On the Parameterized Complexity of Motion Planning for Rectangular Robots

Iyad Kanj
School of Computing, DePaul University, Chicago, IL, USA

Salman Parsa
School of Computing, DePaul University, Chicago, IL, USA

Abstract

We study computationally-hard fundamental motion planning problems where the goal is to translate k axis-aligned rectangular robots from their initial positions to their final positions without collision, and with the minimum number of translation moves. Our aim is to understand the interplay between the number of robots and the geometric complexity of the input instance measured by the input size, which is the number of bits needed to encode the coordinates of the rectangles' vertices. We focus on axis-aligned translations, and more generally, translations restricted to a given set of directions, and we study the two settings where the robots move in the free plane, and where they are confined to a bounding box. We also consider two modes of motion: serial and parallel. We obtain fixed-parameter tractable (FPT) algorithms parameterized by k for all the settings under consideration.

In the case where the robots move serially (i.e., one in each time step) and axis-aligned, we prove a structural result stating that every problem instance admits an optimal solution in which the moves are along a grid, whose size is a function of k, that can be defined based on the input instance. This structural result implies that the problem is fixed-parameter tractable parameterized by k.

We also consider the case in which the robots move in parallel (i.e., multiple robots can move during the same time step), and which falls under the category of Coordinated Motion Planning problems. Our techniques for the axis-aligned motion here differ from those for the case of serial motion. We employ a search tree approach and perform a careful examination of the relative geometric positions of the robots that allow us to reduce the problem to FPT-many Linear Programming instances, thus obtaining an FPT algorithm.

Finally, we show that, when the robots move in the free plane, the FPT results for the serial motion case carry over to the case where the translations are restricted to any given set of directions.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases motion planning of rectangular robots, coordinated motion planing of rectangular robots, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2024.65

Related Version *Full Version*: https://arxiv.org/abs/2402.17846

Funding *Iyad Kanj*: Supported by a Jarvis College of Computing and Digital Media Summer Stipend.

1 Introduction

1.1 Motivation

We study the parameterized complexity of computationally-hard fundamental motion planning problems where the goal is to translate k axis-aligned rectangular robots from their initial positions to their final positions without collision, and with the minimum number of translation moves. The parameter under consideration is the number k of robots, and the input length N is the number of bits needed to encode the coordinates of the vertices of the rectangles. We point out that, in our study, we deviate from using the real RAM model [25], which assumes that arithmetic operations over the reals can be performed in constant time, and use the Turing machine model instead. We believe that our study is more faithful to the geometric setting under consideration than the real RAM model. The input length N can be much larger than the number k of rectangles, and hence, our study of the parameterized complexity of the problem, which aims at investigating whether the problem admits algorithms whose running time is polynomially-dependent on the input size, is both meaningful and significant in such settings.

We study two settings where the robots move in the free plane, and where they are confined to a bounding box. We also consider two modes of motion: serial and parallel. Problems with the latter motion mode fall under the category of Coordinated Motion Planning problems. We point out that the problems under consideration have close connections to well-studied motion planning and reconfiguration problems, including the famous NP-complete $(n^2 - 1)$-puzzle [5, 18] and the PSPACE-hard warehouseman's problem [14] where the movement directions are limited, among many others. Moreover, the Coordinated Motion Planning for robots moving on a rectangular grid featured as the SoCG 2021 Challenge [11].

For most natural geometric (or continuous) motion planning problems, pertaining to the motion of well-defined geometric shapes in an environment with/without polygonal obstacles, the feasibility of an instance of the problem can be formulated as a statement in the first-order theory of the reals. Therefore, it is decidable using Tarski's method in time that is polynomially-dependent on the input length, and exponentially-dependent on the number of variables, number of polynomials and the highest degree over all the polynomials in the statement (see [19, 20, 21, 24]). When the parameter is the number k of robots, if an upper bound in k on the number of moves in a solution exists, then the existence of a solution can be decided in FPT-time using the above general machinery. However, this approach is non-constructive, and we might not be able to extract in FPT-time a solution to a feasible instance, as the only information we have about the solution is that it is algebraic.

There has been very little work on the parameterized complexity of these fundamental geometric motion planning problems, and our understanding of their parameterized complexity is lacking. Most of the early work (e.g., see [17, 22]) on such problems have resulted in algorithms for deciding *only* the feasibility of the instance and whose running time takes the form $\mathcal{O}(n^{f(k)})$, where n is the number of edges/walls composing the polygonal obstacles in the environment. Therefore, it is natural to investigate the parameterized complexity of the more practical variants of the problems, where one seeks a solution that meets a given upper bound on the number of robot moves or an optimal solution w.r.t. the number of robot moves, which remained unanswered by the earlier works.

The goal of this paper is to shed light on the parameterized complexity of these motion planning problems by considering the very natural setting of axis-aligned translations (i.e., horizontal and vertical), and more generally, translations restricted to a given (or a fixed) set of directions. We aim to understand the interplay between the number of robots and the complexity of the input instance (i.e., the input size). Our results settle the parameterized complexity of most of the studied problem variants by showing that they are FPT.

1.2 Related Work

There has been a lot of work, dating back to the 1980's, on the motion planning of geometric shapes (e.g., disks, rectangles, polygons) in the Euclidean plane (with or without obstacles), motivated by their applications in robotics. In this setting, robots may move along continuous curves. The problem is very hard, and most of the work focused on the feasibility of the problem for various shapes and environment settings (disks, rectangles, obstacle-free environment, environment with polygonal obstacles, etc.).

The early works by Schwartz and Sharir [22, 23, 24] showed that deciding the feasibility of an instance of the problem for two disks in a region bounded by n "walls" can be done in time $\mathcal{O}(n^3)$ [22]; they mentioned that their result can be generalized to any number, k, of disks to yield an $\mathcal{O}(n^{h(k)})$-time algorithm, for some function h of k. When studying feasibility, the moves can be assumed to be performed serially, and a move may extend over any Euclidean length. Ramanathan and Alagar [17] improved the result of Schwartz and Sharir [22] to $\mathcal{O}(n^k)$, conjecturing that this running time is asymptotically optimal. The feasibility of the coordinated motion planning of rectangular robots confined to a bounding box was shown to be PSPACE-hard [14, 15]. The problem of moving disks among polygonal obstacles in the plane was shown be strongly NP-hard [16]; when the shapes are unit squares, Solovey and Halprin [26] showed the problem to be PSPACE-hard.

