View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Multi-Robot Motion Planning of k-Colored Discs Is PSPACE-Hard

Thomas Brocken
TU Eindhoven, The Netherlands
t.brocken@student.tue.nl

G. Wessel van der Heijden
TU Eindhoven, The Netherlands
g.w.v.d.heijden@student.tue.nl

Irina Kostitsyna
TU Eindhoven, The Netherlands
i.kostitsyna@tue.nl

Lloyd E. Lo-Wong
TU Eindhoven, The Netherlands
l.e.lo-wong@student.tue.nl

Remco J. A. Surtel
TU Eindhoven, The Netherlands
r.j.a.surtel@student.tue.nl

Abstract

In the problem of *multi-robot motion planning*, a group of robots, placed in a polygonal domain with obstacles, must be moved from their starting positions to a set of target positions. We consider the specific case of unlabeled disc robots of two different sizes. That is, within one class of robots, where a class is given by the robots' size, any robot can be moved to any of the corresponding target positions. We prove that the decision problem of whether there exists a schedule moving the robots to the target positions is PSPACE-hard.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Disc-robot motion planning, algorithmic complexity, PSPACE-hard

Digital Object Identifier 10.4230/LIPIcs.FUN.2021.15

1 Introduction

Due to a wide range of applications, the multi-robot motion planning problem has received a great amount of attention in the theoretical computer science community in recent years. In the most general setting, the problem can be phrased in the following way: given a set of robots placed in a polygonal domain, find a schedule to move the robots from their initial locations to some specified target locations without collisions. From the point of view of identifying which robots move to which target positions, we can distinguish between *labeled* and *unlabeled* robot motion planning. Labeled motion planning is the most studied and, possibly, is a more natural variant of the problem. In it the robots have unique IDs, and each robot has a specifically assigned target location. In this paper, however, we are more interested in unlabeled robot motion planning, where the robots are indistinguishable from one another, and each robot can move to any of the specified target locations. A classic example of a motivating application for this problem is a swarm of robots operating in a warehouse, where it does not matter which of the robots arrives to pick up an item to be transported. Generalizing the notions of labeled and unlabeled motion planning, Solovey and Halperin [6] introduce the *k-color* robot motion planning problem, where given are k classes of robots and k sets of target positions. Within each class the robots are unlabeled,

© Thomas Brocken, G. Wessel van der Heijden, Irina Kostitsyna, Lloyd E. Lo-Wong, and Remco J. A. Surtel;
licensed under Creative Commons License CC-BY
10th International Conference on Fun with Algorithms (FUN 2021).
Editors: Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara; Article No. 15; pp. 15:1–15:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

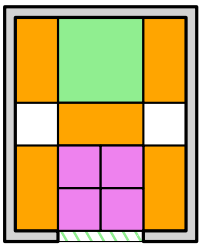


Figure 1 An example of a Sliding Block puzzle. The goal is to take the 2×2 green square outside the box through the exit on the bottom (through which only the green square can slide).

and each robot may move to any location in the corresponding set of target positions. When $k = n$ the problem becomes the standard labeled version of the robot motion planning, and when $k = 1$ it is the unlabeled version.

In this paper we consider the k-color Disc-Robot Motion Planning problem, k-DRMP, where classes of robots differ only by their radii. We show that the problem of deciding whether a particular target location can be reached by a robot from the corresponding class is PSPACE-hard. Our results imply that a version of the Sliding Block game with round pieces can make for a fun and interesting puzzle.

Related work. We start with a brief overview of the known algorithmic results for the disc robot motion planning problem. For unlabeled unit disc robots inside a simple polygon, Adler et al. [1] develop a polynomial-time algorithm to solve the problem under an additional requirement that the distance between any two points from the union of the starting and target locations is at least 4. For unlabeled unit disc robots inside a polygonal domain with obstacles, Solovey et al. [8] show how to find a solution close to optimal in polynomial time. In addition to the same requirement on the separation between the starting/target positions, they require the minimum distance between a robot location and an obstacle to be at least $\sqrt{5}$.

In contrast, for a set of disc robots of possibly different radii in a simple polygon, it is NP-hard to decide whether a target location can be reached by any robot [9].

A wider range of hardness results exists for rectangular or square-shaped robots. Many of these are inspired by *Sliding Block* puzzles, a family of popular games where different shapes are densely packed in a rectangular grid box with little room for movement, and the goal is to free a specific target block and move it outside of the box by sliding the pieces around. Figure 1 shows an example of a puzzle where the blocks are rectangles of integer side length.

One of the earliest results is due to Hopcroft et al. [5], where they show that it is PSPACE-hard to decide whether a given set of rectangular robots enclosed in a rectangular domain can be reconfigured into a particular target configuration. Flake and Baum [3] showed that solving the *Rush Hour* puzzle on an $n \times n$ grid is PSPACE-complete. Rush Hour is a type of a sliding-block puzzle, where the blocks, called cars, are rectangles of width 1 and of length either 2 or 3, and the cars are only allowed to move parallel to their longer side. To prove the hardness of the Rush Hour puzzle, Flake and Baum develop a new specialized model of computation based on "Generalized Rush Hour" logic. They show how to simulate a Finite Turing Machine with circuits built on this logic, which settles the complexity of the problem.

Inspired by Flake and Baum's construction, Hearn and Demaine [4] develop a *Nondeterministic Constraint Logic* (NCL) framework which has proven to be invaluable in showing hardness results for many problems, based on puzzles and otherwise. To showcase

the power of the NCL, they use it to prove PSPACE-completeness of a number of puzzles, including the Sliding Block, even when all blocks are small (in particular, of size 1×2), the classic Rush Hour (for cars of length 2 and 3), and others. Tromp and Cilibrasi [10] used the NCL framework to show that Rush Hour is PSPACE-complete even for the cars of length 2 alone. Finally, using NCL, Solovey and Halperin [7] prove that unlabeled multi-robot motion planning for unit square robots moving amidst polygonal obstacles is PSPACE-hard.

Contribution. The contrast between the abundance of hardness results for rectangular and square robots and a few results, negative as well as positive, for disc robots, suggests that the complexity of the problem greatly depends on the shape of the robots, even when the difference between the shapes is seemingly insignificant. Establishing the complexity of multi-robot motion planning of unit disc robots has been an open problem for quite some time. In this paper we show the first PSPACE-hardness result for motion planning of disc robots. In particular, we show that the 2-color multi-robot motion planning problem for disc robots with radii 1/2 or 1 in a polygonal domain is PSPACE-hard by a reduction from the NCL. In contrast, the NP-hardness construction of [9] uses discs of very different sizes with a large ratio between the largest and the smallest disc.

The rest of the paper is structured in the following way. In Section 2 we introduce a formal problem statement and overview the NCL. In Section 3 we show the hardness reduction. We start with describing the gadgets in Section 3.1, and prove their correctness in Section 3.2. Finally, in Section 3.3 we state our main results.

