

Collision-free Movement on Grids and Beyond

Hendrik Molter ✉ 

Chair for Algorithm Engineering, Hasso Plattner Institute, Potsdam, Germany

Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Meirav Zehavi ✉ 

Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

We study *collision-free movement problems* on graphs, where the task is to coordinate a set of robots so that they reach a target formation satisfying a desired property while minimizing the total travel distance. This framework extends two classical models: (a) *minimizing movement* [Demaine et al., TALG '09, '14], which does not enforce collision avoidance, and (b) *coordinated motion planning* or *multi-agent path finding* [Eiben et al., SoCG '23, Deligkas et al., ICALP '24, among many others], where each robot is assigned an explicit target position.

We focus on the setting where the target formation of the robots should be *connected*. We analyze the parameterized complexity of the problem with respect to the number of (main) robots and the total travel length on grid graphs and two natural generalizations thereof: planar graphs and unit disk graphs.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph algorithms

Keywords and phrases Coordinated Motion Planning, Multi-Agent Path Finding, Planar Graphs, Unit Disk Graphs, Clique Grids, Parameterized Complexity, Approximation, Kernelization

Funding *Hendrik Molter*: Supported by the Israel Science Foundation, grant nr. 1470/24, by the European Union's Horizon Europe research and innovation programme under grant agreement 949707, by the European Research Council, grant nr. 101039913 (PARAPATH), and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 565381415.

Meirav Zehavi: Supported by the Israel Science Foundation, grant nr. 1470/24, and by the European Research Council, grant nr. 101039913 (PARAPATH).

Acknowledgements Main work was done while Hendrik Molter was affiliated with the Ben-Gurion University of the Negev.

1 Introduction

Coordinated motion of robots is a fundamental task in computational geometry and robotics. The environment is frequently modeled by a grid, often consisting also of “obnoxious robots” in addition to the “main robots”. Within the environment, the main robots are set to move so as to perform a predetermined task; the obnoxious robots, on the other hand, are simply meant to model movable obstacles. To date, most efforts were directed at the following:

1. Given a target position for each main robot, plan a *collision-free* motion for all robots so that each main robot will reach its target (see, e.g., [10, 16, 21, 36, 39, 41, 42]). This setting, widely known as *MAPF (Multi-Agent Path Finding)*, was part of the 3rd Computational Geometry Challenge of SoCG 2021 [18].
2. Given a formation for all main robots (where a formation of particular interest is to induce a connected set), plan a motion (*collisions allowed*) of all robots as to achieve this formation (see, e.g., [3, 11, 12, 26]).

Both objectives have been extensively studied in a variety of discrete and continuous settings, from both a theoretical perspective—particularly in the frameworks of parameterized complexity [10, 12, 16, 17, 21] and approximation algorithms [3, 11, 36]—and a practical perspective [18, 40]. The above leaves open a very obvious and natural question:

What about the task of, given a characterization of a formation for all main robots, planning a *collision-free* motion of all robots as to achieve this formation?

To the best of our knowledge, the present work is the first to address this problem. As in the classical settings discussed above, two natural optimization goals arise: (i) minimize the total number of moves taken by all robots, i.e., the *total energy*; (ii) minimize the number of time steps needed, i.e., the *makespan*. Here, we focus on the first goal, while the desired formation is to establish a *connected set*. We provide a spectrum of parameterized results for grids and generalizations thereof, with respect to the two most natural parameters in this context: the number of main robots (k), and the total energy to spend (c). Notably, our results delineate the subtle boundaries between various tractable and intractable cases.

Previous works most closely related to ours—i.e., which focus on parameterized complexity, environments modeled by graphs (particularly grids), and total energy minimization, are Eiben et al. [16] and Deligkas et al. [10] for the first task, and Demaine et al. [12] for the second task. Discussion (non-exhaustive) of some other related works can be found at the end of this section.

Eiben et al. [16] studied MAPF on solid grids with both (total) energy and makespan minimization as objectives, and without obnoxious robots. This generalizes the well-known $(n^2 - 1)$ -Puzzle, which was extensively studied before and is known to be NP-hard [13, 38]. They showed (for solid grids only) that both energy and makespan minimization problems are fixed-parameter tractable (FPT) w.r.t. the number of (main) robots k , the energy minimization problem is FPT w.r.t. the energy c , and the makespan minimization problem is paraNP-hard w.r.t. the makespan. The follow-up work by Deligkas et al. [10] focused on (total) energy minimization, in the presence of obnoxious robots (thereby closely related to the Rush Hour problem, which was also extensively studied before [20, 22]). Further, they considered general graphs, and showed: FPT w.r.t. the number of obnoxious robots ℓ when $k = 1$; FPT w.r.t. the total number of robots $k + \ell$ plus the treewidth of the input graph; FPT approximation w.r.t. $k + \ell$ and additive error; W[1]-hardness w.r.t. c ; FPT w.r.t. c on

planar graphs. In further follow-up, Deligkas et al. [9] show that the problem is $W[1]$ -hard w.r.t. c even if $k = 1$ and FPT w.r.t. k plus the treedepth of the input graph. It remains open whether the problem is FPT w.r.t. $k + \ell$.

Demaine et al. [12] studied the parameterized complexity of a very general problem addressing movement to specific formations and for several objectives, including both main and obnoxious robots. Here, recall, collisions are allowed (also in the final formation itself). They provided a dichotomy that asserted that, when the family of possible formations, $\mathcal{F}$, is decidable and closed under edge-addition, the problem is FPT w.r.t. k if the edge-minimal formations in $\mathcal{F}$ have bounded treewidth, and $W[1]$ -hard otherwise. In particular, this asserts that the problem where we aim to move robots so that the main robots will form a connected set, and with the objective of energy minimization, is FPT w.r.t. k . We remark that this case (of connectivity) has received previous focused attention, particularly from the viewpoint of approximation (see, e.g., [3]). Additionally, Demaine et al. [12] provided $W[1]$ -hardness w.r.t. k for the setting where the formation is a hereditary property that does not contain all cliques or all independent sets.

1.1 Our Contributions

We refer to the problem introduced here as **COLLISION-FREE MOVEMENT TO CONNECTIVITY**. If results hold for more general formations we call the problem **COLLISION-FREE MOVEMENT TO Π** . All formal definitions are given in Section 3. Recall that k is the number of main robots, ℓ is the number of obnoxious robots, and c is the (total) energy bound. We start by discussing some results that are easy to obtain or follow straightforwardly from the literature [11, 12] in Section 3. Notably, we have the following.

- If $\ell = 0$, then **COLLISION-FREE MOVEMENT TO CONNECTIVITY** is fixed-parameter tractable when parameterized by k .
- If Π is hereditary and does not contain all complete graphs and all empty graphs, then **COLLISION-FREE MOVEMENT TO Π** is $W[1]$ -hard when parameterized by $k + c$ even if $\ell = 0$.
- **COLLISION-FREE MOVEMENT TO CONNECTIVITY** is NP-hard even if $\ell = 0$ and the input graph is planar.
- If deciding whether a graph has property Π is polynomial-time solvable, then **COLLISION-FREE MOVEMENT TO Π** is in XP when parameterized by c .

In the following, we give a brief description of each of our results and point to the sections where we present the full proofs. In Section 2 we give a more detailed overview on our results and explain the techniques we use to obtain them (deferring the proofs to the later sections).

Note that since connectivity is not hereditary, the $W[1]$ -hardness result for parameter $k + c$ mentioned above does not apply to **COLLISION-FREE MOVEMENT TO CONNECTIVITY**. In Section 4, show the following as our first hardness result.

► **Theorem 1.** ***COLLISION-FREE MOVEMENT TO CONNECTIVITY** is $W[1]$ -hard when parameterized by $k + c$.*

It follows that if we wish to obtain fixed-parameter tractability results for k , c , or $k + c$, we have to restrict the class of input graphs. We focus our investigation on grid graphs, and two natural generalizations: planar graphs and unit disk graphs.

Our main technical hardness result for grids is the following (Section 4).

► **Theorem 2.** ***COLLISION-FREE MOVEMENT TO CONNECTIVITY** is NP-hard even if $k = 2$ and the input graph is a grid.*

That is, it is hard for just two robots to meet on a grid! We remark that in both hardness results, the instances produced by the reductions have an unbounded number of obnoxious robots, which is not completely unexpected since COLLISION-FREE MOVEMENT TO CONNECTIVITY is fixed-parameter tractable when parameterized by k if there are no obnoxious robots.

On the positive side, we provide the following kernelization results in Section 5.

► **Theorem 3.** *COLLISION-FREE MOVEMENT TO Π parameterized by $k+c$ admits a polynomial kernel of size $O(k \cdot c^2)$ if the input graph is a grid.*

For the case where Π is connectivity, we can give a better bound on the kernel size.

► **Theorem 4.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by $k+c$ admits a polynomial kernel of size $O(\min\{k \cdot c^2, k^2 + c^2\})$ if the input graph is a grid.*

Furthermore, we give a number of kernelization lower bounds in Section 5. The constructions used are all based on ideas from the reduction behind Theorem 2. We first show that we presumably cannot obtain polynomial kernels for the parameter $k+c$ on planar graphs. Hence, exploiting the properties of grids is essential for the kernelization algorithms.

► **Theorem 5.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by c does not admit a polynomial kernel unless $\text{coNP} \subseteq \text{NP/poly}$, even if the input graph is planar and $k=2$.*

Further, we show that the dependency of the polynomial kernel size on k for grids presumably is unavoidable.

► **Theorem 6.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by c does not admit a polynomial kernel unless $\text{coNP} \subseteq \text{NP/poly}$, even if the input graph is a grid.*

Finally, we show that we presumably cannot significantly improve the kernel size given in Theorem 4.

► **Theorem 7.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by $k+c$ does not admit a polynomial kernel of size $O((k+c)^{2-\varepsilon})$ for any $\varepsilon > 0$ unless $\text{coNP} \subseteq \text{NP/poly}$, even if the input graph is a grid.*

In Section 6, we turn our attention to planar graphs and show the following result.

► **Theorem 8.** *COLLISION-FREE MOVEMENT TO Π is fixed-parameter tractable when parameterized by $k+c$ if the input graph is planar.*

We obtain this result by providing a monadic second-order logic (MSO) formulation of the problem and then employing Courcelle's famous theorem [2, 5, 6].

In Section 7, we show that COLLISION-FREE MOVEMENT TO CONNECTIVITY on unit disk graphs admits an FPT approximation algorithm when parameterized by $k+c$ for the canonical optimization variant of the problem where we aim to minimize the energy.

► **Theorem 9.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY admits a fixed-parameter approximation when parameterized by $k+c$ if the input graph is a unit disk graph, that produces a solution with energy at most $2 \cdot \text{OPT} + 3k$, where OPT is the energy of an optimal solution.*

Finally, we give some open questions and future research directions in Section 8.

1.2 Further Related Work

In recent years there has been intensive research on multi-agent path finding (MAPF) problems in multiple variations [39, 41, 42]. As mentioned, MAPF can be seen as a generalization of the $(n^2 - 1)$ -Puzzle and other so-called *pebble motion problems* on graphs [13, 28, 38, 43]. It is also closely related to the problem such as Rush Hour [20, 22], Temporally Disjoint Paths [34, 35], the Snake Game [29], and Chip Reconfiguration [7]. To the best of our knowledge, the current state-of-the-art (optimal) algorithms for MAPF problems employ the so-called conflict-based search approach [40].