Dumitrescu and Jiang [8] studied the problem of moving unit disks in an obstacle-free environment. They consider two types of moves: translation (i.e., a linear move) and sliding (i.e., a move along a continuous curve). In a single step, a unit disk may move any distance either along a line (translation) or a curve (sliding) provided that it does not collide with another disk. They showed that deciding whether the disks can reach their destinations within $r \in \mathbb{N}$ moves is NP-hard, for either of the two movement types. Constant-ratio approximation algorithms for the coordinated motion planning of unit disks in the plane, under some separation condition, where given in [4]. For further work on the motion planning of disks, we refer to the survey of Dumitrescu [7].

The problem of moving unit disks in the plane is related to the problem of reconfiguring/-moving coins, which has also been studied and shown to be NP-hard [1]. Moreover, there has been work on the continuous collision-free motion of a constant number of rectangles in the plane, from their initial positions to their final positions, with the goal of optimizing the total Euclidean covered length; we refer to [2, 10] for some of the most recent works on this topic.

Perhaps the most relevant, but orthogonal, work to ours, in the sense that it pertains to studying the parameterized complexity of translating rectangles, is the paper of Fernau et al. [12]. In [12], they considered a geometric variant of the PSPACE-complete Rush-Hour problem, which itself was shown to be PSPACE-complete [13]. In this variant, cars are represented by rectangles confined to a bounding box, and cars move serially. Each car can either move horizontally or vertically (or not move at all, i.e., is an obstacle), but never both during its whole motion; that is, each car slides on a horizontal track, or a vertical track. The goal is to navigate each car to its destination and a designated car to a designated rectangle in the box (whose corner coincides with the origin). They showed that the problem is FPT when parameterized by either the number of cars or the number of moves.

Finally, we mention that Eiben et al. [9] studied the parameterized complexity of Coordinated Motion Planning in the combinatorial setting where the robots move on a rectangular grid. They presented FPT algorithms, parameterized by the number of robots, for each of the two objective targets of minimizing the makespan and the total travel length [9].

1.3 Contributions

We present fixed-parameter algorithms parameterized by the number k of (rectangular axis-aligned) robots for most of the problem variants and settings under consideration.

(i) We give an FPT-algorithm for the axis-aligned serial motion in the free plane. Our proof relies on a structural result stating that every problem instance admits an optimal solution in which the moves are along a grid that can be defined based on the input instance. This structural result, combined with an upper bound of $4k$ that we prove on the number of moves in the solution to a feasible instance, implies that the problem is solvable in time $\mathcal{O}^*(k^{16k} \cdot 2^{20k^2+8k})$, and hence is FPT.

The structural result does not apply when the translations are not axis-aligned. To obtain FPT results for these cases, we employ a search-tree approach, and perform a careful examination of the relative geometric positions of the robots, that allow us to reduce the problem to FPT-many Linear Programming instances.

(ii) We show that the problem for serial motion in the free plane for any fixed-cardinality given set $\mathcal{V}$ of directions (i.e., part of the input) is solvable in time $\mathcal{O}^*((4^{32k} \cdot k \cdot |\mathcal{V}|)^{4k})$. A byproduct of this FPT algorithm is that the problem is in NP, a result that – up to the authors' knowledge – was not known nor is obvious. We complement this result by showing that the aforementioned problem for any fixed set $\mathcal{V}$ of directions that contains at least two nonparallel directions (which includes the case where the motion is axis-aligned) is NP-hard, thus concluding that the problem is NP-complete.

(iii) We give an FPT algorithm for the problem where the serial motion is axis-aligned and confined to a bounding box, which was shown to be PSPACE-hard in [14]. This result is obtained after proving an upper bound of $2k \cdot 5^{k(k-1)}$ on the number of moves in a feasible instance of the problem.
The approach used in (ii) and (iii) does not extend seamlessly to the case of coordinated motion (i.e., when robots move in parallel), as modelling collision in the case of parallel motion becomes more involved. Nevertheless, by a more careful enumeration and examination of the relative geometric positions of the robots, we give:

(iv) An FPT algorithm for the axis-aligned coordinated motion planing in the free plane that runs in $\mathcal{O}^*(5^{2k^3} \cdot 8^{4k^2})$ time, and an FPT algorithm for the axis-aligned coordinated motion planning confined to a bounding box that runs in time $\mathcal{O}^*(5^{k^2} \cdot 8^{2k^2 \cdot 5^{k^2}} \cdot 5^{k^3 \cdot 5^{k^2}})$. The FPT algorithm for the former problem implies its membership in NP.

2 Preliminaries and Problem Definition

We denote by $[k]$ the set $\{1, \ldots, k\}$. Let $\mathcal{R} = \{R_i \mid i \in [k]\}$ be a set of axis-aligned rectangular robots. For $R_i \in \mathcal{R}$, we denote by $x(R_i)$ and $y(R_i)$ the horizontal and vertical dimensions of R_i, respectively. We will refer to a robot by its identifying name (e.g., R_i), which determines its location in the schedule at any time step, even though, when it is clear from the context, we will identify the robot with the rectangle it represents/occupies at a certain time step.

A *translation move*, or a *move*, for a robot $R_i \in \mathcal{R}$ w.r.t. a direction $\overrightarrow{v}$, is a translation of R_i by a vector $\alpha \cdot \overrightarrow{v}$ for some $\alpha > 0$. For a vector $\vec{u}$, $\mathsf{translate}(R_i, \overrightarrow{u})$ denotes the axis-aligned rectangle resulting from the translation of R_i by vector $\overrightarrow{u}$. We denote by *axis-aligned motion* the translation motion with respect to the set of four directions $\mathcal{V} = \{\overrightarrow{H}^-, \overrightarrow{H}^+, \overrightarrow{V}^-, \overrightarrow{V}^+\}$, which are the negative and positive unit vectors of the x- and y-axis, respectively.

In this paper, we consider two types of moves: *serial* and *parallel*, where the former type corresponds to the robots moving one at a time (i.e., a robot must finish its move before the next starts), and the latter type corresponds to (possibly) multiple robots moving simultaneously. We now define collision for the two types of motion.

For a robot R_i that is translated by a vector $\overrightarrow{v}$, we say that R_i *collides* with a stationary robot $R_j \neq R_i$, if there exists $0 \leqslant x \leqslant 1$ such that R_j and $\mathsf{translate}(R_i, x \cdot \overrightarrow{v})$ intersect in their interior. For two distinct robots R_i and R_j that are simultaneously translated by vectors $\overrightarrow{v}$ and $\overrightarrow{u}$, respectively, we say that R_i and R_j *collide* if there exists $0 \leqslant x \leqslant 1$ such that $\mathsf{translate}(R_i, x \cdot \overrightarrow{v})$ and $\mathsf{translate}(R_j, x \cdot \overrightarrow{u})$ intersect in their interior.

