i) From u_x to x . This part is generated similarly to the part (i) for guaranteed edges.
- (ii) From x to y . This part follows the free boundary of b between x and y .
- (iii) From y to v_y . This part is generated similarly to the part (iii) for guaranteed edges.

The following proposition, lists five properties of the motion graph, which will be used to derive the correctness and the complexity of the algorithm.

► **Proposition 9.** *The following properties of a motion graph G hold.*

1. *There exists a guaranteed path in F for each guaranteed edge in G .*
2. *There exists a path in G consisting solely of guaranteed edges between any two positions inside the same residual component $\overline{F}_j \in \overline{F}$.*
3. *G is connected.*
4. *The number of edges $|E^G|$ in G is bounded by $O(m)$.*
5. *Between any two vertices of G , we can find a path in $O(m)$ time, and the corresponding path in the free space has complexity of $O(m + n)$.*
6. *The motion graph G can be created in $O(mn + m^2)$ time.*

4.3 The algorithm

We are now ready to describe our algorithm. We use the residual components graph H , which is a tree, in order to split the problem into smaller subproblems, and recursively solve them. Using H we select a particular residual component of the free space, and solve the subproblem restricted to it using the motion graph. Proposition 9 will help us ensure that such reconfiguration is always possible. One key point is to select a vertex of H , such that, after solving the subproblem in the corresponding residual component, no robots need to move across the incident blocking areas. That is, we need to choose the residual components in such an order that we can ignore blockers in the solved residual components.

Recall that a charge $q(Q)$, for some $Q \subseteq F$, is the difference between the numbers of the start positions and the target positions in Q . Initially, if there is an edge $e \in E^H$ such that removing e splits H into two subtrees with zero total charge each, then we remove e from H and recurse on the two subtrees.

Let us now assign an orientation to the edges of H in the following way. For each edge $e = (u, v)$, let H_u and H_v be the two trees of $H \setminus \{e\}$ containing u and v , respectively. We orient e from u to v if $q(H_u) > 0 > q(H_v)$, and from v to u if $q(H_u) < 0 < q(H_v)$.

► **Lemma 10.** *There exists at least one sink vertex in the directed acyclic graph H .*

Using Lemma 10, our algorithm selects a sink node σ of H . The respective residual component \overline{F}_σ is solved as follows. First, all robots inside \overline{F}_σ are moved to unoccupied target positions. Since all incident edges of σ in H are directed inwards, each edge requires one or more robot(s) to move into \overline{F}_σ . The exact number can be computed from the charges of the subtrees of the adjacent residual components. We then move the required number of robots the adjacent residual components (and farther residual components if needed) to \overline{F}_σ over the corresponding blocking areas. Consider the blocker t associated with a blocking area b incident to \overline{F}_σ . If t is occupied before the charge of \overline{F}_σ becomes zero, then t has to reside in \overline{F}_σ , as the adjacent residual components were not yet processed by the algorithm. Then we move the robot from t to another unoccupied target in \overline{F}_σ , and the now unoccupied blocker position t is the last target to become occupied by a robot moving across b .

Once all target positions of \overline{F}_σ are filled, σ and its incident edges are removed from H , and we recurse on the remaining subtrees. Due to the way we select σ , and the number of robots that are moved into \overline{F}_σ , each subtree has a total zero-charge.

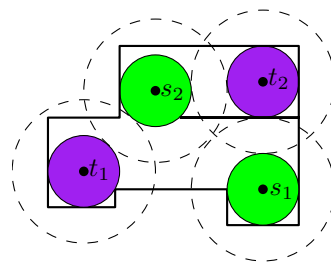
► **Theorem 11.** *When the free space consists of a single connected component, our algorithm finds a solution to the unlabeled motion planning problem for unit-disc robots in a simple workspace, assuming monochromatic separation $\mu = 4$ and bichromatic separation $\beta = 2$. This takes $O(n \log n + mn + m^2)$ time.*

We now show how to extend our approach to $\beta = 0$. In the above algorithm, we use the bichromatic separation $\beta = 2$ to ensure that at every moment in time any subset of nodes of the motion graph can be occupied by robots. If $\beta < 2$ we can no longer assume that any start position and any target position can be occupied at the same time. Nevertheless, even when $\beta = 0$, observe that, due to $\mu = 4$, for a pair of start and target positions s_i and t_j such that $|s_i t_j| < 2$, no other target position t_k can lie in $\mathcal{D}_2(s_i)$, and no other start position s_ℓ can lie in $\mathcal{D}_2(t_j)$. Thus, there is a guaranteed path from s_i to t_j . This can be exploited to adjust the motion graph and the algorithms for $\beta = 0$. Specifically, for each pair of such s_i and t_j , we create a single target node in our motion graph, we move the robot from s_i to t_j , and adjust our algorithm to work for the case of different number of start and target nodes in the motion graph.