The following works are more closely related to our problem. Papadimitriou et al. [36] study coordinated motion planning there is only one main robot and show that the problem is NP-hard on planar graphs. They give some further approximation results. Fioravantes et al. [21] study coordinated motion planning where the goal is to minimize the makespan rather than the total energy. They obtain an extensive number of parameterized complexity results. Eiben et al. [17] investigate the parameterized complexity a problem variant, where robots may move an arbitrary distance in each step and the number of step shall be minimized. There is also work on the parameterized complexity of coordinated motion planning problems where the robots have geometric shapes [1, 31].

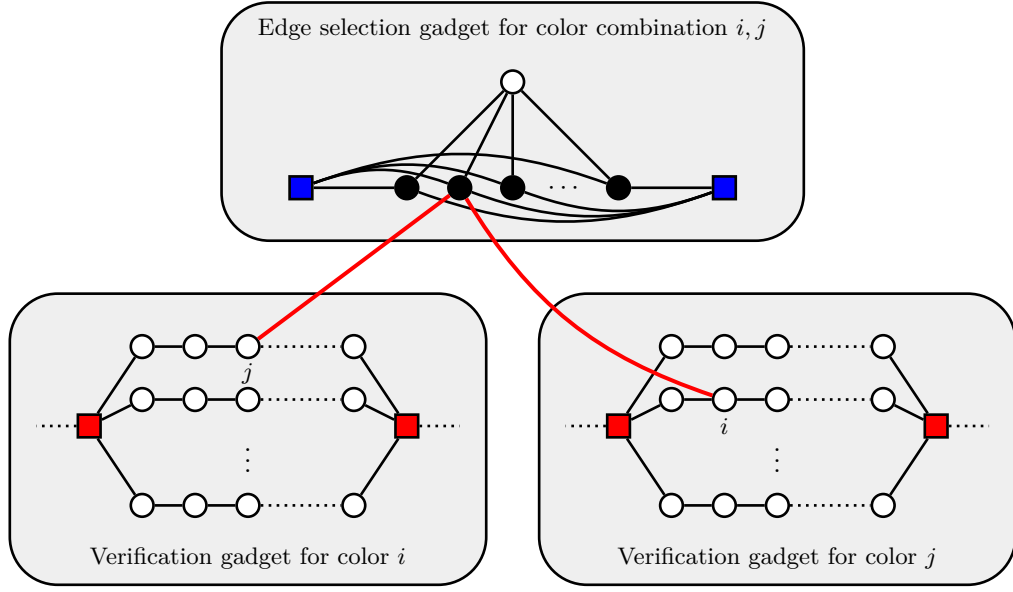
Various version of the movement problem (where collisions do not matter) have been studied by Demaine et al. [11] from an approximation perspective. Improved approximation results have been given by Berman et al. [3]. Friggstad and Salavatipour [26] study a closely related problem: Mobile Facility Location. Here, facilities and clients should be moved in a graph such that every client is close to a facility. They study approximability of this problem.

2 Overview of Results and Techniques

In this section, we present a more detailed overview on our results, together with intuitive and informal explanations and descriptions of the techniques that we use. Full details and formal proofs can be found in the later sections.

Hardness Results (Section 4). We first show that COLLISION-FREE MOVEMENT TO CONNECTIVITY is W[1]-hard when parameterized by $k + c$ (Theorem 1). To this end, we present a parameterized reduction from MULTICOLORED CLIQUE [19]. Here, given an ℓ -partite graph, we are asked whether it contains a clique of size ℓ . The parameter is ℓ . The reduction uses an edge-selection gadget for every color-combination. Each one has two main robots and one obnoxious robot for each edge. One of the obnoxious robots may be moved (the corresponding edge is “selected”) which enables the two main robots to move to verification gadgets for the colors in the color-combination. Each verification gadget of a color contains “immovable” formations of main robots which can only be connected if the selected edges of the color-combinations containing the color have the same endpoints. For an illustration see Figure 1.

Our main hardness result is that COLLISION-FREE MOVEMENT TO CONNECTIVITY is NP-hard on a grid even if there are only two main robots (Theorem 2). We obtain the result via a reduction from LINKED PLANAR 3-SAT [37]. This reduction is described in two construction steps: We first construct a planar instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY and then show how the produced graph can be embedded into a grid (with appropriate modifications). In LINKED PLANAR 3-SAT, we are given a Boolean formula ϕ in 3-CNF with clause set Y and variable set X , and a graph $H = (Y \cup X, F)$ such that the following holds:

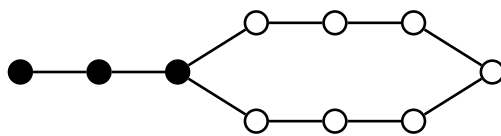


■ **Figure 1** Illustration of the graph $W[1]$ -hardness reduction. The black filled round vertices have obnoxious robots on them, and the blue filled square vertices have main robots on them. The red filled square vertices have main robots on them and paths of sufficient length attached to them that have main robots on each vertex (which are not depicted). The two red edges correspond to connections between the edge selecting gadget and the verification gadgets.

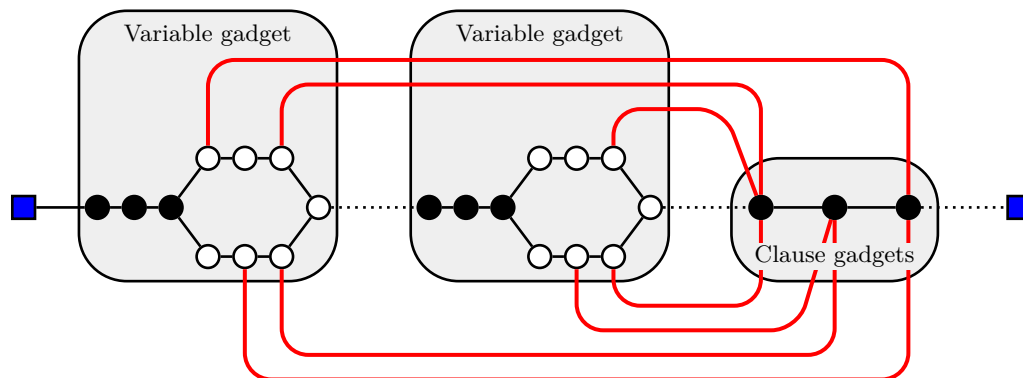
- Graph H is the union of a Hamiltonian cycle that first visits all elements of X and then all elements of Y , and the incidence graph of ϕ .
- Graph H is planar and there is an embedding where each edge between a clause and a variable that appears negated in the clause is inside the cycle, and each edge between a clause and a variable that appears non-negated in the clause is outside of the cycle.
- Each variable appears in at most three clauses.

We may assume that we know the embedding of H and the Hamiltonian cycle [37].

Informally speaking, the main idea on how to reduce from LINKED PLANAR 3-SAT to COLLISION-FREE MOVEMENT TO CONNECTIVITY planar graphs is as follows. We split the Hamiltonian cycle in H between the last clause vertex and the first variable vertex, in that way obtaining a Hamiltonian path that first traverses all variable vertices and then all clause vertices. If we draw this path horizontally (from left to right), then (in a planar drawing) each edge between a clause and a variable that appears negated in the clause is below the path, and each edge between a clause and a variable that appears non-negated in the clause is above of the path. We add an extra vertex to the beginning of the path and an extra vertex to the end, and place a main robot on each of them. Now we “replace” each variable vertex with the variable gadget illustrated in Figure 2. We connect edges above the path to the three vertices on top of the gadget, and edge below the path to the three vertices below. For an illustration see Figure 3. Then these edges (that originally connected the clause and variable vertices) are subdivided a number of times and obnoxious robots are places on all newly created vertices. Furthermore, obnoxious robots are places on all clause vertices. The idea is that the three obnoxious robots in each variable gadget (see Figure 2) need to move either to the three upper or lower vertices, which will encode setting the variable to true or false. Now the main robot on the left can freely move until the last variable gadget. Then



■ **Figure 2** Illustration of a variable gadget. The black vertices have obnoxious robots on them.



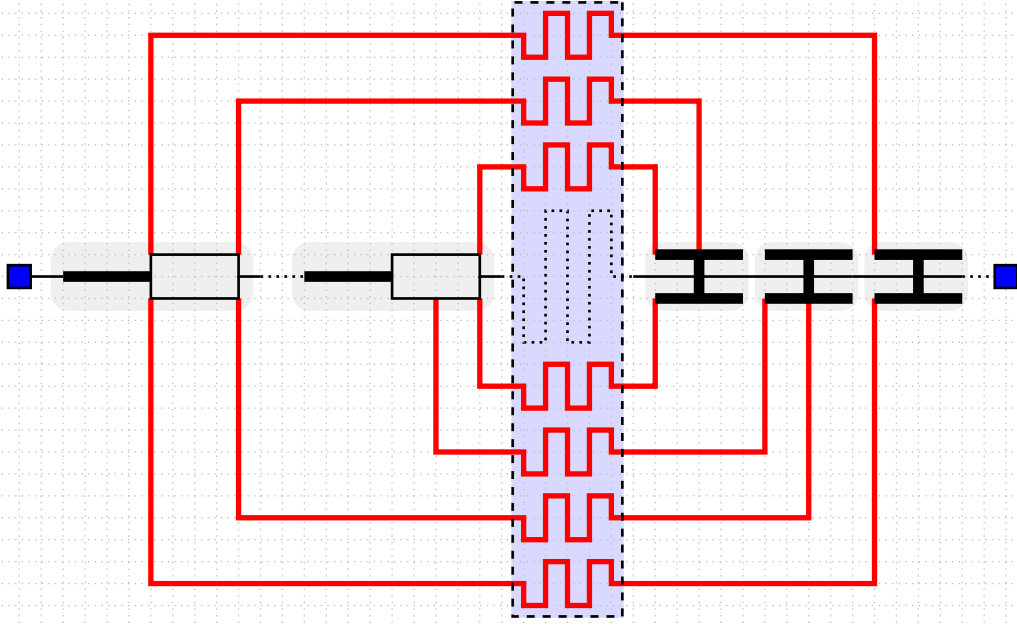
■ **Figure 3** Illustration of the informally described graph G . The blue filled round vertices have obnoxious robots on them, and the black filled square vertices have main robots on them. The red edges correspond to long paths that have obnoxious robots on all inner vertices.

the obnoxious robot on each clause vertex needs to move out of the way. To this end, all obnoxious robots on one of the paths connecting the clause vertex to a variable gadget need to move. This is only possible if the vertex of the variable gadget where the path arrives is free, hence encoding that the corresponding literal satisfies the clause. Now the main robots can freely meet on some adjacent clause vertices.

Afterwards, we modify the informally described graph (call it G) depicted in Figure 3 to produce a graph G' that is a grid (together with an embedding that shows this). Note that there are three main obstacles to overcome:

- The clause vertices in G may have degree 5, which is not possible in a grid. Hence, we have to modify the vertices to “clause gadgets” that are grids.
- All paths connecting clause gadgets to variable gadgets need to have the same length in order for the reduction to work.
- The path between the last variable gadget and the first clause gadget has to be sufficiently long (that is, longer than any path from a clause gadget to a variable gadget) for the reduction to work.