We think of $\mathcal{R}$ as a set of axis-aligned rectangular robots, where each robot is given by the rectangle of its starting position and the congruent rectangle of its desired final position. We assume that the starting rectangles (resp. final destination rectangles) of the robots are pairwise non-overlapping (in their interiors). Let $\mathcal{V} = \{\overrightarrow{\theta_1}, \ldots, \overrightarrow{\theta_c}\}$, where $c \in \mathbb{N}$, be a set of unit vectors. We assume that if a vector $\overrightarrow{\theta_i}$ is in $\mathcal{V}$ then the vector $-\overrightarrow{\theta_i}$ is also in $\mathcal{V}$. For a vector $\vec{v}$, we denote by $x(\vec{v})$ and $y(\vec{v})$ the x-component/coordinate (i.e., projection of $\vec{v}$ on the x-axis) and y-component/coordinate of $\vec{v}$, respectively.

A *valid serial (*resp. *parallel) schedule* $\mathcal{S}$ for $\mathcal{R}$ w.r.t $\mathcal{V}$ is a sequence of collision-free serial (resp. parallel) moves, where each move is along a direction (resp. a set of directions) in $\mathcal{V}$, and after all the moves in $\mathcal{S}$, each R_i ends at its final destination, for $i \in [k]$. The *length* $|\mathcal{S}|$ of the schedule is the number of moves in it. In this paper, we study the following problem:

Rectangles Motion Planning (Rect-MP)
Given: A set of pairwise non-overlapping axis-aligned rectangular robots $\mathcal{R} = \{R_i \mid i \in [k]\}$ each given with its starting and final positions/rectangles; a set $\mathcal{V}$ of directions; $k, \ell \in \mathbb{N}$.
Question: Is there a valid schedule for $\mathcal{R}$ w.r.t. $\mathcal{V}$ of length at most ℓ?

We note that the time complexity for solving the above decision problem will be essentially the same (up to a polynomial factor) as that for solving its optimization version (where we seek to minimize ℓ), as we can binary-search for the length of an optimal schedule.

We also study The Rectangles Coordinated Motion Planning problem (Rect-CMP), which is defined analogously with the only difference that the moves could be performed in parallel. More specifically, the schedule of the robots consists of a sequence of moves, where in each move a subset S of robots move simultaneously, along (possibly different) directions from $\mathcal{V}$, at the same speed provided that no two robots in $\mathcal{R}$ collide. The move ends when all the robots in S reach their desired locations during that move; no new robots (i.e., not in S) can move during that time step.

We focus on the restrictions of Rect-MP and Rect-CMP to instances in which the translations are axis-aligned, but we also extend our results to the case where the directions are part of the input (or are fixed). We also consider both settings where the rectangles move freely in the plane, and where their motion is confined to a bounding box. For a problem $P \in \{\text{Rect-MP}, \text{Rect-CMP}\}$, denote by $+$-P the restriction of P to instances in which the translations are axis-aligned (i.e., $\mathcal{V} = \{\overrightarrow{H}^-, \overrightarrow{H}^+, \overrightarrow{V}^-, \overrightarrow{V}^+\}$), by P-$\square$ its restriction to instances in which the robots are confined to a bounding box (which we assume that it is given as part of the input instance), and by $+$-P-$\square$ the problem satisfying both constraints. For instance, $+$-Rect-MP-$\square$ denotes the problem in which the motion mode is serial, the translations are axis-aligned, and the movement is confined to a bounding box.

In parameterized complexity [3, 6], the running-time of an algorithm is studied with respect to a parameter $k \in \mathbb{N}$ and input size N. The most favorable complexity class is FPT (*fixed-parameter tractable*) which contains all problems that can be decided in time $f(k) \cdot N^{\mathcal{O}(1)}$, where f is a computable function. Algorithms with this running-time are called *fixed-parameter algorithms*. The $\mathcal{O}^*()$ notation hides a polynomial function in the input size N, which is the length of the binary encoding of the instance.

3 Upper Bounds on the Number of Moves

In this section, we prove upper bounds – w.r.t. the number k of robots – on the number of moves in an optimal schedule for feasible instances of several of the problems under consideration in this paper. These upper bounds are crucial for obtaining the FPT results.

3.1 Motion in the Free Plane

The upper bound in the case where the robots move in the free plane follow since, given at least two non-parallel directions, one can translate the rectangles, one by one, to very far and well-separated locations, and then reverse the process to bring them to their destinations:

▶ **Proposition 1.** *Let $\mathcal{I} = (\mathcal{R}, \mathcal{V}, k, \ell)$ be an instance of* Rect-MP *or* Rect-CMP*. If $\mathcal{V}$ contains two non-parallel directions, then there is a schedule for $\mathcal{I}$ of length at most $4k$.*

3.2 Axis-Aligned Motion in a Bounding Box

Let $\mathcal{I} = (\mathcal{R}, \mathcal{V}, k, \ell, \Gamma)$ be an instance of $+$-Rect-MP-$\square$. Fix an ordering on the vertices of any rectangle (say the clockwise ordering, starting always from the top left vertex). For any two robots R and R', the *relative order* of R w.r.t. R' is the order in which the vertices of R, when considered in the prescribed order, appear relatively to the vertices of R' (considered in the prescribed order as well), with respect to each of the x-axis and y-axis.

▶ **Definition 2.** *Fix an arbitrary ordering of the 2-sets of robots in $\mathcal{R}$. A* configuration *of $\mathcal{R}$ is a sequence indicating, for each 2-set $\{R, R'\}$ of robots in $\mathcal{R}$, considered in the prescribed order, the relative order of R with respect to R'. A* realization *of a configuration C is an embedding of the robots in $\mathcal{R}$ such that the relative order of any two robots in $\mathcal{R}$ conforms to that described by C and the robots in the embedding are pairwise non-overlapping.*

The following proposition shows that we can move between any two realizations of the same configuration using at most $2k$ translations:

▶ **Proposition 3.** *For any two realizations ρ, ρ' of a configuration C, there is a sequence of at most $2k$ valid moves within the bounding box Γ that translate the robots from their positions in ρ to their positions in ρ'.*

The above proposition implies that each configuration appears at most $2k$ times in an optimal schedule. By upper bounding the total number of distinct configurations, we get:

▶ **Proposition 4.** *Let $\mathcal{I} = (\mathcal{R}, \mathcal{V}, k, \ell, \Gamma)$ be a feasible instance of $+$-*Rect-MP-$\square$ *or $+$-*Rect-CMP-$\square$*. Then there is a schedule for $\mathcal{I}$ of length at most $2k \cdot 5^{k(k-1)}$.*

4 Axis-Aligned Motion

In this section, we prove a structural result about $+$-Rect-MP. This result, in particular, and the upper bound on the number of moves imply that $+$-Rect-MP is FPT parameterized by the number k of robots. In brief, the structural result states that, in order to obtain an optimal schedule to an instance of $+$-Rect-MP, it is enough to restrict the robots to move along the lines of an axis-aligned grid (i.e., a collection of horizontal and vertical lines of the plane), that can be determined from the input instance. Moreover, the number of lines in the grid is a computable function of the number of robots, and the robots' moves will be defined using intersections of the grid lines.