► **Theorem 12.** *When the free space \mathcal{F} consists of a single component, the algorithm finds a solution to the unlabeled motion planning problem for unit-disc robots in a simple workspace, assuming monochromatic separation $\mu = 4$. This takes $O(n \log n + mn + m^2)$ time.*

5 Multiple free space components

In this section we consider the case where the free space \mathcal{F} consists of multiple connected components. Since a separation of $\beta = 3$ is necessary to guarantee a solution, we now assume separation bounds of $\mu = 4$ and $\beta = 3$.



■ **Figure 8** An example of a position (s_2) blocking movement (s_1 to t_1) in another free space component.

Within each free space component we can use the algorithm from Section 4. However, paths, that are otherwise valid, may be blocked by a robot from another component (see Figure 8). In this example, there is a simple solution: Move the robot away from s_2 toward t_2 in the upper component, before moving the robot from s_1 to t_1 in the lower component. In the following we prove that there always exists an order on the free space components such that the motion planning problem can be solved by solving the problem component by component in that order.

Let F, F' be two distinct components of \mathcal{F} , and let $x \in F$ be such that $\mathcal{D}_2(x) \cap F' \neq \emptyset$. Since the workspace is simple, it is sufficient to prove that for any such pair of components there is an order between them such that we can first solve one component and then the other. There are two reasons why such an ordering may not exist.

Firstly, there might be a start position $s \in F$ and a start position $s' \in F'$ such that $\mathcal{D}_2(s) \cap F' \neq \emptyset$ and $\mathcal{D}_2(s') \cap F \neq \emptyset$, that is, a start position in F interferes with paths in F' and vice versa. However, Adler et al. [1] proved that with $\mu = 4$, this cannot be the case. Likewise it cannot happen that a target position in F interferes with paths in F' and at the same time a target position in F' interferes with paths in F .

Secondly, there might be a start and target positions $s, t \in F$ both interfering with paths in F' . Because we only have a separation bound of $\beta = 3$ between s and t , this may actually occur. However, interference does not always affect the connectivity of the affected free space component. Therefore, we define a position x (start or target) to be a *remote blocker* of a free space component F' if (1) $x \notin F'$, and (2) $\mathcal{D}_2(x)$ intersects $\partial F'$ in more than one connected component.

► **Lemma 13.** *If the unlabeled motion planning problem has no remote blockers, then there is always a solution.*

Now the key geometric observation is that if auras of both s and t intersect F' , they cannot be both remote blockers of F' , and as a consequence we can still always find an order to resolve F and F' .

► **Theorem 14.** *We are given m unit-disc robots in a simple polygonal workspace $\mathcal{W} \subset \mathbb{R}^2$, with start and target positions S, T and separation constraints $\mu = 4$ and $\beta = 3$. Assuming each connected component F of the free space \mathcal{F} for a single unit-disc robot in \mathcal{W} contains an equal number of start and target positions, there exists a collision-free motion plan for the robots starting at S such that all target positions in T are occupied after execution.*

6 Conclusion

In this paper we presented an efficient algorithm for the unlabeled motion planning problem for unit-disc robots with sufficient separation in a simple polygon. Our result is optimal, in the sense that with less separation a solution may not exist. Nevertheless, there remain a number of challenging open problems.

To prove the tightness of the separation bounds, we first constructed domains with straight-line segments and circular arcs as boundaries, and then obtained simple polygons by approximating these. This results in polygons of high complexity. An open question remains whether it is possible to prove the separation bounds with constant-complexity polygons.

Of course, a solution may still exist even if the separation bounds are violated. The complexity of the problem in this setting remains a challenging open problem. The general unlabeled motion planning problem in a polygonal environment with holes is PSPACE-complete [2, 20]. Does the restriction to unit-disc robots and/or simple domains make the problem tractable, in particular if we still enforce some small separation bound?