We introduce clause gadgets, which are, informally speaking, small tree-like grids rooted at the clause vertices and three leaves pointing upward, and three pointing downward. All vertices in the gadgets have obnoxious robots, and the paths to the variable gadgets now start at the leaves. In order to ensure that all paths between variable and clause gadgets have the appropriate length, we introduce a “zone” in the grid between all variable gadgets and all clause gadgets, which is sufficiently large such that we can change the length of each path to the desired length by using “snake-like” embeddings in the grid. For an illustration see Figure 4.



■ **Figure 4** Illustration of an embedding of the graph G' obtained from graph G . The drawing is not up to scale but rather should give the rough idea on how the embedding looks like. The grid points on the thick black lines have obnoxious robots on them, and the blue filled square vertices have main robots on them. The red edges correspond to long paths that have obnoxious robots on all inner vertices. The light blue area surrounded by the black dashed line indicates the part of the grid, where we make sure that all red paths have the same length, and the path connecting the last variable gadget with the first clause gadget is sufficiently long. This is also indicated by the snake-like shape of the paths in the area.

Collision Free Movement on Grids (Section 5). We give several kernelization algorithms and kernelization lower bounds for COLLISION-FREE MOVEMENT TO CONNECTIVITY on grids. In fact, our first kernelization algorithm works for general formations Π . The main insight here is that vertices that are far away from each of the main robots (regardless of whether they are occupied by obnoxious robots or not) cannot be relevant for any solution. This allows us to obtain a reduction rule that, when applied exhaustively, yields a kernel of size $O(k \cdot c^2)$ for COLLISION-FREE MOVEMENT TO Π (Theorem 3). If Π is connectivity, we can improve the size of the kernel. In this case, informally speaking, we have that if a vertex is too far away from *some* main robot, it cannot be relevant, since eventually, all main robots have to be close together. This allows us to obtain an additional reduction rule. Exhaustively applying both rules yields a kernel of size $O(\min\{k \cdot c^2, k^2 + c^2\})$ for COLLISION-FREE MOVEMENT TO CONNECTIVITY (Theorem 4).

Furthermore, we give a number of kernelization lower bounds, using the well-known framework of OR-cross-compositions [4, 25]. For all results, we cross-compose from LINKED PLANAR 3-SAT and reuse (modified versions of) the constructions devised to obtain Theorem 2. We first show that we presumably cannot obtain polynomial kernels for COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by c on planar graphs, even if $k = 2$ (Theorem 5). To this end, we use the planar instances of COLLISION-FREE MOVEMENT TO CONNECTIVITY produced by the reduction behind Theorem 2 before the modifications to obtain a grid are applied (Figure 3). We remove the main robots from each instance, and



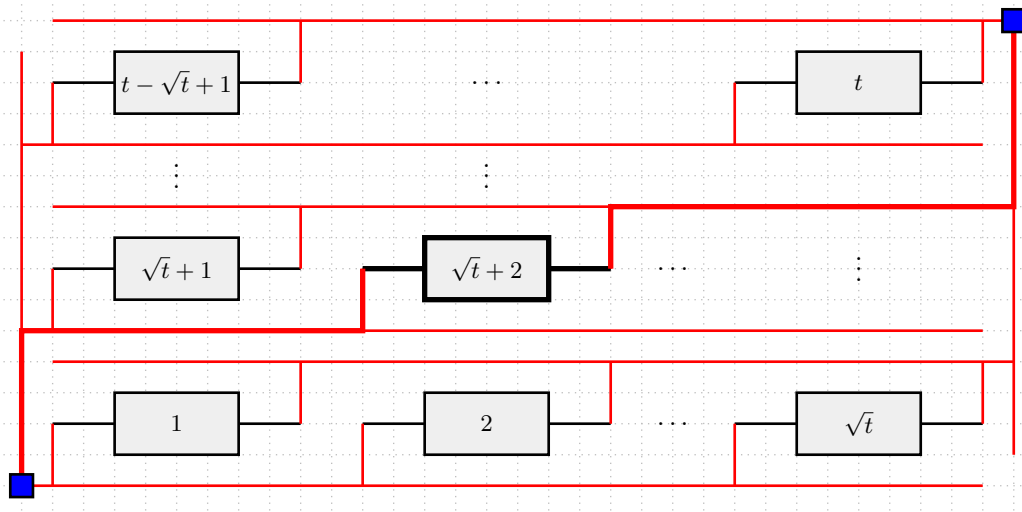
■ **Figure 5** Illustration of the OR-cross-composition. Blue filled square vertices have main robots on them.

connect the (previous) starting locations of the main robots to a new starting location with sufficiently long paths (ensuring that the main robots cannot move between instances). For an illustration see Figure 5. This result shows that the restriction to grids is essential for obtaining polynomial kernels.

We further show that we cannot obtain polynomial kernels for COLLISION-FREE MOVEMENT TO CONNECTIVITY on grids for the single parameter c (Theorem 7). Note that Theorem 2 obviously rules out polynomial kernels for the single parameter k . Here, we use the grid instances produced by the reduction behind Theorem 2. Note that we cannot connect the instances in the same fashion as in the previous cross-composition, as this introduces a high-degree vertex which is not embeddable in a grid. We replace this with a tree-like grid that has a main robots each vertex. Intuitively, we cannot use a single main robot since the distance between its starting location and the instances would be too large. Now, instead of single main robots moving through one of the instances, a “snake” of main robots moves through it to establish a connection between the right and the left side of the instance. To accommodate this, we have to also make some modifications to instances produced by the reduction behind Theorem 2.

Finally, we show that we presumably cannot significantly improve the size of the kernel produced by Theorem 4 (Theorem 7). Here, we use the weak cross-compositions framework [25, 30]. Informally speaking, we have to ensure here that the parameter of the instance produced by the weak cross-composition can be upper-bounded by the maximum size among the input instances multiplied by the square root of the number of input instances. We can achieve this by again using the instances produced by the reduction behind Theorem 2 and arranging the input instances in a grid-like fashion, see Figure 6. Furthermore, we remove the main robots from each input instance and use one main robot starting in the lower left of the arrangement and one in the upper right. Together with appropriate connections between the instances, this ensures that the sum of the distances of both main robots to any of the instances is in $O(\sqrt{t} \cdot x_{\max})$, where t is the number of input instances and $x_{\max}$ is the maximum instance size.

Collision Free Movement on Planar Graphs (Section 6). We show that COLLISION-FREE MOVEMENT TO Π is fixed-parameter tractable when parameterized by $k + c$ if the input graph is planar (Theorem 8). Note that this result holds for general formation Π . We obtain it by providing a monadic second-order logic (MSO) formulation of the problem and then employing Courcelle’s famous theorem [2, 5, 6], stating that MSO MODEL CHECKING is fixed-parameter tractable w.r.t the size of the formula and the treewidth of the relational



■ **Figure 6** Illustration of the weak cross-composition. The drawing is not up to scale but rather should give the rough idea on how the embedding looks like. Black boxes correspond to the instances and red lines to the connecting paths. Blue filled square vertices have main robots on them. The thick lines correspond to the paths visited by the main robots when instance nr. $\sqrt{t} + 2$ is selected.

structure on which the formula is defined.

As the relational structure, we take the input graph together with “time steps” that model at which point in a movement a vertex is visited by a robot or an edge is traversed by a robot. Since the number of moves is upper-bounded by c , we need c of such time steps. Informally speaking, we associate each time step with each of the vertices and edges, which increases the treewidth of the graph by some factor depending on c . Additionally, we can remove vertices that are far away from any main robot (similar to the reduction rule in the kernelization algorithm). This allows us to upper-bound the diameter of the input graph by a function in $k + c$ and, since it is planar, also its treewidth. It follows that the treewidth of the relational structure can be upper-bounded by a function in $k + c$.

Now we define movements as sets of vertex-time step pairs with the following properties:

- For each time step, there is one vertex-time step pair in the set.
- For each pair of consecutive time steps, the corresponding vertices are either equal (the robot does not move) or connected by an edge.
- The vertex corresponding to the first time step is the starting location of a robot.

These properties can be checked with MSO formulas. Furthermore, we can check (with an MSO formula) for each pair of movements, whether the two corresponding robots do not collide. If during a movement, a robot visits a vertex that is the starting location of another robot, there must be another movement moving that robot out of the way. Finally, we can have at most c non-trivial movements (that is, the robot actually moves at least once). Putting this together, we can obtain an MSO formula that checks for a set of c movements, whether they are collision-free and also do not collide with any robot that is not moved. Finally, we can straightforwardly determine the locations of all main robots after the movements and check whether the formation has property II. We remark that the size of the overall MSO formula only depends on c and the size of a subformula that checks whether a graph on k vertices has property II (which, notably, is constant if II is connectivity).

Collision Free Movement on Unit Disk Graphs (Section 7). We show that COLLISION-FREE MOVEMENT TO CONNECTIVITY on unit disk graphs admits an FPT approximation algorithm when parameterized by $k + c$ for the canonical optimization variant of the problem, where we aim to minimize the energy. More specifically, we give a fixed-parameter approximation for COLLISION-FREE MOVEMENT TO CONNECTIVITY on unit disk graphs when parameterized by $k + c$, that produces a solution with energy at most $2 \cdot \text{OPT} + 3k$, where OPT is the energy of an optimal solution (Theorem 9). The algorithm makes use of the fact that unit disk graphs are a subclass of so-called clique grids. Informally speaking, a clique grid, each vertex is associated with a grid cell, and all vertices in the same cell form a clique. All additional edges are between vertices in grid cells of distance at most two.

We start by “guessing” the optimal value for c , and removing all vertices that are too far away from any main robot starting location (as in the kernelization algorithms). Afterwards, the number of cliques / grid cells in the clique grid is upper-bounded by some function in $k + c$. Furthermore, we guess a spanning tree that witnesses the connectivity of the formation of the main robots after the movements.

We introduce a *type* for each vertex of a clique that, intuitively, describes via which cliques the vertex can be reached from each main robot and whether it is occupied by an obnoxious robot. We show how to upper-bound the number of types in a function of c and k . We guess the type of each vertex in the spanning tree. Now, informally speaking, we guess up to c movement vectors of types that move the main robots to vertices of the correct type (as specified by the guesses for the spanning tree), and check whether they are collision-free. When two robots move to vertices of the same type, we need to ensure that these are different vertices to be collision-free. This, intuitively, is where the factor two in the approximation comes from. We potentially need to move robots to a “free” vertex of a certain type, which requires at most one extra move as all vertices of the same type form a clique. Finally, at the end we may need to move all main robots to the “correct” vertex of the guessed type in order to obtain the guessed spanning tree, which, as we show, requires at most three extra moves per main robot.

3 Preliminaries and Known Results

In this section, we provide all necessary notation and terminology, as well as some preliminary results that are easy to obtain or straightforwardly follow from known results.