▶ **Definition 5.** *Let $\mathcal{I} = (\mathcal{R}, \mathcal{V}, k, \ell)$ be an instance of $+$-*Rect-MP*. We define an axis-aligned grid $G_{\mathcal{I}}$, associated with the instance $\mathcal{I}$, as follows.*

- *Initialize $G_{\mathcal{I}}$ to the set of horizontal and vertical lines through the starting and final positions of the centers of the robots in $\mathcal{R}$; call these lines the basic grid lines.*

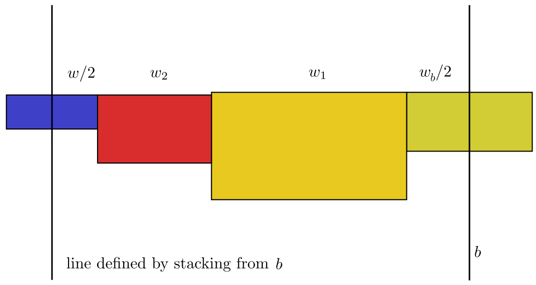

Figure 1 An illustration of a stacking to define new vertical lines.

- *Add to $G_{\mathcal{I}}$ the lines which are defined using "stackings" of robots on the basic lines as follows; see Figure 1. Let $b \in G_{\mathcal{I}}$ be a vertical basic line with x-coordinate $x(b)$, and w_b be the width of the robot whose center could be on b. For each number $1 \leqslant i \leqslant \ell$, and each i-multiset $\{R_{j_1}, \ldots, R_{j_i}\}$ of robots, and for each choice of a horizontal width w of a robot, add to $G_{\mathcal{I}}$ the two vertical lines with x-coordinates $x(b) \pm (w_b/2 + w/2 + \sum_{r=1}^{i} x(R_{j_r}))$.*
- *Add to $G_{\mathcal{I}}$ the analogous lines for the horizontal basic lines.*

▶ **Theorem 6.** *Every instance $\mathcal{I} = (G, \mathcal{R}, k, \ell)$ of* $+$-RECT-MP *has an optimal schedule in which every robot's move is between two grid points along a grid line in $G_{\mathcal{I}}$. The number of vertical (resp. horizontal) lines in $G_{\mathcal{I}}$ is at most $k^3 \cdot 2^{k+\ell+1}$.*

Proof. We argue by induction on the number ℓ' of moves in the schedule. If $\ell' = 1$, then the schedule has a single move that must be along a line defined by both the starting and ending positions of a robot in $\mathcal{R}$, and the statement is true in this case. Thus, assume henceforth that the statement of the theorem is true when the optimal schedule has at most $\ell' - 1$ moves. Let R be the robot that performs the first move (in the schedule) from some point p_1 to some point p_2, and assume, w.l.o.g., that the move is horizontal in the direction of $\overrightarrow{xx'}$. We define a new problem instance $\mathcal{I}'$, which is the same as $\mathcal{I}$, with the exceptions that in $\mathcal{I}'$ the robot R now has starting position p_2 and the upper bound on the number of moves is $\ell' - 1$. Let $G_{\mathcal{I}}$ and $G_{\mathcal{I}'}$ be the grids associated with instances $\mathcal{I}$ and $\mathcal{I}'$, respectively, as defined in Definition 5. The instance $\mathcal{I}'$ has a schedule of $\ell' - 1$ moves and hence, there is a schedule for $\mathcal{I}'$ such that each robot moves along a grid line in $G_{\mathcal{I}'}$.

The lines in the set $L_2 = G_{\mathcal{I}'} - G_{\mathcal{I}}$ are basic vertical grid lines defined by R being at p_2 plus all the lines defined by stackings of these lines. Note that L_2 contains only vertical lines and that $G_{\mathcal{I}'} - L_2 \subseteq G_{\mathcal{I}}$. Let G' be the set of grid lines in $G_{\mathcal{I}'} - L_2$ union the set of vertical lines obtained by stacking every robot on every line in $G_{\mathcal{I}'} - L_2$. Observe that $G' \subset G_{\mathcal{I}}$, and that we are allowed to perform this additional stacking operation since the construction of $G_{\mathcal{I}}$ involves ℓ' stackings, whereas the construction of $G'_{\mathcal{I}}$ involves $\ell' - 1$ stackings.

From among all schedules of length $\ell' - 1$ for $\mathcal{I}'$ along the grid lines $G_{\mathcal{I}'}$, consider a schedule that uses the maximum number of grid lines from G'.

Note that the move of R from p_1 to p_2 and the schedule for $\mathcal{I}'$ give an optimal schedule for the original problem instance $\mathcal{I}$; however, this schedule is not along the grid lines of $G_{\mathcal{I}}$, and some robots may move along the lines in L_2.

Let M be the set of segments of the grid lines in L_2 traversed by robot moves that are along the lines of L_2; note that all of these are vertical. Now we push back the robot R from p_2 towards p_1. Let $R(p)$ denote the robot R located at point $p \in p_1p_2$. The move from p_1 to p remains valid, however, there might be intersections between $R(p)$ and other robots in future moves, and between this move and future positions of R itself.