2 Problem statement and preliminaries

In this section we start with a few definitions, and we state the k-color Disc-Robot Motion Planning (k-DRMP) problem more formally.

Let P be a polygonal domain in the plane. By $D(p, r)$ we denote a disc of radius r centered at a point p. A point $p \in P$ is a *valid* position for a disc robot with radius $r > 0$, if $D(p, r)$ is fully contained in P. A set of points $S = \{p_1, \ldots, p_n\} \subset P$ is a *valid configuration* for a set of robots with radius r if (1) p_i is a valid position for a robot of radius r for all $1 \leq i \leq n$, and (2) discs $D(p_i, r)$ and $D(p_j, r)$ do not intersect in their interior, that is, if $|p_i - p_j| \geq 2r$, for all $1 \leq i < j \leq n$.

For k distinct positive radii $\{r_1, r_2, \ldots, r_k\}$ and k positive integers $\{n_1, n_2, \ldots, n_k\}$, denote a *$k$-configuration* to be a set of k configuration-radius pairs $\mathcal{S} = \{(S_i, r_i) : |S_i| = n_i\}$. A k-configuration is *valid* if each S_i is a valid configuration for disc robots of radius r_i, and for all $i < j$, all $x \in S_i$ and all $y \in S_j$, the discs of respective radii centered at x and y do not intersect in their interior, that is, $|x - y| \geq r_i + r_j$. We will refer to the set of robots with the same radius as a *class* of robots. Thus, each S_i specifies a configuration of a class of robots with radius r_i.

We say that a set of n disc robots with radius r can be *reconfigured* from a valid configuration S into a valid configuration T, if $|S| = |T| = n$, and there exist n paths $\{\pi_1, \pi_2, \ldots, \pi_n\}$, where each path $\pi_i : [0, 1] \to \mathbb{R}^2$ is a continuous curve, such that their starting points form the set S (i.e., $\bigcup_i \pi_i(0) = S$), their final points form the set T, (i.e., $\bigcup_i \pi_i(1) = T$), and at any moment in time $t \in [0, 1]$, the set of points $\{\pi_1(t), \pi_2(t), \ldots, \pi_n(t)\}$ forms a valid configuration for the given value of r.

Analogously, we say that k classes of robots can be *reconfigured* from a valid k-configuration $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ into a valid k-configuration $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$, if $|S_i| = |T_i|$ for all i, and there exist a set of paths which reconfigure each S_i into T_i, such that no two robots overlap at any moment in time.

Drawing inspiration from Hearn and Demaine [4] and Solovey and Halperin [7], we define a few variants of the k-DRMP problem.

Multi-to-multi k-DRMP Given k classes of robots and two valid k-configurations $\mathcal{S}$ and $\mathcal{T}$, decide whether the robots can be reconfigured from $\mathcal{S}$ to $\mathcal{T}$.

Multi-to-single k-DRMP Given k classes of robots, a valid k-configuration $\mathcal{S}$, and a target position $t \in P$, decide whether there exists a valid k-configuration $\mathcal{T}$ with $t \in T_i$ for some $T_i \in \mathcal{T}$, such that the robots can be reconfigured from $\mathcal{S}$ to $\mathcal{T}$.

Multi-to-single-in-class k-DRMP Given k classes of robots with distinct radii $\{r_1, \ldots, r_k\}$, some $1 \leq i \leq k$, a valid k-configuration $\mathcal{S}$, and a target position $t \in P$, decide whether there exists a valid k-configuration $\mathcal{T}$ with $t \in T_i$, where $T_i \in \mathcal{T}$ is the target configuration for the robots with radius r_i, such that the robots can be reconfigured from $\mathcal{S}$ to $\mathcal{T}$.

Intuitively, the *multi-to-multi* problem can be interpreted as follows: Let k-configurations $\mathcal{S}$ and $\mathcal{T}$ represent the start and target positions respectively for the k classes of robots. Can the robots move from $\mathcal{S}$ to $\mathcal{T}$ without any collisions? In this paper we will prove that this, and the other two variants of the k-DRMP problem, are PSPACE-hard. Note, that it is possible to define more variants of the k-DRMP problem by varying which starting or target positions might be fixed, possibly with a fixed matching on them, with specified robot radii, or in any other way along these lines. Many of them can be shown PSPACE-hard with a slight modification to our reduction.

2.1 Nondeterministic constraint logic

We will now briefly introduce the *nondeterministic constraint logic* (NCL). Hearn and Demaine [4] define an NCL machine as a weighted graph $G = (V, E)$, with nonnegative integer weights on the edges, and with integer *minimum in-flow* constraints on the nodes. A *state* of the NCL machine is an assignment of directions onto the edges of G. A state is *valid* if for every node the total weight of incoming edges is at least the value of the minimum in-flow constraint of that node.

Consider a valid state of the NCL machine, and some edge $e = (u, v) \in E$ directed from u to v. We can perform an *edge flip* by reassigning the orientation of e from v to u, as long as the state after the flip remains valid, that is, the in-flow constraint of u is still satisfied. The edge flip operation describes possible transitions between the states of the NCL machine.

Hearn and Demaine [4] show that, even for very restricted versions of the NCL machine, it is PSPACE-complete to decide whether there exists a sequence of valid edge flips which transforms one valid state into another. In particular, the PSPACE-completeness holds for the following four decision problems on an NCL machine which is (1) defined on a simple planar graph $G = (V, E)$, (2) has edge weights either 1 or 2, (3) has the minimum in-flow constraint 2 on all nodes, and (4) has nodes of types *AND* or *protected OR* (which we describe later). The decision problems are:

State-to-state Given two states σ_1 and σ_2, decide whether there exists a valid sequence of edge flips that transforms σ_1 into σ_2.

State-to-edge Given a state σ_1 and an edge $e \in E$, decide whether there exists a state σ_2 such that e has the opposite orientations in σ_1 and σ_2, and there exists a valid sequence of edge flips that transforms σ_1 into σ_2.

Edge-to-edge Given two edges e_1 and e_2 with specific orientations, decide whether there exist two states σ_1 and σ_2 with e_1 and e_2 of prescribed orientation respectively, and there exists a valid sequence of edge flips that transforms σ_1 into σ_2.

Figure 2 Two types of nodes. Edges with weight 1 are shown in red and edges with weight 2 are shown in blue. Left: An *AND* node. The minimum in-flow requirement of the vertex is 2. Edge *out* can be directed outwards only if both in_1 and in_2 are directed inwards, if the minimum in-flow constraint were to be maintained. Right: A *protected OR* node. Edge *out* can be directed outwards if either in_1 and in_2 is directed inwards. In a *protected OR* it is not possible for both in-edges to be directed inwards simultaneously.

Edge-to-state (symmetric to state-to-edge) Given an edge $e \in E$ and a state σ_2, decide whether there exists a state σ_1 such that e has the opposite orientations in σ_1 and σ_2, and there exists a valid sequence of edge flips that transforms σ_1 into σ_2.