Parameterized Complexity. We use the following standard concepts from parameterized complexity theory [8, 15, 23, 25]. A *parameterized problem* $L \subseteq \Sigma^* \times \mathbb{N}$ is a subset of all instances (x, k) from $\Sigma^* \times \mathbb{N}$, where k denotes the *parameter*. A parameterized problem L is in the class FPT (or *fixed-parameter tractable*) if there is an algorithm that decides every instance (x, k) for L in $f(k) \cdot |x|^{O(1)}$ time for some computable function f that depends only on the parameter. A decidable parameterized problem L admits a *kernel* if there is a polynomial-time algorithm that transforms each instance $(x, k) \in \{0, 1\} \times \mathbb{N}$ into an instance $(x', k') \in \{0, 1\} \times \mathbb{N}$ such that $|x'| + k' \leq f(k)$, for some computable function f , and $(x', k') \in L$ if and only if $(x, k) \in L$. It is known that a decidable parameterized problem L admits a kernel if and only if it is fixed-parameter tractable [25]. If f is a polynomial, that is, $|x'| + k' \in r^{O(1)}$, we say that L admits a *polynomial kernel*. Presumably, not all problems that are fixed-parameter tractable admit a polynomial kernel [25]. We give more details on how to exclude polynomial kernels under standard complexity assumptions in the respective section of the paper. A parameterized problem L is in the class XP, if there is

an algorithm that decides every instance (x, k) for L in $|x|^{f(k)}$ time for some computable function f that depends only on the parameter. If a parameterized problem L is W[1]-hard, then it is presumably not contained in FPT [8, 15, 23].

Graph Theory. We use standard notation and terminology from graph theory [14]. Let $G = (V, E)$ with $E \subseteq \binom{V}{2}$ be a graph. We denote by $V(G) = V$ and $E(G) = E$ the sets of its vertices and edges, respectively. We use n to denote the number of vertices of G . We call two vertices $u, v \in V$ *adjacent* if $\{u, v\} \in E$. The *degree* of a vertex $v \in V$ is $\deg(v) = |\{u \mid \{u, v\} \in E\}|$. We say that a graph G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. For some vertex set $V' \subseteq V$, we denote by $G[V']$ the *induced* subgraph of G on the vertex set V' , that is, $G[V'] = (V', E')$ where $E' = \{\{v, w\} \mid \{v, w\} \in E \wedge v \in V' \wedge w \in V'\}$. We further denote $G - V' = G[V \setminus V']$. We say that a sequence $P = (\{v_{i-1}, v_i\})_{i=1}^\ell$ of edges in E forms a *path* of length ℓ in G from v_0 to v_ℓ if $v_i \neq v_j$ for all $0 \leq i < j \leq \ell$. We say that vertices u and v are *connected* in G if there is a path from u to v in G . We say that graph G is connected if for all $u, v \in V$ we have that u and v are connected. We say that $G[V']$ for some $V' \subseteq V$ is a *connected component* of G if $G[V']$ is connected and for all $v \in V \setminus V'$ we have that $G[V' \cup \{v\}]$ is not connected. We say that a path P of length ℓ in G from vertex u to v is a *shortest path* if there is no path from u to v in G of length $\ell - 1$. The *distance* between u and v in G is the length of a shortest path from u to v in G and denoted $\text{dist}(u, v)$. If u and v are in different connected components of G , then we set $\text{dist}(u, v) = \infty$. The *diameter* of a graph G is the maximum distance between any two vertices that are in the same connected component, that is, $\text{diam}(G) = \max_{u, v \in V(G) \mid \text{dist}(u, v) \neq \infty} \text{dist}(u, v)$.

A graph G is *planar* if it can be embedded in the plane (drawn with points for vertices and curves for edges) without crossing edges. A graph $G = (V, E)$ is a *unit disk graph* if it has an intersection model consisting of disks of unit size, that is, there is a $f : V \rightarrow \mathbb{R}^2$ such that for all $\{u, v\} \in E$ it holds that $|f(u) - f(v)| \leq 1$. The set $D = \{f(v) \mid v \in V\}$ is called the *point set* of G . A graph $G = (V, E)$ is a *grid* if there exists a function $f : V \rightarrow [t] \times [t']$ for some $t, t' \in \mathbb{N}$ such that for all $\{u, v\} \in E$ it holds that if $f(u) = (i, j)$ and $f(v) = (i', j')$ then $|i - i'| = 1$ and $j = j'$, or $|j - j'| = 1$ and $i = i'$. Note that grids are planar graphs and unit disk graphs.

Problem Definition. Assume we are given a graph $G = (V, E)$ and two sets $R, O \subseteq V$ with $R \cap O = \emptyset$. Here, R denotes the starting positions of the *main robots* and O denotes the starting positions of the *obnoxious robots*. We denote $k = |R|$ and $\ell = |O|$. Let $m \in \mathbb{N}$ $r \in R \cup O$. We call a vector $M_r = (v_1, v_2, v_3, \dots, v_m)$ a *movement* for r with m steps if $v_1 = r$ and for all $i \in [m - 1]$ it holds that $v_i = v_{i+1}$ or $\{v_i, v_{i+1}\} \in E$. Intuitively, in every step of a movement, the robot may stay on its current position or move to an adjacent vertex. The *length* $\text{len}(M_r)$ of a movement $M_r = (v_i)_{i \in [m]}$ is $\text{len}(M_r) = |\{i \in [m - 1] \mid v_i \neq v_{i+1}\}|$, that is, the number of steps in which r moves to an adjacent vertex. We say that a movement M_r is *trivial* if $\text{len}(M_r) = 0$. We say that two movements $M_r = (v_i)_{i \in [m]}$ and $M_{r'} = (v'_i)_{i \in [m]}$ for some $r, r' \in R \cup O$ with m steps each are *collision-free* if for all $i \in [m]$ it holds that $v_i \neq v'_i$ and for all $i \in [m - 1]$ it holds that $\{v_i, v_{i+1}\} \neq \{v'_i, v'_{i+1}\}$. Intuitively, robots collide if they occupy the same vertex in the same step or if they use the same edge (in opposite directions) to move to an adjacent vertex in the same step. A *movement plan* for $R \cup O$ is a set $\mathcal{M} = \{M_r \mid r \in R \cup O\}$ containing a movement for each robot in $R \cup O$ with the same number of steps. The length of a movement plan $\text{len}(\mathcal{M})$ is the sum of length of all movements in $\mathcal{M}$, that is, $\text{len}(\mathcal{M}) = \sum_{M_r \in \mathcal{M}} \text{len}(M_r)$. We say that a movement plan $\mathcal{M}$ is *collision-free* if for all $M_r, M_{r'} \in \mathcal{M}$ with $r \neq r'$ we have that the movements M_r and $M_{r'}$

are collision-free. The *end-configuration* of R after a movement plan $\mathcal{M} = \{M_r \mid r \in R \cup O\}$ is $R^*(\mathcal{M}) = \{v_m \mid \exists M_r \in \mathcal{M} \text{ s.t. } r \in R \text{ and } M_r = (v_i)_{i \in [m]}\}$, that is, the set of vertices that are occupied by robots in R after all movements.

Let Π be a recursively enumerable graph property. We define the problem COLLISION-FREE MOVEMENT TO Π as follows.

Collision-Free Movement to Π

Input: A graph $G = (V, E)$, two sets $R, O \subseteq V$ with $R \cap O = \emptyset$, and an integer $c \in \mathbb{N}$.

Question: Is there a collision-free movement plan $\mathcal{M}$ for $R \cup O$ such that $\text{len}(\mathcal{M}) \leq c$ and $G[R^*(\mathcal{M})] \in \Pi$?

In this work, we will focus on the case of Π being the property of a graph to be *connected*. We call this problem COLLISION-FREE MOVEMENT TO CONNECTIVITY.

First Observations and Results. Since COLLISION-FREE MOVEMENT TO Π contains the problem of deciding whether a graph has property Π as a special case, we immediately get the following.

► **Observation 10.** *If deciding whether a graph has property Π is NP-hard, then COLLISION-FREE MOVEMENT TO Π is NP-hard even if $\ell = 0$ and $c = 0$.*

If we do not require the movement plan to be collision-free, then we call the problem MOVEMENT TO Π . A generalization of MOVEMENT TO Π has been studied by Demaine et al. [11, 12]. We can observe that COLLISION-FREE MOVEMENT TO Π and MOVEMENT TO Π coincide if $\ell = 0$.

► **Lemma 11.** *Let $I = (G = (V, E), R, O, c)$ be an instance of COLLISION-FREE MOVEMENT TO Π . If $O = \emptyset$, then I is a yes-instance of COLLISION-FREE MOVEMENT TO Π if and only if I is a yes-instance of MOVEMENT TO Π .*

Proof. If I is a yes-instance of COLLISION-FREE MOVEMENT TO Π , then clearly it is also a yes-instance of MOVEMENT TO Π . For the other direction, assume that I is a yes-instance of MOVEMENT TO Π and let $\mathcal{M}$ be a movement plan that is a solution for I . Assume that $\mathcal{M}$ is not collision-free. Now let $M = (v_1, v_2, \dots, v_m)$ and $M' = (v'_1, v'_2, \dots, v'_{m'})$ be two movements for robots $r, r' \in R$, respectively, in $\mathcal{M}$ that collide. Let t be the step in which a collision happens, that is, $v_t = v'_t$, or $v_t = v'_{t-1}$ and $v'_t = v_{t-1}$. Consider the latter case first, that is, $v_t = v'_{t-1}$ and $v'_t = v_{t-1}$. Here, instead of the robots swapping their position, they can “swap their identities” instead. Formally, we can define a new movement M^* for robot r which equals M for the first $t-1$ steps, and equals M' for all subsequent steps, that is, $M^* = (v_1, v_2, \dots, v_{t-1}, v'_t, v'_{t+1}, \dots, v'_{m'})$. Analogously, we define a new movement for r' . Clearly, the end-configuration of $\mathcal{M}$ stays the same. By iterating this modification, we can eliminate all collisions of this type. Now consider the former case, that is, $v_t = v'_t$. First consider the case that both r and r' moved to $v_t = v'_t$ in step t , that is $v_{t-1} \neq v_t$ and $v'_{t-1} \neq v'_t$. Then we create a new movement for r where the movement from v_{t-1} to v_t happens one time step later. By iterating this modification, we can create a movement plan where no two robots simultaneously move to the same vertex. Now assume w.l.o.g. that robot r moves to v_t in step t , that is, $v_{t-1} \neq v_t$ and robot r' was already at $v_t = v'_t$ at the previous step, that is, $v'_t = v'_{t-1}$. Furthermore, assume that t is the latest step where such a collision happens. Then at some later time step $t' > t$ we must have that one of the robots moves away. Assume w.l.o.g. that r' moves away first (if only r moves away, they can “swap identities” according to the modification above). This implies that $v'_{t-1} = v'_t = \dots = v'_{t'-1}$ in

M' . Then we can create a new movement M^* for r' that is equal to M' until step $t - 1$ and then “skips” steps t to $t' - 1$ and immediately moves to $v'_{t'}$ at step t and then continues with the steps of M' from t' onward, that is, $M^* = (v'_1, v'_2, \dots, v'_{t-1}, v'_{t'}, v'_{t'+1}, \dots, v'_{m'})$. Note that this may introduce new collisions, but it (strictly) shortens one of the movement vectors. It follows that iterating this modification will eventually eliminate all collisions. $\blacktriangleleft$

Lemma 11 implies that algorithmic results for MOVEMENT TO Π carry over to COLLISION-FREE MOVEMENT TO Π (and vice versa) if $\ell = 0$. In particular, we get the following.