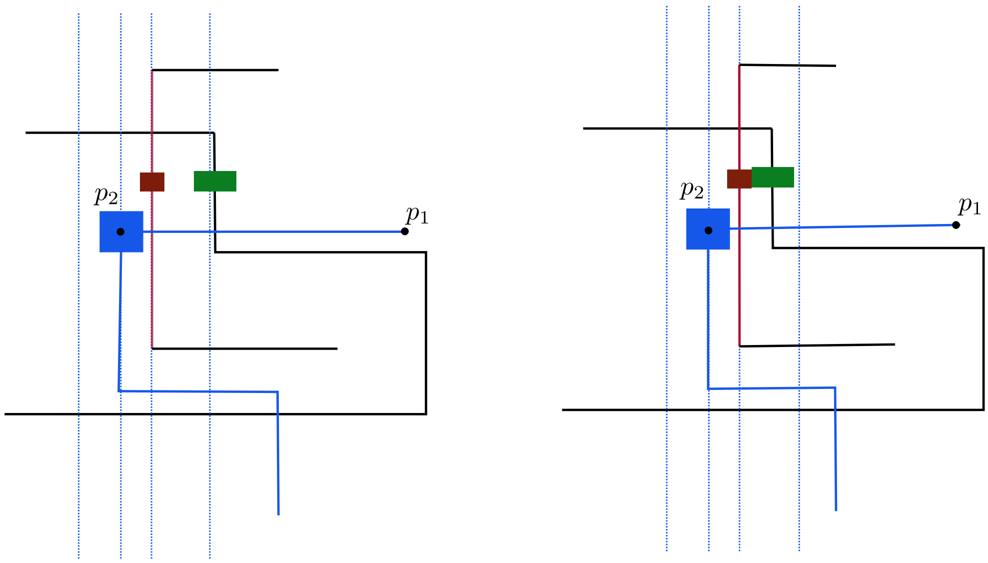

Figure 2 An illustration for the proof of Theorem 6. The blue broken line is the path traversed by the center of R in a schedule. The black broken lines are the paths for the centers of other robots. The vertical dashed blue lines are grid lines in L_2. The red segment is a move along a line in L_2. The blue rectangle is R after the first move, and the green and red rectangles are some other robots at some time during their movement. After moving the lines in L_2 to the right, the red segment becomes a stacking over lines not in L_2.

When $p = p_2$, we have a schedule, and thus no collisions exist. From the construction of the grid lines, it follows that when the robot R moves, the (updated) grid lines in L_2 move by exactly the same distance in the same direction. We now move all the segments of M also in the same direction and distance, i.e., we move the grid lines in L_2 together with all the robot moves along them; see Figure 2. As we push back R towards p_1, we stop the first time that the right edge of some robot, say Q, that travels along a segment in M, hits a vertical line that is defined by the left edge of a robot Q' located on a line in $G_{\mathcal{I}'} - L_2$, and hence cannot be pushed further without potentially introducing a collision. Now the center of Q is positioned at a line that is defined by some stacking of $G_{\mathcal{I}'} - L_2$, and hence is a vertical line of G'. Since we have not introduced any collisions during the pushing, we have obtained a grid schedule that uses more grid lines from G', which contradicts the maximality of the chosen schedule for $\mathcal{I}'$. Therefore, there is a schedule in which all moves are along G'.

Finally, we upper bound the number of lines in the grid. We upper bound the number of vertical lines; the upper bound on the number of horizontal lines is the same. Let $\mathcal{L}_V$ be the set of vertical lines in the grid, and $\mathcal{L}_H$ be that of the horizontal lines.

The number of starting and ending positions of the robots is $2k$, and hence the number of basic vertical lines is at most $2k$. For each i, where $1 \leqslant i \leqslant \ell$, and for each basic vertical line b, we fix two robots: the one of width w_b whose center could fall on b and the one of width w whose center could fall on the newly-defined vertical line based on the stacking. There are $\binom{k}{2} \leqslant k^2/2$ choices for these two robots. Afterwards, we enumerate each selection of an i-multiset of robots, and for each i-multiset $\{R_{j_1}, \ldots, R_{j_i}\}$, we add the two vertical lines with offset $(w_b/2 + w/2 + \sum_{k=1}^{i} x(R_{j_k}))$ to the left and right of b. Note that this offset is determined by the two fixed robots and the i-multiset of robots, and hence by the two fixed robots and the set of robots in this multiset together with the multiplicity of each robot in this set. Therefore, in step i, to add the vertical lines, we can enumerate every

2-set of robots, each r-subset of robots, where $r \leqslant i$, order the robots in the r-set, and for each of the at most 2^k r-subsets of robots, enumerate all partitions $(\alpha_1, \ldots, \alpha_r)$ of i into r parts. The number of such partitions is at most 2^i. Hence, the number of vertical lines we add in step i is at most $2k \cdot k^2/2 \cdot 2^k \cdot 2^i$, and the total number of vertical lines we add over all the ℓ steps is at most $k^3 \cdot 2^k \cdot \sum_{i=1}^{\ell} 2^i \leqslant k^2 \cdot 2^k \cdot 2^{\ell+1} = k^3 \cdot 2^{k+\ell+1}$. It follows that $|\mathcal{L}_V| \leqslant k^3 \cdot 2^{k+\ell+1}$ and $|\mathcal{L}_H| \leqslant k^3 \cdot 2^{k+\ell+1}$ as well. Therefore, the total number of lines in the grid is $\mathcal{O}(k^3 \cdot 2^{k+\ell})$. ◀

▶ **Theorem 7.** $\boxplus$-Rect-MP*, parameterized by the number of robots, can be solved in* $\mathcal{O}^*(k^{16k} \cdot 2^{20k^2+8k})$ *time, and hence is* FPT*.*

Proof. By the above theorem and by Proposition 1, the number of horizontal/vertical lines in the grid is at most $k^3 \cdot 2^{5k+1}$. The algorithm enumerates all potential schedules of length at most $\ell \leqslant 4k$ along the grid. The running time of the algorithm is $\mathcal{O}^*(k^{16k} \cdot 2^{20k^2+8k})$. ◀

The following result is also a byproduct of our structural result, since one can, in polynomial time, "guess" and "verify" a schedule of length at most ℓ to an instance of $\boxplus$-Rect-MP based on the grid corresponding to the instance:

▶ **Corollary 8.** $\boxplus$-Rect-MP *is in* NP*.*

The above corollary will be complemented with Theorem 15 in Section 7 to show that $\boxplus$-Rect-MP is NP-complete.

5 An FPT Algorithm When the Directions are Given

In this section, we give an FPT algorithm for the case of axis-aligned rectangles that serially translate along a given (i.e., part of the input) fixed-cardinality set of directions. We first start by discussing the case where the robots move in the free plane, and then explain how the algorithm extends to the case where the robots are confined to a bounding box.

Let $\mathcal{I} = (R, \mathcal{V}, k, \ell)$ be an instance of Rect-MP, where $\mathcal{R} = \{R_1, \ldots, R_k\}$ is a set of axis-aligned rectangular robots, and $\mathcal{V} = \{\overrightarrow{\theta_1}, \ldots, \overrightarrow{\theta_c}\}$, where $c \in \mathbb{N}$ is a constant, is a set of unit vectors; we assumed herein that c is a constant, but in fact, the results hold for any set of directions whose cardinality is a function of k. Let (s_i^1, s_i^2) be the coordinates of the initial position of the center of R_i and (t_i^1, t_i^2) be those of its final destination. We present a nondeterministic algorithm for the problem that makes a function of k many guesses. The purpose of doing so is two fold. First, it serves the purpose of proving that Rect-MP is in NP since the nondeterministic algorithm runs in polynomial time (assuming that $|\mathcal{V}|$ is a constant or polynomial in k). Second, it will render the presentation of the algorithm much simpler. We will then show in Theorem 9 how to make the algorithm deterministic by enumerating all possibilities for its nondeterministic guesses, and analyze its running time.