The two types of nodes, the *AND* and the *protected OR*, are both degree three nodes with the following properties (refer to Figure 2). The *AND* node has two incident edges of weight 1, and one incident edge of weight 2. Thus, to satisfy the in-flow constraint, the weight-2 edge can be directed outwards only if both weight-1 edges are directed inwards. All incident edges of an *OR* node have weight 2. Thus, an edge can be directed outwards if at least one other incident edges is directed inwards. In a *protected OR* node two incident edges are labeled as *input*, and one as *output*. The "protected" property forbids the two input edges to be directed inwards at the same time. In many cases, including ours, this restriction simplifies reductions.

▶ **Theorem 1** (Theorem 11 [4]). *State-to-edge, edge-to-state, state-to-state, and edge-to-edge are PSPACE-complete, even when the constraint graph is simple, planar, and only has nodes of types AND and protected OR.*

3 From NCL to 2-DRMP

In this section we will prove that the k-DRMP problem is PSPACE-hard for k classes of unlabeled disc robots moving amidst obstacles constructed out of line segments and circular arcs, even if $k = 2$. We reduce from the NCL problem, for a given constraint graph G we construct a 2-DRMP instance that emulates the NCL machine built on G. Then, by considering the *state-to-edge* and *state-to-state* versions of the NCL problem we will show that the three variants of the 2-DRMP problem defined in Section 2 are PSPACE-hard.

Consider an instance of a constraint graph $G = (V, E)$ with *AND*- and *protected OR*-nodes, and consider its orthogonal drawing on a square grid, with the nodes having integer coordinates, and edges having at most one bend [2] (refer to Figure 3 for an example). We will construct an instance of the 2-DRMP problem such that it is equivalent to the NCL problem on G.

Similarly to the constructions in [3, 4, 7], we create a grid-like environment (refer to Figure 4): square cells of size 18×18 are separated with walls of width 1. The cells correspond to the nodes of G or to the edges passing through grid points. If there is an edge in G passing between two adjacent cells, there is an opening of width 2 in the middle of the wall

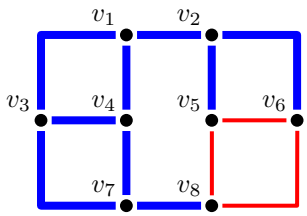


Figure 3 Orthogonal drawing of a constraint graph G. Edges with weight 1 are shown in red, while edges with weight 2 are shown in blue. Nodes v_5, v_6, and v_8 are *AND*-nodes, and nodes v_1, v_2, v_3, v_4 and v_7 are *protected OR*-nodes.

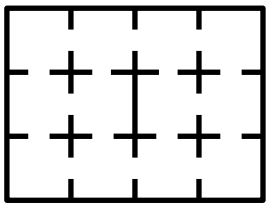


Figure 4 The grid-environment to be filled with gadgets, corresponding to graph G shown in Figure 3. Square cells either correspond to the nodes of G, or grid points that edges pass through.

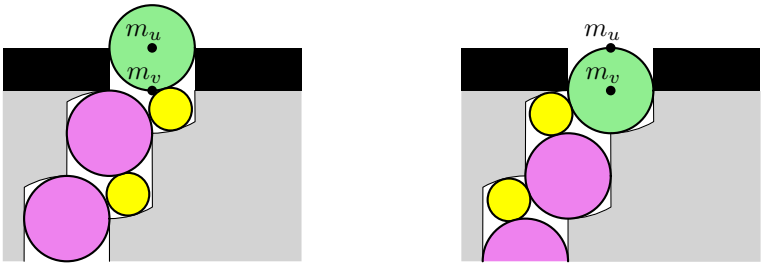


Figure 5 Two terminal positions of an edge robot. Left: edge directed from u to v. Right: edge directed from v to u.

separating the two cells. In each cell we construct free space (by filling its complement with obstacles) and densely place two classes of disc robots with radii 1/2 and 1 to emulate the nodes and the edges of G. The details of the three types of gadgets are described in the following section.

3.1 Gadgets

The three types of gadgets needed to emulate an NCL machine are *AND* gadgets, *protected OR* gadgets, and connector gadgets. Before we describe the specifics of each of the gadgets, we introduce a few building components which will appear in all of them.

Edge robot. Every gadget has two or three length-2 openings in the middle of the wall edges surrounding the square cell of the gadget. Next to each opening a radius-1 *edge* robot is placed (shown in green in the figures), which is shared by the two adjacent gadgets, and which may go back and forth through the opening. The construction of the gadgets is such that only the edge robots can enter and leave their corresponding gadgets. The rest of the robots, called *internal* robots, will always remain within their gadget. Edge robots are restricted in their movement by obstacles either as depicted in Figure 5 or by an obstacle as depicted in Figure 6. Consider an opening between two gadgets corresponding to some edge $(u, v) \in E$. Denote by m_u and m_v the two midpoints on the longer edges of the 1×2 rectangle forming the opening, respectively closer to the gadgets corresponding to u and v (refer to Figure 5). We will call these points *terminal* positions of the edge robot. With respect to a given gadget, we will distinguish between an *inside* terminal position and an *outside* terminal position of an edge robot. The terminal positions of the edge robot will correspond to the orientation of the edge (u, v) in G: the robot centered at m_u corresponds

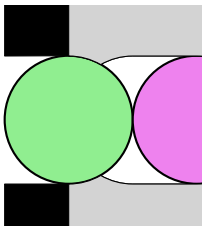


Figure 6 Nudges at an edge robot. The slight narrowing in the free space does not let the interior violet disc to travel more than a unit distance to the left.

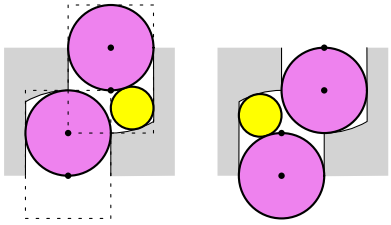


Figure 7 Two configurations of a parallel component.

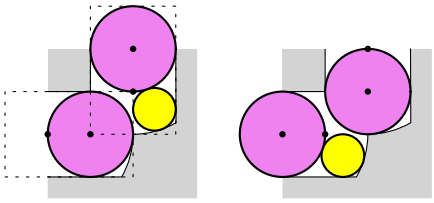


Figure 8 Two configurations of a perpendicular component.

to the edge directed from u to v (Figure 5 (left)), and the robot centered at m_v corresponds to the edge directed from v to u (Figure 5 (right)). Intermediate positions of the robot do not define a specific orientation of the edge, and may correspond to any direction. Thus, the positions of the edge robots, if they are all in terminal positions, will fully describe a state of an NCL machine.