► **Theorem 12** ([12]). *If $\ell = 0$, then COLLISION-FREE MOVEMENT TO CONNECTIVITY is fixed-parameter tractable when parameterized by k .*

Due to the similarity of MOVEMENT TO Π and COLLISION-FREE MOVEMENT TO Π , the reductions behind some known hardness results for MOVEMENT TO Π can be straightforwardly modified to work for COLLISION-FREE MOVEMENT TO Π . In particular, Demaine et al. [12] show that if Π is hereditary and does not contain all complete graphs and all empty graphs, then MOVEMENT TO Π is $W[1]$ -hard when parameterized by $k + c$ even if $\ell = 0$ via a reduction from the well-known Π -subgraph detection problem [33]. The reduction formalizes the intuition that already finding a suitable target location for the main robots is hard, rather than finding out in which way the robots move to that location. The reduction can be straightforwardly modified to yield the following result.

► **Theorem 13** ([12, 33]). *If Π is hereditary and does not contain all complete graphs and all empty graphs, then COLLISION-FREE MOVEMENT TO Π is $W[1]$ -hard when parameterized by $k + c$ even if $\ell = 0$.*

Demaine et al. [11] show that MOVEMENT TO CONNECTIVITY is NP-hard even if $\ell = 0$ via a reduction from HAMILTONIAN PATH [32]. This reduction can be straightforwardly modified as well. If additionally we reduce from HAMILTONIAN PATH on planar graphs [27], we obtain the following result.

► **Theorem 14** ([11]). *COLLISION-FREE MOVEMENT TO CONNECTIVITY is NP-hard even if $\ell = 0$ and the input graph is planar.*

If the main robots have prescribed destinations, that is, for each $r \in R$ there is a $d_r \in V$ given as part of the input, and we ask for a collision free movement plan $\mathcal{M}$ (of length at most c) that moves each robot in R to its destination, that is, for each $r \in R$ we have $M_r \in \mathcal{M}$ with $M_r = (v_i)_{i \in [m]}$ and $v_m = d_r$, then the problem is called COLLISION-FREE MOVEMENT. A generalization of COLLISION-FREE MOVEMENT has been studied by Deligkas et al. [10].

Observe that, given an instance (G, R, O, c) of COLLISION-FREE MOVEMENT TO Π , if we can “guess” a destination vertex d_r for each main robot $r \in R$ such that the subgraph of G induced on the destination vertices has property Π , then we can reduce the problem of finding a collision-free movement plan of length at most c to an instance of COLLISION-FREE MOVEMENT. In particular, this gives us the following.

► **Observation 15.** *If deciding whether a graph has property Π is in NP, then COLLISION-FREE MOVEMENT TO Π is in NP.*

Last but not least, we can observe that COLLISION-FREE MOVEMENT TO Π can be solved by a simple guess-and-check algorithm if deciding whether a graph has property Π is polynomial-time solvable. We can iterate over all $n^{O(c)}$ possibilities to pick c moves and

assign each one to one of the robots. Then we check whether this yields a collision-free movement plan and whether the end-configuration yields a subgraph with property Π . Hence, we get the following.

► **Theorem 16.** *If deciding whether a graph has property Π is polynomial-time solvable, then COLLISION-FREE MOVEMENT TO Π is in XP when parameterized by c .*

4 Hardness Results

In this section, we give our main hardness results for COLLISION-FREE MOVEMENT TO CONNECTIVITY. We begin by proving Theorem 1 (restated below), that is, showing that the problem is $W[1]$ -hard when parameterized by $k + c$ on general graphs. Afterwards, we investigate the cases where the input graph is a grids.

► **Theorem 1.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY is $W[1]$ -hard when parameterized by $k + c$.*

To prove Theorem 1, we present a parameterized reduction from MULTICOLORED CLIQUE [19]. Here, given an ℓ -partite graph $H = (W_1 \uplus W_2 \uplus \dots \uplus W_\ell, F)$, we are asked whether H contains a clique of size ℓ . If $v \in W_i$, then we say that v has *color* i . Let $F_{i,j}$ denote the set of all edges between vertices from W_i and W_j . This problem is known to be $W[1]$ -hard when parameterized by ℓ [19].

► **Construction 1.** Given an instance $H = (W_1 \uplus W_2 \uplus \dots \uplus W_\ell, F)$ of MULTICOLORED CLIQUE, we construct an instance $(G = (V, E), R, O, c)$ of COLLISION-FREE MOVEMENT TO CONNECTIVITY as follows.

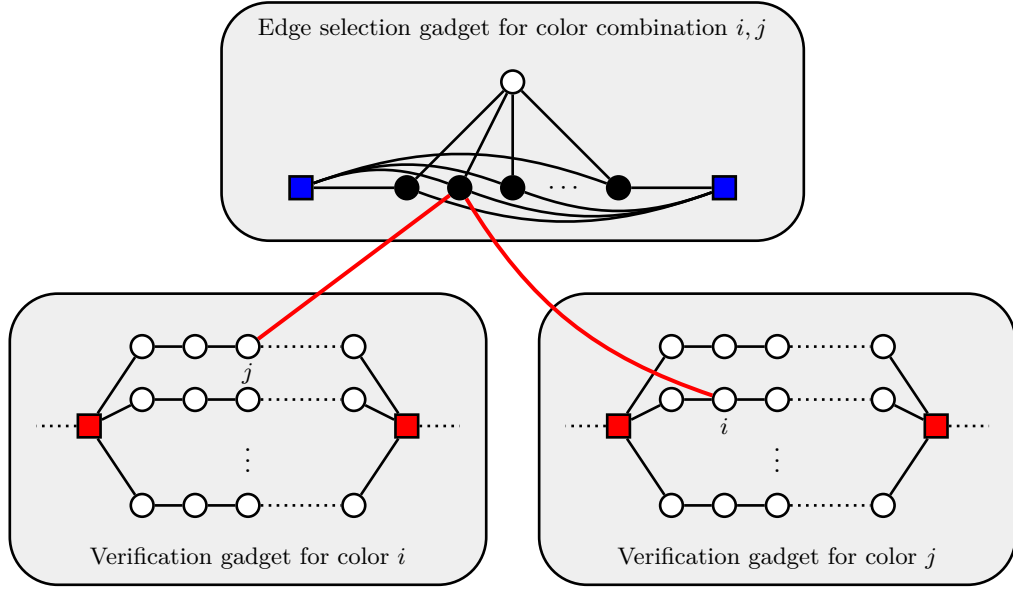
For each color combination i, j with $i < j$ we create an edge selection gadget $G_{i,j}$.

- For every edge $e \in F_{i,j}$ we add a vertex v_e , place an obnoxious robot on it.
- We add three additional vertices v_1, v_2, v_3 . We connect v_1 to each v_e with $e \in F_{i,j}$ and place a main robot on v_1 . We connect v_2 to each v_e with $e \in F_{i,j}$ and place a main robot on v_2 . We connect v_3 to each v_e .

For an illustration see Figure 7. For each color i we create a validation gadget G_i as follows.

- For every vertex $w \in W_i$ we create $\ell - 1$ vertices $w_1, w_2, \dots, w_{i-1}, w_{i+1}, \dots, w_\ell$ and connect them to a path (from w_1 to w_ℓ).
- We add two vertices $u_1^{(i)}, u_2^{(i)}$ and place main robots on them. We connect $u_1^{(i)}$ to each w_1 with $w \in W_i$. We connect $u_2^{(i)}$ to each w_ℓ with $w \in W_i$. We create a path on c vertices, place main robots on each vertex, and connect one of the endpoints to $u_1^{(i)}$, where we specify c later. We create a second path on c vertices, place main robots on each vertex, and connect one of the endpoints to $u_2^{(i)}$.

The graph G is now the disjoint union of the edge selection gadgets of all color combinations i, j with $i < j$ and the verification gadgets of all colors i . We connect the gadgets as follows. Let $j = i + 1 \leq \ell$, then we connect vertex $u_2^{(i)}$ from the verification gadget G_i and vertex $u_1^{(j)}$ from the verification gadget G_j with an edge. Finally, let $e = \{w, w'\} \in F_{i,j}$ such that $w \in W_i$ and $w' \in W_j$. We connect vertex v_e from the edge selection gadget $G_{i,j}$ with vertex w_j from the verification gadget G_i . Then we subdivide ℓ^2 times the edge $\{v_e, w_j\}$. We connect vertex v_e from the edge selection gadget $G_{i,j}$ with vertex w'_i from the verification gadget G_j . Then we subdivide ℓ^2 times the edge $\{v_e, w'_i\}$. Finally, we set $c = (2\ell^2 + 3)\binom{\ell}{2}$.



■ **Figure 7** Illustration of the graph G produced by Construction 1. The black filled round vertices have obnoxious robots on them, and the blue filled square vertices have main robots on them. The red filled square vertices have main robots on them and paths of length c attached to them that have main robots on each vertex (which are not depicted). The two red edges correspond to paths of length ℓ^2 connecting the edge selecting gadget to the verification gadgets.

This finished the construction, which can clearly be performed in polynomial time. Note that the number of main robots is in $O(\ell^4)$ and $c \in O(\ell^4)$. Now we prove the correctness of the reduction, that is, the constructed instance (G, R, O, c) is a yes-instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY if and only if H is a yes-instance of MULTICOLORED CLIQUE. This together with the above observations yields Theorem 1.

Proof of Theorem 1. Let be $H = (W_1 \uplus W_2 \uplus \dots \uplus W_\ell, F)$ an instance of MULTICOLORED CLIQUE and let (G, R, O, c) be an instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY created from H by Construction 1.

$(\Rightarrow)$: Assume that H is a yes-instance of MULTICOLORED CLIQUE and we are given a clique $X \subseteq V(H)$. We construct a collision-free movement plan $\mathcal{M}$ of length c for (G, R, O, c) as follows.

For each color combination i, j with $i < j$ let $e = \{w, w'\} \in F_{i,j}$ such that $w \in W_i$, $w' \in W_j$, and $w, w' \in X$. We move the obnoxious robot on vertex v_e of the edge selection gadget $G_{i,j}$ to vertex v_3 of the edge selection gadget $G_{i,j}$. By Construction 1, we have that v_e is connected to vertex w_j from the verification gadget G_i via a path of length $\ell^2 + 1$ and is connected to vertex w'_i from the verification gadget G_j via a path of length $\ell^2 + 1$. We move the main robot on v_1 to w_j via v_e and the connecting path, and we move the main robot on v_2 to w'_i via v_e and the connecting path. Note that this is collision-free and takes $2\ell^2 + 2$ moves. This finishes the description of the movement plan. Note that it uses $2\ell^2 + 3$ moves for every edge selection gadget and hence $c = (2\ell^2 + 3) \binom{\ell}{2}$ moves in total.

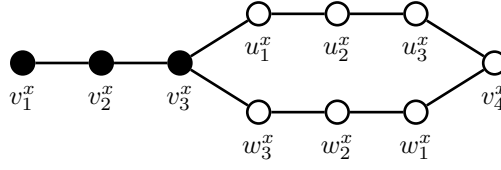
It remains to show that the end-configuration forms a connected subgraph. To this end, note that for some fixed i , all edges of a color combination that contains i that are contained in the clique X have the same endpoint in W_i . Let that endpoint be $w \in W_i$

Hence, we have that in the verification gadget for color i , one main robot is moved to each of $w_1, w_2, \dots, w_{i-1}, w_{i+1}, \dots, w_\ell$ and hence vertices $u_1^{(i)}, u_2^{(i)}$ (which have main robots on them) are connected via a path that has a main robot on each vertex. Since this is the case for every verification gadget and there are no main robots left in the edge selection gadgets, we have that the main robots form a connected subgraph.