The algorithm consists of three main steps: (1) guess the order in which the k robots move in a schedule of length ℓ (if it exists); (2) guess the direction (i.e., the vector in $\mathcal{V}$) of each move; and (3) use Linear Programming (LP) to check the existence of corresponding amplitudes for the unit vectors associated with the ℓ moves that avoid collision.

We start by guessing the exact length, w.l.o.g. call it ℓ (since it is a number between 0 and ℓ), of the schedule sought. We then guess a sequence of ℓ events $\mathcal{E} = \langle e_1, \ldots, e_\ell \rangle$, where each event is a pair $(R_i, \overrightarrow{v_j})$, $i \in [k]$, $j \in [c]$, that corresponds to a move/translation of a robot $R_i \in \mathcal{R}$ along a vector $\overrightarrow{v_j} \in \mathcal{V}$ in the sought schedule. The remaining part of the algorithm is to check if there is a schedule of length ℓ that is "faithful" to the guessed sequence $\mathcal{E}$ of events. That is, a schedule in which the robots' moves, and the translation in each move, correspond to those in $\mathcal{E}$. To do so, we will resort to LP. Basically, we will rely on LP to give us the exact translation vector (i.e., the amplitude) in each event e_i, $i \in [\ell]$, while ensuring no collision, in case a schedule of length ℓ exists.

For each event $e_i = (R, \overrightarrow{v})$, $i \in [\ell]$, we introduce LP variables x_i, y_i to encode the coordinates (x_i, y_i) of the center of R at the beginning of the event. We also introduce an LP variable $\alpha_i > 0$ that encodes the amplitude of the translation of R in the direction $\overrightarrow{v}$ in e_i.

We form a set of LP instances such that the feasibility of one of them would produce the desired schedule, and hence, would imply a solution to instance $\mathcal{I}$. We explain next how this set of LP instances is formed. The LP constraints will stipulate the following conditions:

(i) Each robot ends at its final destination.

(ii) Each robot starts at its initial position, and the starting position of robot R_i in $e_q = (R_i, \overrightarrow{v})$ is the same as its final position after $e_p = (R_i, \overrightarrow{v'})$, where e_p is the previous event to e_q in $\mathcal{E}$ involving R_i (i.e., p is the largest index smaller than q).

(iii) The translation in each e_i is collision free.

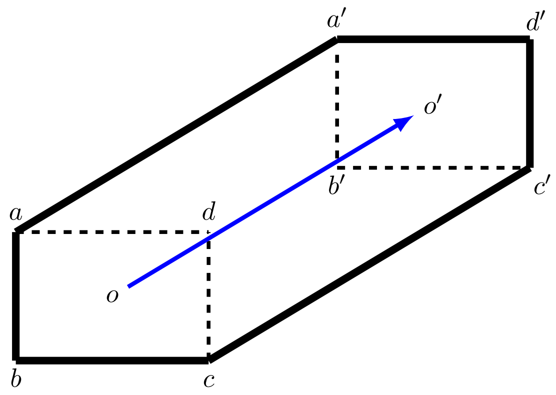

Figure 3 Illustration of the trace of a rectangle $abcd$ with respect to a vector $\overrightarrow{v} = \overrightarrow{oo'}$. Rectangle $a'b'c'd' = \mathsf{translate}(abcd, \overrightarrow{v})$ and the polygon $abcc'd'a'$, shown with solid lines, is $\mathsf{trace}(abcd, \overrightarrow{v})$. Observe that the edges of a trace are either edges of the rectangles, or are parallel to $\overrightarrow{v}$.

Conditions (i) and (ii) are easy to enforce using linear constraints. We discuss how the condition in (iii) can be enforced. For a robot R and a vector $\overrightarrow{v}$, denote by $trace(R, \overrightarrow{v})$ the boundary of the polygonal region of the plane covered during the translation of R by the vector $\overrightarrow{v}$; see Figure 3. It is clear that $\mathsf{trace}(R, \overrightarrow{v})$ is a polygon whose edges are either edges of R, or edges of $\mathsf{translate}(R, \overrightarrow{v})$, or line segments formed by a vertex of R and a vertex of $\mathsf{translate}(R, \overrightarrow{v})$ whose slope is equal to that of $\overrightarrow{v}$. Therefore, if R and $\overrightarrow{v}$ are fixed, then the slope of each edge of $\mathsf{trace}(R, \overrightarrow{v})$ is fixed (i.e., independent of the LP variables).

Now observe that robot $R_j \in \mathcal{R}$ does not collide with $R \in \mathcal{R}$ during the translation of R by a vector $\alpha \cdot \overrightarrow{v}$, where $\overrightarrow{v} \in \mathcal{V}$, if and only if no edge of R_j and an edge of $\mathsf{trace}(R, \alpha \cdot \overrightarrow{v})$ intersect in their interior. To stipulate that event $e_i = (R, \overrightarrow{v})$ is collision free, for each pair of edges (pq, rs), where pq is an edge of $\mathsf{trace}(R, \alpha \cdot \overrightarrow{v})$ and rs is an edge of R_j, we would like to add a linear constraint stipulating that the interiors of rs and pq do not intersect. If the slopes of the straight lines determined by pq and rs are the same, which we could check since the two slopes are given/fixed, then no such constraint is needed for this pair. Suppose now that the two straight lines (rs) and (pq) intersect at a point $\eta = (x_0, y_0)$; we add linear constraints to stipulate that point η does not lie in the interior of both segments rs and pq, and hence the two segments do not intersect. To do so, we guess (i.e., branch into) one of the following four cases. Let $r = (x_r, y_r), s = (x_s, y_s), p = (x_p, y_p), q = (x_q, y_q)$ and assume, without loss of generality, that $x_p \leqslant x_q$ and that $x_r \leqslant x_s$.

Case (1): Point η is exterior to pq and $x_0 \leqslant x_p$.
Case (2): Point η is exterior to pq and $x_0 \geqslant x_q$.
Case (3): Point η is exterior to rs and $x_0 \leqslant x_r$.
Case (4): Point η is exterior to rs and $x_0 \geqslant x_s$.