Parallel and perpendicular components. The two constructions shown in Figures 7 and 8 are called a *parallel component* and a *perpendicular component* respectively. The parallel component consists of free space formed by two parallel 2×3 rectangles (outlined with a dashed line) overlapping in a corner 1×1 square. Place two radius-1 discs (violet in the figure) and a radius-(1/2) disc (yellow in the figure) in the resulting free space. For each 2×3 rectangle consider two points placed at a unit distance from three of the four sides of the rectangle. These points are the terminal positions for the two radius-1 discs. Finally, the short unit-length boundary edges of the free space are rounded (replaced with radius-2 arcs centered at the further terminal positions) so that the radius-(1/2) disc touches a radius-1 disc when moving into a corner of a rectangle.

The perpendicular component consists of two perpendicular 2×3 rectangles (in dashed) overlapping in a corner 1×1 square. Similarly, place two radius-1 discs and a radius-(1/2) disc in the free space, and consider the four terminal positions for the radius-1 discs. Again, the short unit length boundary edges are replaced with radius-2 arcs centered at the further terminal positions.

The parallel and perpendicular components are designed for propagating a signal of the position of an edge robot. In Section 3.2 we will show the following property of a chain of parallel and perpendicular configurations. Let $\mathcal{C}$ be a chain of at least 1 parallel and 1 perpendicular configuration, possibly extended with aligned radius-1 discs (for example, as the one used in the connector gadget shown in Figure 11 (right)). Let A^* be the first radius-1 robot in $\mathcal{C}$, and B^* be the last radius-1 robot in $\mathcal{C}$. Let t_1 and t_2 be the terminal positions of A^*, and t_3 and t_4 be the terminal positions of B^*, such that t_1, t_2, t_3, and t_4 appear in order along $\mathcal{C}$. Then the following lemma holds.

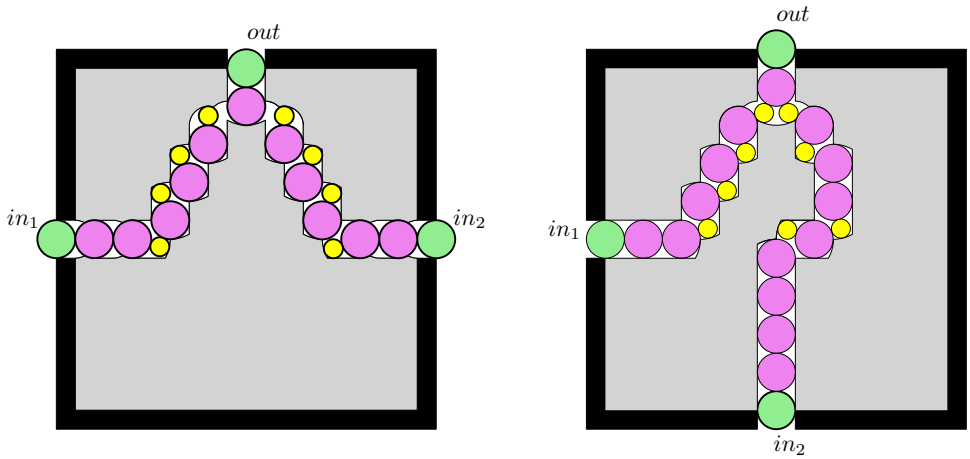


Figure 9 *AND*-gadgets representing *AND* nodes. Edge robot *out* can move inside if and only if both edge robots in_1 and in_2 are outside.

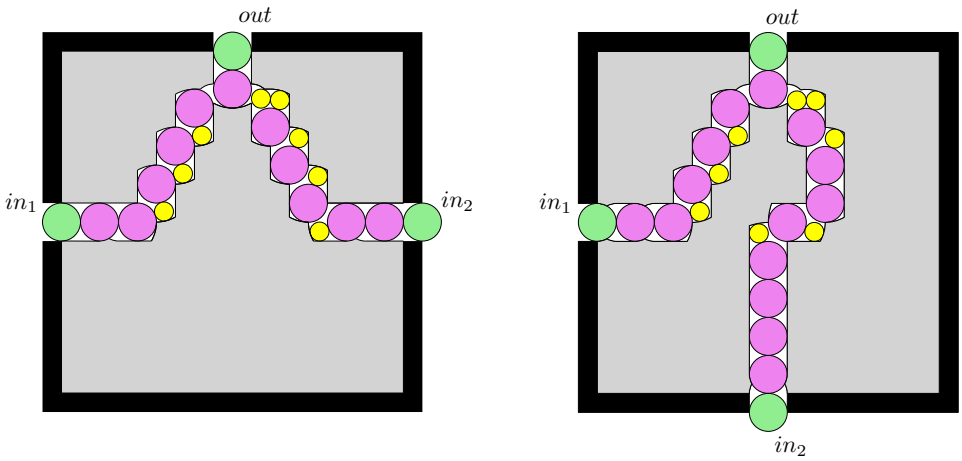


Figure 10 The *protected OR*-gadgets representing *protected OR* vertices. Robot *out* can move inside if and only if either in_1 or in_2 is outside. In this figure, in_2 has moved out allowing robot *out* to move in.

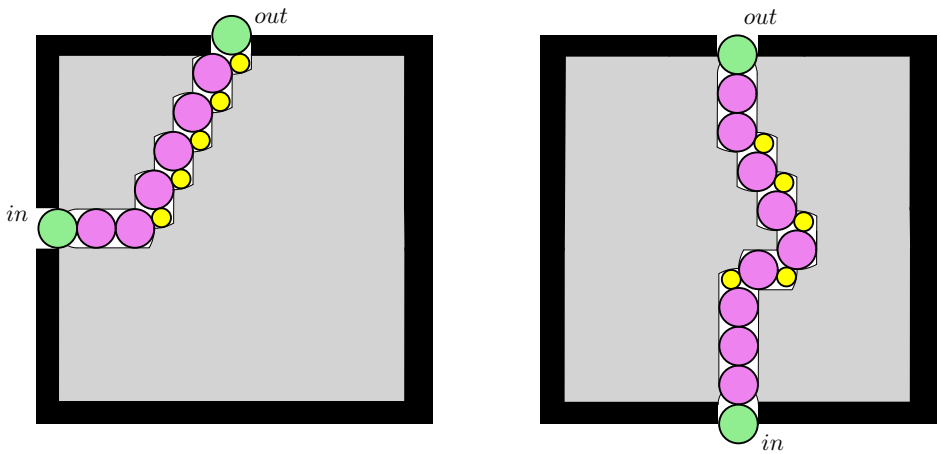


Figure 11 Connector gadgets representing connector vertices. Robot *out* can move inside only if robot *in* is outside and vice versa.

▶ **Lemma 2.** *Robot A^* can move to its terminal position t_2 only if B^* is in its terminal position t_4. Similarly, B^* can move to its terminal position t_3 only if A^* is in its terminal position t_1.*

Using this lemma we will use a sequence of at least 2 parallel and perpendicular components to force the edge robots to be located at one of their terminal positions. We are now ready to show the construction of the gadgets.