($\Leftarrow$) : Assume that (G, R, O, c) is a yes-instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY and we are given a collision-free movement plan $\mathcal{M}$ of length c that is a solution for (G, R, O, c) . We construct a clique of size ℓ in H as follows.

Assume that $\mathcal{M}$ is a movement plan with a minimum number of moves. First, observe the following. For each verification gadget for a color i it holds that the main robots on $u_1^{(i)}, u_2^{(i)}$ do not move. Recall that by Construction 1, each of the vertices is connected to a path of length c that has main robots on every vertex. Moving the main robot on $u_1^{(i)}$ or $u_2^{(i)}$ breaks the connection to the main robots on the respective path. To reestablish connection, either the robot needs to move back to $u_1^{(i)}$ or $u_2^{(i)}$, respectively, which does not happen in a movement plan with a minimum number of moves, or all robots on the path need to be moved by at least one step, which costs more than c moves in total and hence cannot happen in a solution. It follows that only the $2\binom{\ell}{2}$ main robots in edge selection gadgets move. Furthermore, notice that the paths between verification gadgets and edge selection gadgets have length $\ell^2 > 2\binom{\ell}{2}$. It follows that a connection cannot be established via those paths since there are not enough main robots that can move. We can conclude that for each color i the vertices $w_1, w_2, \dots, w_{i-1}, w_{i+1}, \dots, w_\ell$ for some $w \in W_i$ need to be occupied by main robots in the end-configuration. The distance of a starting position of a main robot in an edge selection gadget to such a vertex is at least $\ell^2 + 2$. Furthermore, all neighboring vertices of starting position of a main robot in an edge selection gadget are occupied by obnoxious robots that need to be moved away first. Note that for each edge selection gadget, there are at most one move available to do this. It follows that in each edge selection gadget, we move exactly one obnoxious robot from some v_e to the v_3 vertex of that gadget. We say that edge e is “selected”. Furthermore, we have that for each color i , exactly the vertices $w_1, w_2, \dots, w_{i-1}, w_{i+1}, \dots, w_\ell$ for some $w \in W_i$ are occupied by main robots in the end-configuration, which each were moved there with exactly $\ell^2 + 2$ moves.

Let X be the set of all endpoints of selected edges. We claim that X forms a clique of size ℓ in H . Note that X contains at least one vertex of each color, hence $|X| \geq \ell$. In the remainder, we show that also $|X| \leq \ell$. Note that this implies that X is a clique, since for each color i we have that all edges of a color combination that contains i have the same endpoint of color i . Assume for contradiction that $w, w' \in X$ with $w, w' \in W_i$ for some color i . Then there are selected edges e, e' such that w is an endpoint of e and w' is an endpoint of e' . Consider the edge selection gadgets where e and e' were selected. Both correspond to a color combination that involves i , so both are connected to the verification gadget of color i . If two main robots from one of the gadgets are moved into the verification gadget of color i , this involves at least $\ell^2 + 3$ moves. Hence, by arguments made above, this is not possible in a solution. We can conclude that from both edge selection gadgets, one main robot must be moved to the verification gadget of color i with $\ell^2 + 2$ moves. Then, however, the main robots are moved to vertices w_j and $w_{j'}$, for some j, j' in the verification gadget for color i . This is a contradiction to the observation above, that for color i the vertices $w_1, w_2, \dots, w_{i-1}, w_{i+1}, \dots, w_\ell$ for some $w \in W_i$ need to be occupied by main robots in the end-configuration. We can conclude that X is a clique in H . $\blacktriangleleft$



■ **Figure 8** Illustration of a variable gadget G_x for a variable x (Construction 2). The black filled vertices have obnoxious robots on them.

Hardness on Grids. Now we prove Theorem 2 (restated below), that is, we show that COLLISION-FREE MOVEMENT TO CONNECTIVITY is NP-hard if the input graph is a grid, even if there are only two main robots. To this end, we first give a reduction that produces a planar graph, and then we show how to modify the construction such that we obtain a grid.

► **Theorem 2.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY is NP-hard even if $k = 2$ and the input graph is a grid.*

To prove Theorem 2, we present a polynomial-time reduction from LINKED PLANAR 3-SAT [37]. Here, we are given a Boolean formula ϕ in 3-CNF with clause set Y and variable set X , and a graph $H = (Y \cup X, F)$ such that the following holds:

- Graph H is the union of a Hamiltonian cycle that first visits all elements of X and then all elements of Y , and the incidence graph of ϕ (i.e., the graph on $Y \cup X$ where $c \in Y$ and $x \in X$ are connected by an edge if and only if variable x appears on clause c).
- Graph H is planar and w.l.o.g. there is an embedding where each edge between a clause and a variable that appears negated in the clause is inside the cycle, and each edge between a clause and a variable that appears non-negated in the clause is outside of the cycle.
- Each variable appears in at most three clauses.

This problem is known to be NP-hard even if H together with its embedding are given in the input [37]. Note that the Hamiltonian cycle can easily be computed from H .

In the following, we first describe a construction that, given an instance of LINKED PLANAR 3-SAT, produces an instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY with a planar graph. Later, we describe how to modify the instance such that the graph is a grid.

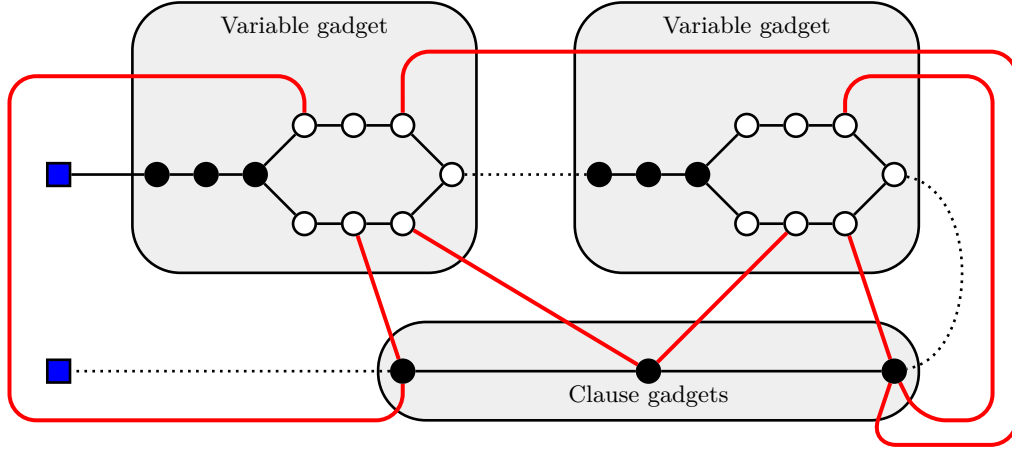
► **Construction 2.** Given an instance (ϕ, H) of LINKED PLANAR 3-SAT, we construct an instance $(G = (V, E), R, O, c)$ of COLLISION-FREE MOVEMENT TO CONNECTIVITY with $k = 2$ as follows. Let the variables X and clauses Y in ϕ be ordered in the way they appear in the Hamiltonian cycle in H .

For each variable $x \in X$, we create a variable gadget $G_x = (V_x, E_x)$ with

- $V_x = \{v_1^x, v_2^x, v_3^x, v_4^x, u_1^x, u_2^x, u_3^x, w_1^x, w_2^x, w_3^x\}$, and
- $E_x = \{\{v_1^x, v_2^x\}, \{v_2^x, v_3^x\}, \{v_3^x, u_1^x\}, \{u_1^x, u_2^x\}, \{u_2^x, u_3^x\}, \{u_3^x, v_4^x\}, \{v_3^x, w_3^x\}, \{w_3^x, w_2^x\}, \{w_2^x, w_1^x\}, \{w_1^x, v_4^x\}\}$.

Furthermore, we place obnoxious robots on the vertices v_1^x, v_2^x, v_3^x . For an illustration of the variable gadget see Figure 8.

We construct the graph G of the COLLISION-FREE MOVEMENT TO CONNECTIVITY instance by first taking the graph H and replacing each vertex corresponding to a variable,



■ **Figure 9** Illustration of an embedding described in the proof of Observation 17 of the graph G produced by Construction 2. The black filled round vertices have obnoxious robots on them, and the blue filled square vertices have main robots on them. There are two variable gadgets on the left and three clause gadgets on the right (indicated by light gray boxes). The red lines correspond to paths of length s that have obnoxious robots on all inner vertices.

say x , with variable gadget G_x in the following way. If two vertices corresponding to variables x, y are adjacent in H , that is, they appear consecutive in the ordering (first x and then y), then we add edge $\{v_4^x, v_1^y\}$ to G . Let x be the last variable in the ordering and let c be the first clause in the ordering, then we add edge $\{v_4^x, v^c\}$ to G and subdivide it $s + 2$ times, where $s = |X| + |Y|$. For each clause $c \in C$ we add call the vertex corresponding to it v^c and place an obnoxious robot on v^c . Let variable x appear in clauses c_1, c_2, c_3 . Then the vertex x in H is connected to vertices $v^{c_1}, v^{c_2}, v^{c_3}$ in H . Let w.l.o.g. the order c_1, c_2, c_3 be the linearization of the cyclic ordering in which the edges incident with x to the vertices c_1, c_2, c_3 are embedded around x in the planar embedding of H , such that all clauses in which x appears non-negated are ordered before the clauses in which x appears negated. For $i \in \{1, 2, 3\}$, if x appears in c_i non-negated, then we connect u_i^x with v^{c_i} with an edge and subdivide the edge s times. Furthermore, we place an obnoxious robot on each vertex created by a subdivision. If x appears in c_i negated, then we connect w_i^x with v^{c_i} with an edge and subdivide the edge s times. Furthermore, we place an obnoxious robot on each vertex created by a subdivision. Finally, we add two vertices r_1, r_2 to G and place a main robot on each of them. Let x be the first variable in the ordering, then we add edge $\{r_1, v_1^x\}$ to G . Let c be the last clause in the ordering, then we add edge $\{r_2, v^c\}$ to G . For an illustration, see Figure 9. Finally, we set $c = 16 \cdot |X| + (s + 2) \cdot (|Y| + 1) + 1$. We will assume in the proofs of correctness that $s^2/4 > c$. Note that this is clearly the case if $|X|$ and $|Y|$ are sufficiently large.

This finishes the construction, which can clearly be performed in polynomial time. It is fairly straightforward to check that the constructed graph G is planar.

► **Observation 17.** *The graph G obtained from Construction 2 is planar.*

Proof. For an illustration, see Figure 9. Recall that H is the union of a Hamiltonian cycle that first visits all variables X and then all clauses Y , and the incidence graph of ϕ . Consider a planar embedding of H where each edge between a clause and a variable that appears negated in the clause is inside the cycle, and each edge between a clause and a variable that

appears non-negated in the clause is outside of the cycle. In Construction 2, when we replace the vertex corresponding to a variable x with variable gadget G_x , we can arrange it such that vertices u_1^x, u_2^x, u_3^x point towards the outside of the cycle and w_1^x, w_2^x, w_3^x to the inside. The gadget G_x itself is clearly planar. Now we can embed the paths in G connecting the clause vertices with the variable gadgets along the edges between clauses and variables in H . Note that in the construction of G , when we connect the variable gadgets with the clause vertices, we connect clause vertices to vertices of the variable gadget in the same cyclic ordering as the edges around the corresponding variable in H . It follows that we can embed the paths connecting variable gadgets and clause vertices in G in a planar way. Finally, vertices r_1, r_2 and their incident edges can clearly be embedded in a planar way. $\blacktriangleleft$

Next, we show correctness of the reduction, that is, the constructed instance (G, R, O, c) is a yes-instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY if and only if (ϕ, H) is a yes-instance of LINKED PLANAR 3-SAT.