What challenges arise when the workspace is no longer simple, but rather contains obstacles? Intuitively, obstacles seem to pose an issue when defining an ordering for solving multiple free space components, since positions can interfere between components at multiple locations. Are there conditions, similar to the separation bounds, which can guarantee a solution (together with an efficient algorithm) for unlabeled multi-robot motion planning amidst obstacles?

References

- 1 Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. In *Algorithmic Foundations of Robotics XI*, pages 1–17. Springer, 2015.
- 2 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 *10th International Conference on Fun with Algorithms (FUN 2021)*, volume 157 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:16, 2020.
- 3 Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT Press, 2005.
- 4 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 Journal on Computing*, 48(6):1727–1762, 2019.
- 5 Dan Halperin, Lydia Kavraki, and Kiril Solovey. Robotics. In Jacob E. Goodman, Joseph O’Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 51, pages 1343–1376. Chapman & Hall/CRC, 3rd edition, 2018.
- 6 Dan Halperin, Micha Sharir, and Oren Salzman. Algorithmic motion planning. In Jacob E. Goodman, Joseph O’Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50, pages 1311–1342. Chapman & Hall/CRC, 3rd edition, 2018.
- 7 Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343:72–96, 2005.
- 8 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.
- 9 Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
- 10 Stephen Kloder and Seth Hutchinson. Path planning for permutation-invariant multirobot formations. *IEEE Transactions on Robotics*, 22(4):650–665, 2006.
- 11 Daniel M. Kornhauser, Gary Miller, and Paul Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. Master’s thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1984.
- 12 James J. Kuffner and Steven M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, 2000.
- 13 Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
- 14 Gildardo Sanchez and Jean-Claude Latombe. Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2112–2119. IEEE, 2002.
- 15 Jacob T. Schwartz and Micha Sharir. On the “piano movers” problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4(3):298–351, 1983.
- 16 Jacob T. Schwartz and Micha Sharir. On the piano movers’ problem: III. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research*, 2(3):46–75, 1983.
- 17 Micha Sharir and Shmuel Sifrony. Coordinated motion planning for two independent robots. *Annals of Mathematics and Artificial Intelligence*, 3(1):107–130, 1991.
- 18 Rahul Shome, Kiril Solovey, Andrew Dobson, Dan Halperin, and Kostas E. Bekris. dRRT^{*}: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots*, 44(3-4):443–467, 2020.

12:16 Unlabeled Multi-Robot Motion Planning with Tighter Separation Bounds

- 19 Israela Solomon and Dan Halperin. Motion planning for multiple unit-ball robots in \mathbb{R}^d . In Marco Morales, Lydia Tapia, Gildardo Sánchez-Ante, and Seth Hutchinson, editors, *Proc. 13th Workshop on the Algorithmic Foundations of Robotics, WAFR*, volume 14 of *Springer Proceedings in Advanced Robotics*, pages 799–816. Springer, 2018.
- 20 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *The International Journal of Robotics Research*, 35(14):1750–1759, 2016.
- 21 Kiril Solovey, Oren Salzman, and Dan Halperin. Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *International Journal of Robotics Research*, 35(5):501–513, 2016.
- 22 Kiril Solovey, Jingjin Yu, Or Zamir, and Dan Halperin. Motion planning for unlabeled discs with optimality guarantees. In *Robotics: Science and Systems XI*. Robotics: Science and Systems Foundation, 2015.
- 23 Paul Spirakis and Chee K. Yap. Strong NP-hardness of moving many discs. *Information Processing Letters*, 19(1):55–59, 1984.
- 24 Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In Pavel Surynek and William Yeoh, editors, *Proc. 12th International Symposium on Combinatorial Search, SOCS*, pages 151–159. AAAI Press, 2019.
- 25 Petr Svestka and Mark H. Overmars. Coordinated path planning for multiple robots. *Robotics and Autonomous Systems*, 23(3):125–152, 1998.
- 26 Matthew Turpin, Nathan Michael, and Vijay Kumar. Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In *IEEE International Conference on Robotics and Automation*, pages 842–848. IEEE, 2013.
- 27 Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.
- 28 Chee Yap. Coordinating the motion of several discs. *Courant Institute of Mathematical Sciences*, 1984.
- 29 Jingjin Yu. Constant factor time optimal multi-robot routing on high-dimensional grids. In Hadas Kress-Gazit, Siddhartha S. Srinivasa, Tom Howard, and Nikolay Atanasov, editors, *Robotics: Science and Systems XIV*, 2018.
- 30 Jingjin Yu and Steven M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.