AND gadgets. There are two versions of the *AND* gadget (up to mirror symmetry and rotation by 90°, 180°, or 270°), shown in Figure 9. Also refer to Figure 17. They are designed in such a way that the edge robot *out* can only be moved to the inside terminal position if edge robots in_1 and in_2 are both moved to their outside terminal positions. Indeed, the edge robot *out* can move to the inside terminal position only if the two yellow discs of radius 1/2 below it move to the left and to the right, respectively. These yellow discs can move left and right only if both radius-1 violet discs touching them move down, which by Lemma 2 can happen only if both edge robots in_1 and in_2 move to their outside terminal positions. In Section 3.2 we will prove the following lemma.

▶ **Lemma 3.** *In the AND gadgets, the edge robot "out" can be moved to the inside terminal position only if the edge robots "in_1" and "in_2" are both moved to (or beyond) their outside terminal positions. The edge robot "in_1" ("in_2") can be moved to its inside terminal position only if the edge robot "out" is moved to (or beyond) its outside terminal position.*

Protected OR gadgets. Similarly to the *AND* gadgets, there are two versions of the *protected OR* gadget shown in Figure 10. Also refer to Figure 18. The *protected OR* gadgets are designed in such a way that the edge robot *out* can move to the inside terminal position of the gadget if and only if either in_1 or in_2 move to their outside terminal positions. Indeed, the edge disc *out* can move to the inside terminal position only if the yellow discs of radius 1/2 below it move to the left or to the right. These yellow discs can move left or right only if at least one of the adjacent radius-1 violet discs moves one unit down, which by Lemma 2 can happen only if both edge robots in_1 and in_2 move to their outside terminal positions. In Section 3.3 we will prove that the "protected" property of this gadget is preserved. That is, we will show a correspondence between a valid reconfiguration of the robots and a sequence of edge flips in the graph G, such that no two input edges of a *protected OR* in G are both directed inwards at the same time. In Section 3.2 we will prove the following lemma.

▶ **Lemma 4.** *In the protected OR gadgets, the edge robot "out" can be moved to the inside terminal position only if at least one of the edge robots "in_1" and "in_2" is moved to (or beyond) their outside terminal positions. The edge robot "in_1" ("in_2") can be moved to its inside terminal position only if the edge robot "out" is moved to (or beyond) its outside terminal position.*

Connector gadgets. Our last type of gadget, the connector gadget, is shown in Figure 11. The two versions of the connector gadget represent a corner piece of an edge (Figure 11 (left)) and a straight piece of an edge (Figure 11 (right)). The connector gadgets consist of a chain of parallel and/or perpendicular components, possibly extended by a sequence of aligned unit discs. The following lemma follows directly from Lemma 2.

▶ **Lemma 5.** *In the connector gadget, an edge robot can only be moved to its inside terminal position if the other edge robot is moved to (or beyond) its outside terminal position.*

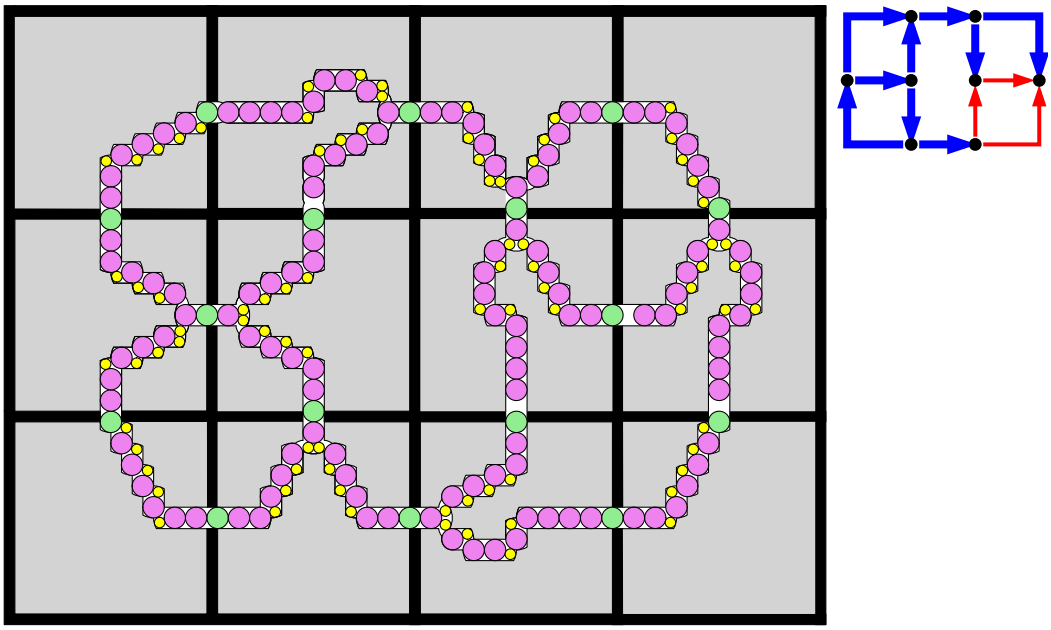


Figure 12 An instance of k-DRMP built for the NCL constraint graph G from Figure 3.

Using the three described types of gadgets, we now build a complete 2-DRMP instance corresponding to the NCL machine on a constraint graph G. We fill the cells of the grid-like environment, dual to an orthogonal drawing of G, with the *AND*, *protected OR*, and connector gadgets. Figure 12 shows an example of a 2-DRMP instance constructed for the constraint graph from Figure 3. The construction of the 2-DRMP instance, and Lemmas 3, 4, and 5, imply the main results of this paper, which we formally prove in the next section.

▶ **Theorem 6.** *The multi-to-multi, multi-to-single, and multi-to-single-in-class k-DRMP problems are PSPACE-hard for two classes of unlabeled disc robots moving amidst obstacles constructed out of line segments and circular arcs.*

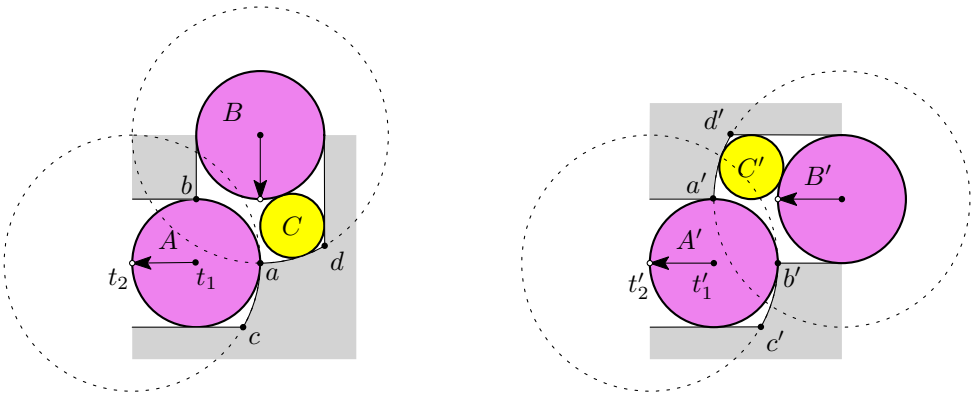
Finally, we argue that the circular arcs in the construction can be approximated with circumscribed polygonal chains without changing the validity of the reduction.