► **Lemma 18.** *If (ϕ, H) is a yes-instance of LINKED PLANAR 3-SAT, then (G, R, O, c) is a yes-instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY.*

Proof. Assume we are given a satisfying assignment for ϕ . We create the following movement plan.

- We iterate over the variables in some arbitrary but fixed order. Let x be a variable. If x is set to true, then we move the three obnoxious robots on v_1^x, v_2^x, v_3^x to w_1^x, w_2^x, w_3^x in a collision-free way using 9 moves: We first move the robot on v_3^x to w_3^x , then we move the robot on v_2^x to w_2^x , and finally we move the robot on v_1^x to w_1^x . It is straightforward to check that this requires 9 moves and is collision-free. If x is set to false, then we move the three obnoxious robots on v_1^x, v_2^x, v_3^x to u_1^x, u_2^x, u_3^x in an analogous way. Doing this for each variable requires $9 \cdot |X|$ moves in total.
- Let z be the last variable in the ordering (implied by the Hamiltonian cycle). We now move r_1 to v_4^z . Note that in each variable gadget for some variable x , the vertices $v_1^x, v_2^x, v_3^x, v_4^x$ are not occupied by any robot. Furthermore, either vertices u_1^x, u_2^x, u_3^x or vertices w_1^x, w_2^x, w_3^x are not occupied by any robot. It follows that we can move r_1 collision-free to v_4^z using $7 \cdot |X| + 1$ moves.
- Now for each clause c we identify a literal ℓ that satisfies the clause. Let x be the variable of literal ℓ . If x appears in ℓ non-negated, then by construction we have that v^c is connected by a path of length $(s + 1)$ to one of the vertices u_1^x, u_2^x, u_3^x and no other literal of any other clause is connected to the same vertex as ℓ . Furthermore, we have that these vertices are not occupied by any robot. Let v^c be connected to u_i^x with $i \in [3]$. The vertex v^c as well as all inner vertices on the path of length $(s + 1)$ from v^c to u_i^x are occupied by obnoxious robots. We move all robots towards u_i^x by one step, starting with the one closest to u_i^x and ending with the one occupying v^c . This takes $(s + 1)$ moves and is clearly collision-free. Afterwards, vertex v^c is not occupied by any vertex. If x appears in ℓ negated, we have an analogous situation with vertices w_1^x, w_2^x, w_3^x instead of u_1^x, u_2^x, u_3^x . Doing this for each clause requires $(s + 1) \cdot |Y|$ moves in total.
- Let c^* be the last clause in the ordering (implied by the Hamiltonian cycle). We now move r_1 from v_4^z to v^{c^*} . This is collision-free since, after the previous moves, all vertices corresponding to clauses are not occupied by any robot, and it requires $|Y| + s + 2$ moves.

After the above-described movement plan, main robot r_1 occupies vertex v^{c^*} and main robot r_2 still occupies its starting location. Since these two vertices are connected, we have

that the end-configuration of the movement plan forms a connected subgraph. Furthermore, the movement plan clearly has length $16 \cdot |X| + (s + 2) \cdot (|Y| + 1) + 1$. $\blacktriangleleft$

► **Lemma 19.** *If (G, R, O, c) is a yes-instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY, then (ϕ, H) is a yes-instance of LINKED PLANAR 3-SAT.*

Proof. Assume we are given a movement plan $\mathcal{M}$ that is a solution to (G, R, O, c) . We create a satisfying assignment for ϕ as follows.

First, we start with the following observation: If a main robot moves (collision-free) along one of the paths of length $(s + 1)$ from a vertex of a variable gadget to a clause vertex, all obnoxious robots need to move out of the way. This requires at least $2 \cdot \sum_{i=1}^{\lfloor s+1 \rfloor} i > s^2/4 > c$ moves. Hence, we can conclude that in the movement plan $\mathcal{M}$, no main robot moves along such a path. It follows that each variable gadget is traversed by one of the main robots, and each clause vertex is traversed by one of the main robots. In particular, we can conclude that the main robots make at least $7 \cdot |X| + |Y| + s + 3$ moves.

We can conclude the following: In each variable gadget, the three obnoxious robots need to be moved such that one of the main robots can pass through the gadget. Moving them inside the gadget to u_1^x, u_2^x, u_3^x or w_1^x, w_2^x, w_3^x requires 9 moves. Moving them in any other way that allows a main robot to pass (e.g. to an adjacent gadget) costs strictly more than 9 moves. Hence, at least $9 \cdot |X|$ moves are required to move obnoxious robots in the variable gadgets.

Now consider the clause vertices. Moving the obnoxious robot away from the clause vertex requires either moving each vertex on one of the paths of length $(s + 1)$ by one step and hence requires $(s + 1)$ moves in total, or moving the obnoxious robot into a path connected to a different clause vertex, which requires at least $(s + 2)$ moves, or moving the obnoxious robot into a variable gadget, which also requires at least $(s + 2)$ moves. Hence, at least $(s + 1) \cdot |Y|$ moves are required to move obnoxious robots away from the clause vertices.

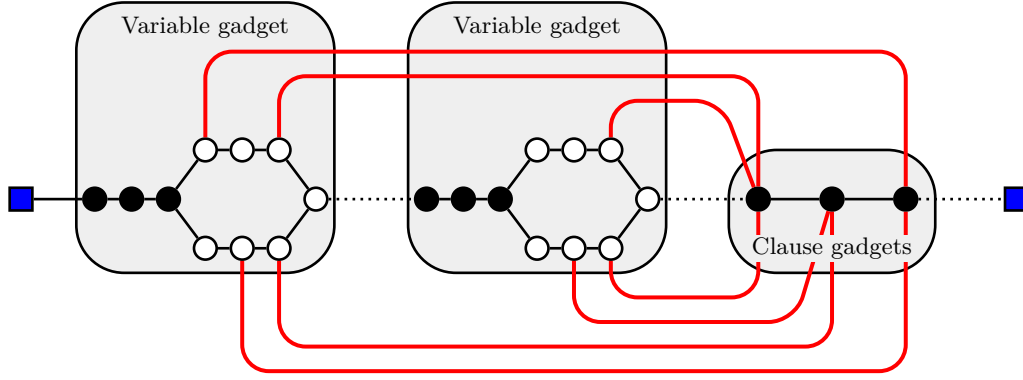
Adding up the above-derived lower bounds for the required moves yields exactly c . We can conclude that in each variable gadget, the three obnoxious robots move to either u_1^x, u_2^x, u_3^x or w_1^x, w_2^x, w_3^x . Furthermore, for each clause vertex, all robots from the vertex along one of the paths of length $(s + 1)$ to a variable gadget are moved by one step each.

We construct an assignment for the variables in X as follows. For each $x \in X$, if three obnoxious robots move to u_1^x, u_2^x, u_3^x , then we set x to true. Otherwise, we set x to false. We claim that this is a satisfying assignment for ϕ .

Assume for contradiction that this is not a satisfying assignment. Then there is a clause c that is not satisfied. Consider the corresponding clause vertex v^c . If none of c 's literals are satisfied, then by construction the vertices of the variable gadgets that are connected to v^c by paths of length $(s + 1)$ are occupied by obnoxious robots. It follows that at least $(s + 2)$ moves are necessary to move the obnoxious robot from v^c . Then, however, by the arguments given above, the total number of moves is strictly larger than c , a contradiction. $\blacktriangleleft$

From Observation 17 and Lemmas 18 and 19 follows that COLLISION-FREE MOVEMENT TO CONNECTIVITY is NP-hard even if $k = 2$ and the input graph is planar. In the following, we show how to modify Construction 2 to produce a graph that is a grid.

First, we argue that we can find an alternative embedding for G by “unfolding” it, such that r_1 and r_2 lie on the outer face, see Figure 10 for an “unfolded” version of Figure 9. More formally, we can “break” the Hamiltonian cycle in H between the first variable vertex and the last clause vertex, creating a Hamiltonian path that first visits all variable vertices and then all clause vertices. We can draw this path along a horizontal line. Then all edges that were outside the cycle are now above the path, and all edges inside the cycle are now



■ **Figure 10** Illustration of a “unfolded” embedding of the graph G produced by Construction 2. The blue filled round vertices have obnoxious robots on them, and the black filled square vertices have main robots on them. The red edges correspond to paths of length s that have obnoxious robots on all inner vertices.

below the path. We take this embedding as a starting point to construct a modified graph G' that is a grid (together with an embedding that shows this). Note that there are three main obstacles to overcome:

- The clause vertices in G may have degree 5, which is not possible in a grid. Hence, we have to modify the vertices to “clause gadgets” that are grids.
- All paths connecting clause gadgets to variable gadgets need to have the same length in order for the reduction to work.
- The path between the last variable gadget and the first clause gadget has to be sufficiently long (that is, longer than any path from a clause gadget to a variable gadget) for the reduction to work.

Formally, we construct G' as follows.

► **Construction 3.** Given an instance (ϕ, H) of LINKED PLANAR 3-SAT, we construct an instance $(G' = (V', E'), R', O', c')$ of COLLISION-FREE MOVEMENT TO CONNECTIVITY with $k = 2$ as follows. Let the variables X and clauses Y in ϕ be ordered in the way they appear in the Hamiltonian cycle in H .

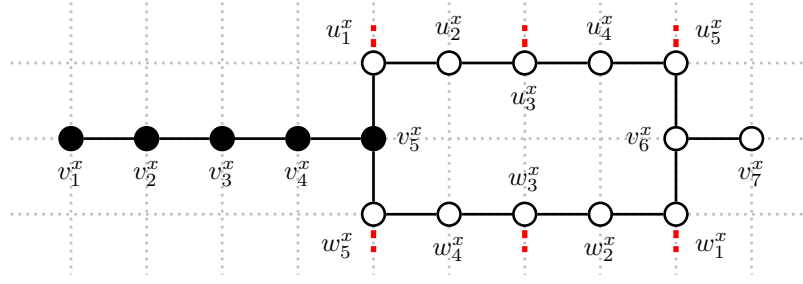
For each variable $x \in X$, we create a variable gadget $G'_x = (V'_x, E'_x)$ with

- $V'_x = \{v_1^x, v_2^x, v_3^x, v_4^x, v_5^x, v_6^x, v_7^x, u_1^x, u_2^x, u_3^x, u_4^x, u_5^x, w_1^x, w_2^x, w_3^x, w_4^x, w_5^x\}$, and
- $E'_x = \{\{v_1^x, v_2^x\}, \{v_2^x, v_3^x\}, \{v_3^x, v_4^x\}, \{v_4^x, v_5^x\}, \{v_5^x, u_1^x\}, \{u_1^x, u_2^x\}, \{u_2^x, u_3^x\}, \{u_3^x, u_4^x\}, \{u_4^x, u_5^x\}, \{u_5^x, v_6^x\}, \{v_6^x, w_5^x\}, \{w_5^x, w_4^x\}, \{w_4^x, w_3^x\}, \{w_3^x, w_2^x\}, \{w_2^x, w_1^x\}, \{w_1^x, v_6^x\}, \{v_6^x, v_7^x\}\}$.