Note that pq and rs do not intersect in their interior if and only if (at least) one of the above cases holds. The algorithm guesses which case of the above four holds, and adds to the LP linear constraints stipulating the conditions of the guessed case. For instance, suppose that the algorithm guesses that Case (1) holds. Let β, γ be the slopes of lines (pq) and (rs), respectively, and note that β and γ are known/fixed at this point . It is easy to verify that $x_0 = (y_s - y_p + \beta x_p - \gamma x_s)/(\beta - \gamma)$. Therefore, to enforce the conditions in Case (1), we add to the LP the linear constraint:

$$(y_s - y_p) + \beta x_p - \gamma x_s \leqslant (\beta - \gamma) x_p.$$

For each event $e_i = (R, \overrightarrow{v})$, and for each robot $R_j \in \mathcal{R}$, where $R_j \neq R$, and for each pair of edges (pq, rs), where pq is an edge of $\mathsf{trace}(R, \alpha \cdot \overrightarrow{v})$ and rs is an edge of R_j, the algorithm guesses which case of the above four cases applies and adds the corresponding linear constraint. The algorithm then solves the resulting LP. If the LP has a solution, then so does the instance $\mathcal{I}$. If the LP is not feasible, then the algorithm rejects the instance.

▶ **Theorem 9.** *Given an instance $(\mathcal{R}, \mathcal{V}, k, \ell)$ of* Rect-MP*, in time $\mathcal{O}^*((4^{32k} \cdot k \cdot |\mathcal{V}|)^{4k})$, we can construct a solution to the instance or decide that no solution exists, and hence* Rect-MP *is* FPT*.*

▶ **Corollary 10.** Rect-MP *is in* NP*.*

Proof. The number of guesses made by the nondeterministic algorithm is polynomial. ◀

Next we discuss Rect-MP-$\square$, in which the robots are confined to a bounding box. In this case, the problem becomes PSPACE-hard as we observe in Section 7. It is easy to see that the LP part of the above approach can be easily modified to work for any rectangular bounding box by adding linear constraints stipulating that all rectangles resulting from the translations are confined to the box. (Basically, we only need to add constraints stipulating that the x/y-coordinate of each point are within the vertical/horizontal lines of the bounding box.) The only issue is upper bounding the number of moves, ℓ, in a feasible schedule.

For the case of axis-aligned motion, that is, $+$-Rect-MP-$\square$, Proposition 4 provides us with an upper bound of $2k \cdot 5^{k(k-1)} \leqslant 2k \cdot 5^{k^2}$ on ℓ in case the instance is feasible. Note that if the instance is not feasible, then the algorithm will end up rejecting the instance.

▶ **Theorem 11.** *Given an instance $(\mathcal{R}, \mathcal{V}, k, \ell, \Gamma)$ of $+$-*Rect-MP-$\square$*, in time $\mathcal{O}^*(5^{k^2} \cdot (4^{32k+1} \cdot k)^{2k \cdot 5^{k^2}})$, we can construct a solution to the instance or decide that no solution exists, and hence $+$-*Rect-MP-$\square$ *is* FPT*.*

6 An FPT Algorithm for $+$-Rect-CMP

In this section, we present an FPT algorithm for $+$-Rect-CMP. The only major challenge now is to stipulate non-collision in the case of parallel motion. We again first discuss $+$-Rect-CMP and then extend the FPT result to $+$-Rect-CMP-$\square$.

Let $(\mathcal{R} = \{R_1, \dots, R_k\}, \mathcal{V}, k, \ell)$ be an instance of $+$-Rect-CMP, where $\mathcal{V}$ is the set of unit vectors of the negative and positive x-axis and y-axis. We again present a nondeterministic algorithm for the problem, which will imply its membership in NP. The algorithm proceeds in a similar fashion to that of Rect-MP by guessing the exact number ℓ of moves in the sought schedule and then guessing the sequence $\mathcal{E} = \langle e_1, \dots, e_\ell \rangle$ of ℓ events corresponding to the schedule, with the exception that now each event – instead of containing the single robot that moves in that event and the direction of its translation – contains a subset of robots and their corresponding directions of translations in $\mathcal{V}$; that is, each event e_i is now a pair of the form (S_i, V_i), $i \in [\ell]$, where $S_i \subseteq \mathcal{R}$ and $V_i \subseteq \mathcal{V}$.

We again introduce, for each event $e_i = (S_i, V_i)$, and for each robot $R_j \in S_i$, variables x_j, y_j to encode the coordinates (x_j, y_j) of the center of R_j at the beginning of event e_i. We also introduce a variable $\alpha_j > 0$ that encodes the amplitude of the translation in the direction $\overrightarrow{v_j} \in V_i$ corresponding to the translation of R_j.

Consider a pair $R_p, R_q \in S_i$, and let $\overrightarrow{v_p}, \overrightarrow{v_q}$ be the directions of their translations in V_i, respectively, and let α_p, α_q be the LP variables corresponding to the amplitudes of $\overrightarrow{v_p}, \overrightarrow{v_q}$, respectively. We only discuss here how to encode in the LP that the translations of R_p and R_q in event e_i are collision-free and refer to the full version of the paper for the complete details. The algorithm makes several guesses to distinguish all possible cases. We focus here only on the two most involved cases arising when $\mathsf{trace}(R_p, \alpha_p \cdot \overrightarrow{v_p})$ and $\mathsf{trace}(R_q, \alpha_q \cdot \overrightarrow{v_q})$ intersect in their interior. We make guesses to distinguish the two (out of several) subcases below, and the conditions under which they apply; refer to Figure 4.

First Subcase: R_p is to the right of R_q when R_q reaches the horizontal line determined by the bottom-horizontal segment of R_p. We add the constraint:

$$(y_p - y(R_p)/2) - (y_q + y(R_q)/2) \geqslant x_q + x(R_q)/2 - (x_p - x(R_p)/2).$$

Second Subcase: R_q is above R_p when R_p reaches the vertical line determined by the left-vertical segment of R_q. We add the linear constraint:

$$(x_q - x(R_q)/2) - (x_p + x(R_p)/2) \geqslant y_p + y(R_p)/2 - (y_q - y(R_q)/2).$$

▶ **Theorem 12.** *Given an instance $(\mathcal{R}, \mathcal{V}, k, \ell)$ of $+$-Rect-CMP, in time $\mathcal{O}^*(5^{2k^3} \cdot 8^{4k^2})$, we can compute a solution to the instance or decide that no solution exists, and hence $+$-Rect-CMP is* FPT.

▶ **Corollary 13.** *$+$-Rect-CMP is in* NP.

We extend the above result to $+$-Rect-CMP-$\square$, which is PSPACE-hard (see Section 7):

▶ **Theorem 14.** *Given an instance $(\mathcal{R}, \mathcal{V}, k, \ell)$ of $+$-Rect-CMP-$\square$, in time $\mathcal{O}^*(5^{k^2} \cdot 8^{2k^2 \cdot 5^{k^2}} \cdot 5^{k^3 \cdot 5^{k^2}})$, we can compute a solution to the instance or determine that no solution exists, and hence $+$-Rect-CMP-$\square$ is* FPT.