▶ **Theorem 7.** *The multi-to-multi, multi-to-single, and multi-to-single-in-class k-DRMP problems are PSPACE-hard for two classes of unlabeled disc robots moving in a polygonal environment.*

3.2 Correctness of the gadgets

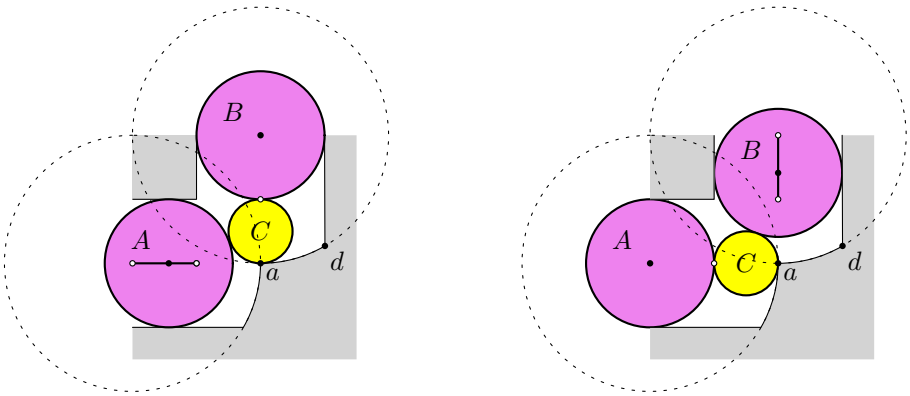
We now prove that the gadgets described in the previous section indeed correspond to their respective nodes in a constraint graph. Let us first take a closer look at the parallel and perpendicular components that make up our gadgets.

Properties of parallel and perpendicular components. Figure 13 shows a detailed design of the parallel (left) and perpendicular (right) components. Discs A and B have radius 1, disc C has radius 1/2, and the dashed circles have radius 2. These configurations can be mirrored and rotated by 90°, 180° and 270°. We will chain the components to enforce the terminal positions of the edge robots. The red arrows indicate movement of the unit discs in the sketched situation. The dashed circles indicate the circular segments of the boundary of the free space. The white dots indicate terminal positions of A and B.



(a) Perpendicular component. **(b)** Parallel component.

Figure 13 Detailed design of the perpendicular (a) and parallel (b) components which are present in every gadget. The two possible components of corner situations inside a gadget.



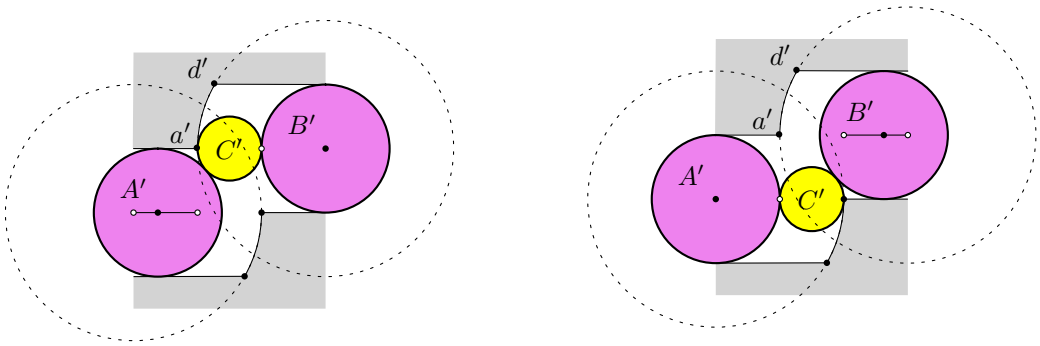
(a) Small disc on the corner and on the same axis as B. **(b)** Small disc on the corner and on the same axis as A.

Figure 14 Perpendicular movement with the discs shifted.

We first look at the definition of the perpendicular component as shown in Figure 13 (left). Let point a be at $(0,0)$. Then points $b = (-1,1)$, $c = (\sqrt{3}-2,-1)$, and $d = (1, 2-\sqrt{3})$. Points $t_1 = (-1,0)$ and $t_2 = (-2,0)$ are terminal positions of disc A. Indeed, by construction the center of disc A must lie on the closed segment t_1t_2. In the configuration of the discs depicted in the figure, disc A is centered at t_1, disc B is centered at $(0,2)$, and disc C is centered at $(0.5, 2-\sqrt{2})$. The circular segments ac and ad are arcs of 30° of radius-2 circles centered at t_2 and $(0,2)$ respectively.

Now we define the parallel component as shown in Figure 13 (right). Let point a' be at $(0,0)$. Then $b' = (1,-1)$, $c' = (\sqrt{3}-1,-2)$, and $d' = (2-\sqrt{3},1)$. Points $t'_1 = (0,-1)$ and $t'_2 = (-1,-1)$ are terminal positions of disc A'. In the example depicted in the figure, disc A' is centered at t'_1, disc B' is centered at $(2,0)$, and disc C' is centered at $(2-\sqrt{2}, 0.5)$. The circular segments $b'c'$ and $a'd'$ are arcs of 30° of the radius-2 circles centered at t'_2 and $(2,0)$ respectively.

▶ **Lemma 8.** *In the perpendicular component, disc B can not move down before disc A has moved by distance of at least $\sqrt{2}-1$ from t_1 towards t_2.*



(a) Small disc on the corner and on the same axis as B.

(b) Small disc on the corner and on the same axis as A

Figure 15 Perpendicular movement with the discs shifted.

Proof. Consider the perpendicular configuration depicted in Figure 13a. Since the arc ad belongs to a circle of radius 2, B can not move before the x-coordinate of the center of C is smaller than the x-coordinate of a. When the x-coordinate of the center of C is equal to the x-coordinate of a, B is still in its topmost position, but A must have moved by at least $\sqrt{2}-1$ from t_1 to prevent overlap with C (see Figure 14a). ◀

▶ **Lemma 9.** *In the parallel component, disc B' can not move left before disc A' has moved by a distance at least $\frac{\sqrt{5}}{2}-\frac{1}{2}$ from t'_1 towards t'_2.*

Proof. Consider the parallel configuration depicted in Figure 13b. Since the arc $a'd'$ belongs to a circle of radius 2, B' can not move before the y-coordinate of C' becomes smaller than the y-coordinate of a'. When the y-coordinate of C' is equal to the y-coordinate of a', B' has not been able to move yet, but A' must have moved by at least $\frac{\sqrt{5}}{2}-\frac{1}{2}$ to prevent overlap with C' (see Figure 15a). ◀

Proof of Lemma 2. The proof directly follows from Lemmas 8 and 9. Consider, as an example a chain of a perpendicular and a parallel component as depicted in Figure 16. By Lemma 9, disc B' can only move away from its terminal position t'_2 when the radius-(1/2) disc C' moves left beyond point a'. Thus, disc B has to move down by at least $\frac{\sqrt{5}}{2}-\frac{1}{2}$ before B' can move. However, if disc B moves $2-\sqrt{2}$ or more, by Lemma 8, A must be in the terminal position t_2. As $2-\sqrt{2}<\frac{\sqrt{5}}{2}-\frac{1}{2}$, we have that either disc B' must be in its terminal position t'_2, or disc A must be in its terminal position t_2. If the chain of parallel and perpendicular components is longer, the extremal positions of the radius-1 discs beyond either A or B' are enforced. Thus, if there is a perpendicular and a parallel component in the chain, the lemma holds. ◀

Correctness of connector gadgets. The correctness of the connector gadgets follows immediately from Lemma 2.

Proof of Lemma 5. As the connector gadgets consist of chains containing a parallel and a perpendicular component, by Lemma 2, the current lemma holds. ◀

Correctness of AND gadgets. The *AND* gadgets, as shown in Figure 9, use two small disc robots with a radius of 1/2. They have a specific layout of the obstacles, which enable the functionality of the gadgets. A close-up view of this layout can be seen in Figure 17. Disc

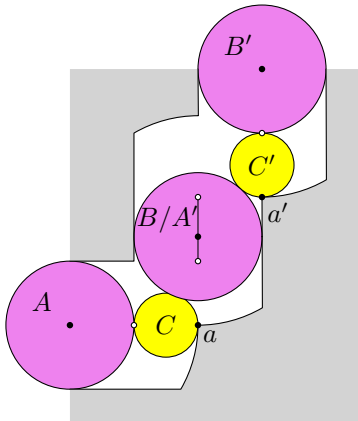


Figure 16 A parallel and a perpendicular component chained. If disc C is horizontally aligned with a, and disc C' is vertically aligned with a', then disc B must overlap either C or C'.

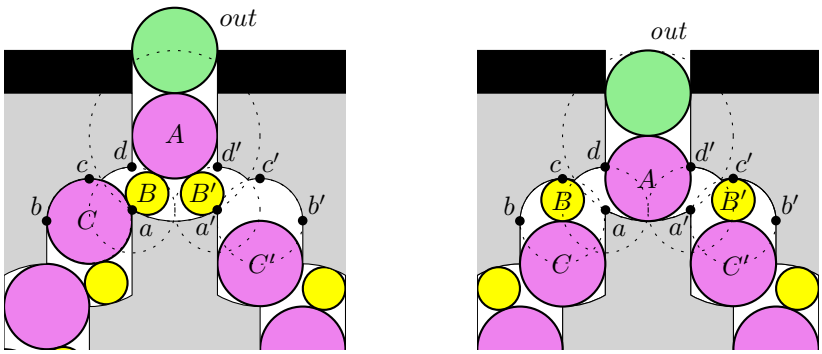


Figure 17 A close-up view of the functionality of the *AND* gadgets, the full gadgets can be seen in Figure 9.

robots B and B' can move along the arc $a'a$ which lies on a circle of radius 2. This means that A can not move as long as either disc B or disc B' is on the arc $a'a$. Arcs cd, $c'd'$, bc, and $b'c'$ lie on circles of radius 1. Thus discs C and C' tightly fit along the respective arcs bc and $b'c'$. Points c and c' are terminal position of the discs B and B'. Indeed, they cannot move further than c (or c'), otherwise they would overlap with disc C (or C'). Thus, discs B and B' cannot both fit in the free space above C or in the free space above C'. Therefore, both discs C and C' must move down to enable disc A to move down. If A is able to reach the arc aa', the discs B and B' should have x-coordinates at most $a - 1/2$ and at least $a' + 1/2$ respectively. Then, discs C and C' must move down by at least distance 0.8539. A similar argument to the one in Lemmas 8 and 9 will force the radius-1 discs to be moved to their terminal positions in the direction away from A.

Proof of Lemma 3. In Figure 17 the *AND* gadget is depicted in more detail. Since the construction is similar to the connector gadget, we know by Lemma 5 that discs C and C' from Figure 17 can only move when the input edge robots are in their terminal configurations. By the described construction above, disc A can only move if both C and C' have moved down. Since C and C' can only move when the inputs are in a terminal configuration, the edge robot *out* can only move when both edge robots in_1 and in_2 are in their outside terminal configuration. ◀

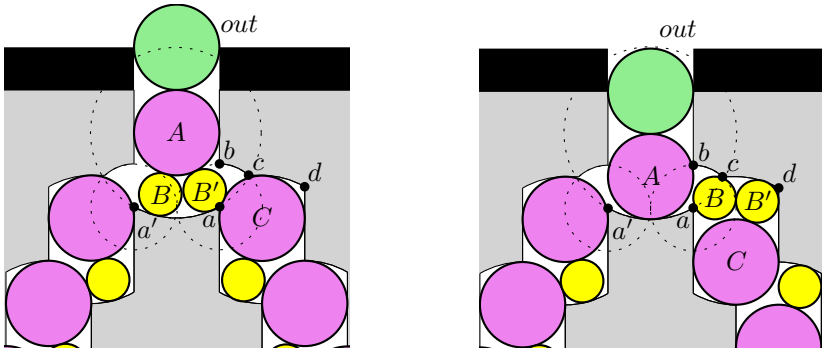


Figure 18 A close-up view of the functionality of the *protected OR* gadget, the full gadget can be seen in Figure 10.

Correctness of protected OR gadgets. The *protected OR* gadgets, as shown in Figure 10, use two small disc robots with a radius of 1/2, just like the *AND* gadgets. However, unlike in the *AND* gadgets, the *protected OR* gadgets have a different layout of the obstacles, which changes the functionality of the gadgets. A close-up view of this layout can be seen in Figure 18. Disc robots B and B' can move along the arc $a'a$ which lies on a circle of radius 2. As long as B or B' lies on this arc, disc A can not move. Disc C might move down, which makes space for discs B and B' to move above C. The arc bc lies on a circle of radius 1. Discs B and B' have a radius of 1/2, so they tightly fit along arc bc when C has made space. Since the arc cd lies on a circle of radius 2, both B and B' fit above C. When both B and B' have moved into one of the sides, A can move down such that the edge disc can move inside the gadget. In our construction we forbid the two robots B and B' to separate and move into the free space pockets above the two different radius-1 discs. We can do so because, as we will argue later, for any valid reconfiguration of robots that separates the discs, there will be an equivalent reconfiguration which keeps the discs always together.

Proof of Lemma 4. The proof directly follows from Lemma 2. ◀

Observe that, after putting all the gadgets together, all edge robots have a limited set of movements and that inner gadget robots remain in their gadgets.