7 Hardness Results

The $+$-Rect-MP-$\square$ is PSPACE-hard; this follows from the reduction of [14], since the rectangles in hard instances of [14] move horizontally or vertically. Also, an instance, when feasible, is feasible by a serial motion. Therefore, $+$-Rect-CMP-$\square$ is also PSPACE-hard. The restriction to axis-aligned motion actually makes the reduction in [14] simpler.

The following theorem shows that Rect-MP, restricted to instances in which the set $\mathcal{V}$ of directions is a fixed set containing at least two nonparallel directions, is NP-hard. From this and the results of Sections 4 and 5, it follows that the problem is NP-complete.

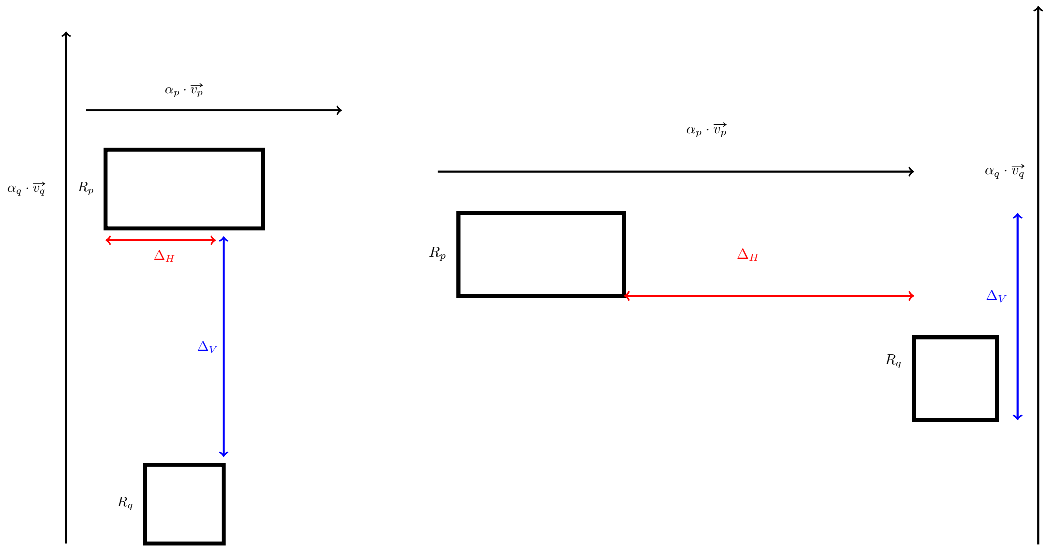

Figure 4 Illustration of the two subcases. In the left figure (first subcase), the distance Δ_V between the top edge of R_q and the bottom edge of R_p is larger than the distance Δ_H between the left edge of R_p and the right edge of R_q. Hence, R_p manages to "escape" R_q in time. Similarly, in the right figure (second subcase), R_q manages to escape R_p in time.

▶ **Theorem 15.** RECT-MP *restricted to the set of instances in which $\mathcal{V}$ is fixed and contains two non-parallel directions, is* NP*-complete.*

8 Concluding Remarks

We studied the complexity and developed parameterized algorithms for fundamental computational geometry problems pertaining to the motion planning of rectangular robots in the plane. Several follow-up questions ensue from this work:

1. What is the parameterized complexity of the problem variant in which there is no restriction on the translation directions? One possible approach to show FPT for this variant is to show that there is a computable set of possible positions for the robots that depends on the geometric complexity polynomially, and that transforms the continuous problem into a discrete one. We conjecture this to be true.
2. What is the parameterized complexity of the problem for other geometric shapes (e.g., congruent disks)?
3. What is the parameterized complexity of the problem for environments with obstacles?

References

1. Manuel Abellanas, Sergey Bereg, Ferran Hurtado, Alfredo García Olaverri, David Rappaport, and Javier Tejel. Moving coins. *Computational Geometry*, 34(1):35–48, 2006.
2. Pankaj K. Agarwal, Dan Halperin, Micha Sharir, and Alex Steiger. Near-optimal min-sum motion planning for two square robots in a polygonal environment. *arXiv preprint*, 2023. arXiv:2310.20615.
3. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

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 Erik D. Demaine and Mikhail Rudoy. A simple proof that the $(n^2 - 1)$-puzzle is hard. *Theoretical Computer Science*, 732:80–84, 2018.

6 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

7 Adrian Dumitrescu. Mover problems. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 185–211, New York, NY, 2013. Springer New York.

8 Adrian Dumitrescu and Minghui Jiang. On reconfiguration of disks in the plane and related problems. *Computational Geometry*, 46(3):191–202, 2013.

9 Eduard Eiben, Robert Ganian, and Iyad Kanj. The parameterized complexity of coordinated motion planning. In *Proceedings of the 39th International Symposium on Computational Geometry*, volume 258 of *LIPIcs*, pages 28:1–28:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

10 Guillermo Esteban, Dan Halperin, Víctor Ruíz, Vera Sacristán, and Rodrigo I Silveira. Shortest coordinated motion for square robots. In *Proceedings of the 18th International Symposium on Algorithms and Data Structures*, volume 14079 of *Lecture Notes in Computer Science*, pages 430–443. Springer, 2023.

11 Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing coordinated motion plans for robot swarms: The CG: SHOP challenge 2021. *ACM J. Exp. Algorithmics*, 27:3.1:1–3.1:12, 2022.

12 Henning Fernau, Torben Hagerup, Naomi Nishimura, Prabhakar Ragde, and Klaus Reinhardt. On the parameterized complexity of the generalized rush hour puzzle. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 6–9, 2003.

13 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.

14 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.

15 John E. Hopcroft and Gordon T. Wilfong. Reducing multiple object motion planning to graph searching. *SIAM Journal on Computing*, 15(3):768–785, 1986.

16 Daniel Kornhauser, Gary Miller, and Paul Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*, pages 241–250, 1984.

17 Geetha Ramanathan and Vangalur S. Alagar. Algorithmic motion planning in robotics: Coordinated motion of several disks amidst polygonal obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 514–522, 1985.

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

19 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I: Introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals. *Journal of symbolic computation*, 13(3):255–299, 1992.

20 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part II: The general decision problem. Preliminaries for quantifier elimination. *Journal of Symbolic Computation*, 13(3):301–327, 1992.

21 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part III: Quantifier elimination. *Journal of Symbolic Computation*, 13(3):329–352, 1992.

22 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*, 2:46–75, 1983.

23 Jacob T. Schwartz and Micha Sharir. On the "piano movers'" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on pure and applied mathematics*, 36(3):345–398, 1983.

24 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.

25 Michael Ian Shamos. *Computational geometry*. PhD thesis, Yale University, USA, 1978.

26 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.