▶ **Observation 10.** *Each edge robot can be in at most two distinct terminal configurations.*

3.3 Reduction

We are now ready to prove our main results.

▶ **Theorem 6.** *The multi-to-multi, multi-to-single, and multi-to-single-in-class k-DRMP problems are PSPACE-hard for two classes of unlabeled disc robots moving amidst obstacles constructed out of line segments and circular arcs.*

Proof. For a given NCL machine built on a constraint graph G, consider the corresponding instance of the k-DRMP problem constructed as described above. Furthermore, consider a state σ of the NCL machine, and a corresponding 2-configuration $\mathcal{S}$. The positions of the edge robots in $\mathcal{S}$ correspond to the orientation of the respective edges of G in σ. By Lemmas 3, 4, and 5, if an edge $e \in G$ cannot be flipped, the corresponding inner-gadget radius-1 robots are forced to be in one of their terminal positions, and cannot move. If, however, e can be flipped, some of the corresponding radius-1 robots are able to move between their terminal

positions (refer to Figure 12 for an example). By Lemmas 3, 4, and 5, a flip of the edge e is valid if and only if the corresponding edge robot can move to the opposite terminal position.

Consider the first problem, the *multi-to-multi* k-DRMP. We will show that it is PSPACE-hard by a reduction from the *state-to-state* NCL problem. Recall that the *state-to-state* NCL problem asks whether for a given constraint graph G and for two valid states σ_1 and σ_2 of the NCL machine, σ_1 can be transformed into σ_2 with edge-flip operations. From $\mathcal{S}$ and $\mathcal{T}$ we construct two 2-configurations $\mathcal{S}'$ and $\mathcal{T}'$, such that all the edge robots are in the same positions as in $\mathcal{S}$ and $\mathcal{T}$, all inner radius-1 and radius-(1/2) robots are shifted to their terminal positions consistent with the orientation of the corresponding edges in G. We claim that σ_1 can be transformed into σ_2 with edge-flip operations if and only if the robots can be reconfigured from $\mathcal{S}'$ to $\mathcal{T}'$.

Assume that there is a sequence of edge flips transforming σ_1 into σ_2. For each flip, by Lemmas 3, 4, and 5, we can reconfigure the robots of the k-DRMP instance in correspondence to the changes of orientations of the flipped edges.

It remains to show that if the robots of the k-DRMP instance can be reconfigured from $\mathcal{S}'$ to $\mathcal{T}'$, then there is a valid edge-flip sequence transforming σ_1 into σ_2. Consider the reconfiguration over time, and extract the order in which the edge robots reach one of their terminal configurations. If two edge robots are both in some intermediate positions between their terminal locations, then these edge robots can move independently from one another. We can modify the reconfiguration schedule such that at each moment in time only one edge robot can be located at an intermediate position between its terminal positions.

We still need to argue that we can preserve the "protected" property of the *protected OR* gadgets. Suppose that in the process of reconfiguration, at some moment, two input edge robots are moved to the outside terminal positions. If one of them, say in_1, reverts before the robot *out* moves to its inner terminal position, then we simply ignore the move of in_1 (and the robots in the chain from in_1 to *out*) outside. Let robot *out* move to the inner terminal position. Consider the positions of the discs B and B' (recall Figure 18). If they both are located above one radius-1 disc in the chain from in_1 to *out*, then we can change the schedule to stop in_2 from moving to the outside terminal position. If B and B' are separated, then we can modify the schedule to move B and B' into the same free-space pocket, and stop the other input edge robot from moving outside. In all cases, we can modify the schedule to prevent both input edge robots to be in their outside terminal positions. Thus, we have a reconfiguration schedule which preserves the "protected" property of the *protected OR* nodes, and has edge robots move between their terminal positions one at a time.

The order in which the edge robots move between their terminal positions gives the order of valid edge flips in G. Indeed, by Lemmas 3, 4, and 5, if an edge robot, corresponding to some edge $e = (u, v) \in G$, can move, the in-flow property of the two corresponding nodes u and v in G is satisfied by the edges other than e. Thus, the multi-to-multi k-DRMP problem is PSPACE-hard.

Now, consider the multi-to-single and multi-to-single-in-class versions of the k-DRMP problem. By a reduction from state-to-edge NCL problem, we show that these two problems are PSPACE-hard. The argument follows the same lines as for the multi-to-multi case, except that instead of the target 2-configuration $\mathcal{T}$, we are given a target location for a robot. We will select the edge robot corresponding to the edge to be flipped in the NCL problem, and specify a proper terminal positions as the target location for the robot. ◀

Remark. Note, that we can remove the use of circular arcs in our construction. Consider a small fixed value $\varepsilon > 0$. There exists a value $d(\varepsilon) > 0$, such that, if we replace the arcs in the construction with circumscribed polygonal chains with edge length at most d, the edge robots

will be bound to ε-neighborhoods of the terminal positions. Indeed, for small enough ε, Lemmas 3, 4, and 5 will still hold, with modified statements considering the ε-neighborhoods of the terminal positions instead of simply the terminal positions. Thus, the following result holds.

▶ **Theorem 7.** *The multi-to-multi, multi-to-single, and multi-to-single-in-class k-DRMP problems are PSPACE-hard for two classes of unlabeled disc robots moving in a polygonal environment.*

4 Conclusion

In this paper we have shown that the three variants of the disc-robot motion planning problem are PSPACE-hard, even for two classes of unlabeled disc robots with two different radii, moving in a polygonal environment. This is a first step towards settling the complexity of unlabeled unit disc robot motion planning. Our gadgets do not seem to generalize to a single class of robots. The complexity of unlabeled unit disc robot motion planning remains an interesting open problem.

References

1. 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. doi:10.1109/TASE.2015.2470096.
2. Tiziana Calamoneri and Rossella Petreschi. An efficient orthogonal grid drawing algorithm for cubic graphs. In *International Computing and Combinatorics Conference (COCOON)*, LNCS, volume 959, pages 31–40. 1995. doi:10.1007/BFb0030817.
3. Gary W. Flake and Eric B. Baum. Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1-2):895–911, 2002. doi:10.1016/S0304-3975(01)00173-6.
4. 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(1-2):72–96, 2005.
5. John E. Hopcroft, Jacob T. 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.
6. Kiril Solovey and Dan Halperin. k-Color multi-robot motion planning. *The International Journal of Robotics Research*, 33(1):82–97, 2014. doi:10.1177/0278364913506268.
7. 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.
8. 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. doi:10.15607/RSS.2015.XI.011.
9. Paul Spirakis and Chee K. Yap. Strong NP-hardness of moving many discs. *Information Processing Letters*, 19(1):55–59, 1984.
10. John Tromp and Rudi Cilibrasi. Limits of rush hour logic complexity. Manuscript, 2005. URL: http://arxiv.org/abs/cs/0502068.