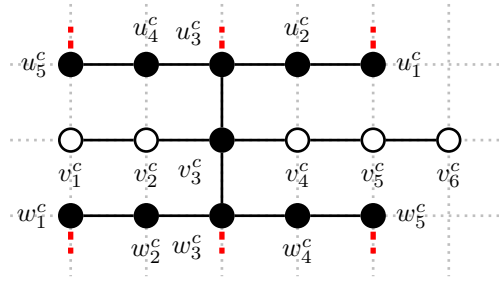
Furthermore, we place obnoxious robots on the vertices $v_1^x, v_2^x, v_3^x, v_4^x, v_5^x$. For an illustration of the variable gadget see Figure 11.

For each clause $c \in Y$, we create a clause gadget $G'_c = (V'_c, E'_c)$ with

- $V'_c = \{v_1^c, v_2^c, v_3^c, v_4^c, v_5^c, v_6^c, u_1^c, u_2^c, u_3^c, u_4^c, u_5^c, w_1^c, w_2^c, w_3^c, w_4^c, w_5^c\}$, and
- $E'_c = \{\{v_1^c, v_2^c\}, \{v_2^c, v_3^c\}, \{v_3^c, v_4^c\}, \{v_4^c, v_5^c\}, \{v_5^c, u_1^c\}, \{u_1^c, u_2^c\}, \{u_2^c, u_3^c\}, \{u_3^c, u_4^c\}, \{u_4^c, u_5^c\}, \{u_5^c, w_5^c\}, \{w_5^c, w_4^c\}, \{w_4^c, w_3^c\}, \{w_3^c, w_2^c\}, \{w_2^c, w_1^c\}\}$.



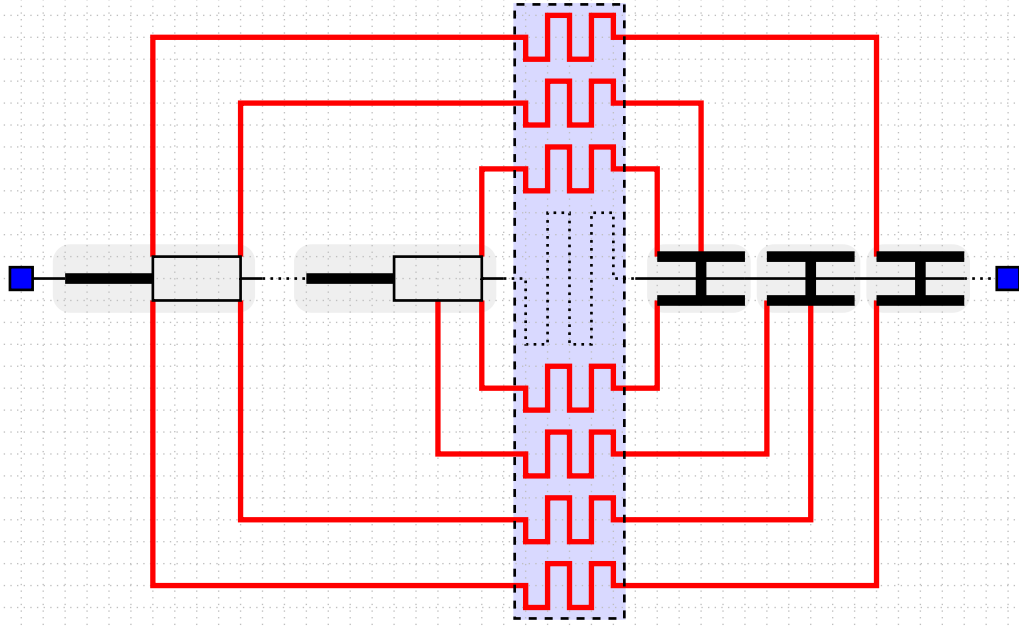
■ **Figure 11** Illustration of a variable gadget G'_x for a variable x (Construction 3). The black filled vertices have obnoxious robots on them. Red dashed lines indicate where the paths between variable gadgets and clause gadgets are connected.



■ **Figure 12** Illustration of a clause gadget G'_c for a clause c (Construction 3). The black filled vertices have obnoxious robots on them. Red dashed lines indicate where the paths between variable gadgets and clause gadgets are connected.

Furthermore, we place obnoxious robots on the vertices $v_3^c, u_1^c, u_2^c, u_3^c, u_4^c, u_5^c, w_1^c, w_2^c, w_3^c, w_4^c, w_5^c$. For an illustration of the clause gadget see Figure 12.

We construct the graph G' of the COLLISION-FREE MOVEMENT TO CONNECTIVITY instance by first taking the graph H and replacing each vertex corresponding to a variable, say x , with variable gadget G'_x in the following way. If two vertices corresponding to variables x, y are adjacent in H , that is, they appear consecutive in the ordering (first x and then y), then we add edge $\{v_7^x, v_1^y\}$ to G' . Let x be the last variable in the ordering and let c be the first clause in the ordering, then we add edge $\{v_7^x, v_1^c\}$ to G' and subdivide it $s + 4$ times, where $s = 90 \cdot (|X| + |Y|) - 4$. For each clause $c \in C$ we replace the corresponding vertex with clause gadget G'_c in the following way. If two clauses c, c' appear consecutive in the ordering (first c and then c'), then we add edge $\{v_6^c, v_1^{c'}\}$ to G . Let variable x appear in clauses c_1, c_2, c_3 . Then the vertex x in H is connected to vertices $v^{c_1}, v^{c_2}, v^{c_3}$ in H . Let w.l.o.g. the order c_1, c_2, c_3 be the linearization of the clockwise cyclic ordering in which the edges incident with x to the vertices c_1, c_2, c_3 are embedded around x in the planar embedding of H , such that all clauses in which x appears non-negated are ordered before the clauses in which x appears negated. For each clause c , let w.l.o.g. the order $\ell_c^1, \ell_c^2, \ell_c^3$ the linearization of the anti-clockwise cyclic ordering in which edges incident with c to the variables corresponding to literals $\ell_c^1, \ell_c^2, \ell_c^3$ are embedded around c . For $i \in \{1, 2, 3\}$, if x appears in c_i non-negated as the j th literal according to the above ordering, then we connect u_i^x with $u_j^{c_i}$ with an edge. If $j \neq 2$ and subdivide the edge s times and otherwise $s + 2$ times. Furthermore, we place an obnoxious robot on each vertex created by a subdivision. If x appears in c_i negated as the j th literal according to the above ordering, then we connect w_i^x with $w_j^{c_i}$ with an edge. If



■ **Figure 13** Illustration of an embedding described in the proof of Lemma 20 of the graph G' produced by Construction 3. The drawing is not up to scale but rather should give the rough idea on how the embedding looks like. The grid points on the thick black lines have obnoxious robots on them, and the blue filled square vertices have main robots on them. The red edges correspond to long paths that have obnoxious robots on all inner vertices. The light blue area surrounded by the black dashed line indicates the part of the grid, where we make sure that all red paths have the same length, and the path connecting the last variable gadget with the first clause gadget is sufficiently long. This is also indicated by the snake-like shape of the paths in the area.

$j \neq 2$ and subdivide the edge s times and otherwise $s + 2$ times. Furthermore, we place an obnoxious robot on each vertex created by a subdivision. Finally, we add two vertices r_1, r_2 to G and place a main robot on each of them. Let x be the first variable in the ordering, then we add edge $\{r_1, v_1^x\}$ to G . Let c be the last clause in the ordering, then we add edge $\{r_2, v_6^c\}$ to G . For an illustration, see Figure 13. Finally, we set $c = 37 \cdot |X| + (s + 8) \cdot |Y| + s + 5$. We will assume in the proofs of correctness that $s^2/4 > c$. Note that this is clearly the case if $|X|$ and $|Y|$ are sufficiently large.

This finishes the construction, which can clearly be performed in polynomial time. In the following, we argue that G' is indeed a grid.

► **Lemma 20.** *The graph G' obtained from Construction 3 is a grid.*

Proof. For an illustration see Figure 13. The main idea of the embedding is as follows. It has a very similar structure as the embedding of G (by Construction 2) illustrated in Figure 10. Informally speaking, we arrange the variable gadgets in a line (more specifically, according to Figure 11 and the next leftmost vertex of the next gadget connects via a horizontal edge to the rightmost vertex of the current gadget). Afterwards, we leave a horizontal “gap” of size $10 \cdot (|X| + |Y|)$ and then we arrange the clause gadgets (Figure 12) in a line. This gap corresponds to the horizontal size of the light blue area in Figure 13, where the paths make the snake-like shaped. Informally speaking, this gap is large enough (and the paths are sufficiently far apart from each other), that we can realize paths of length s , $s + 2$, or $s + 4$

no matter from which vertices exactly they start and finish, and no matter how far up or down from the middle we embed those paths.

Formally, we start replacing the variable gadget of G' in the way shown in Figure 11 on the grid, such that the vertex v_1^x for the first variable x according to the ordering is placed on grid point $(0, 0)$. Then we place the second variable gadget next to it, formally also in the way depicted in Figure 11 such that $v_1^{x'}$ (where x' is the second variable) is placed on grid point $(10, 0)$, and so on. After the last variable gadget, we keep $10 \cdot (|X| + |Y|)$ free grid points (horizontally). This corresponds to the horizontal size of the light blue area on Figure 13. Then we embed the clause gadgets according to Figure 12. Formally, the first clause gadget is positions such that v_1^c (where c is the first clause) is placed on grid point $(10 \cdot |X| + 10 \cdot (|X| + |Y|), 0)$, and so on. Note that by Construction 3, the last variable gadget and the first clause gadget are connected by a path of length $s + 4 = 90 \cdot (|X| + |Y|)$. This we can do by arranging the path snake-like, going 4 grid-cells up and 4 grid-cells down, as depicted in Figure 13. We embed each path connecting a clause gadget with a variable gadget as follows. Assume that the path connects the j th literal (for $j \in [3]$) of the i th clause with a variable gadget. Assume that the literal is non-negated. Then we embed the path upward using $10 \leq 10 \cdot (3i + j - 3) \leq 30 \cdot |Y|$ grid points. Note that this ensures that any two paths are at least 10 horizontal grid points apart from each other. The vertex of the variable gadget to which the path connects is at most $6 \cdot |Y| + 10 \cdot |X| + 10 \cdot (|X| + |Y|)$ and at least $10(|X| + |Y|)$ grid points to the left. Note that $66 \cdot |Y| + 10 \cdot |X| + 10 \cdot (|X| + |Y|) \leq s$, hence embedding a path like this never creates a path that is already too long. If the path is too short when embedded like this, we can add snake-like shapes going 4 grid-cells up and 4 grid-cells down (note that does not interfere with any other path, since they are at least 10 grid cells above or below) until the path has the correct length. Here it is important to see that the connecting points of variable gadgets and clause gadgets have an even horizontal distance and the “initial” embedding has even length. By adding snake-like shaped, we can make the path longer by an arbitrary even number that is smaller than $s + 4$. Hence, we can ensure all paths can be embedded in the described way by adding an appropriate amount of snake-like shapes. If the literal is negated. Then we embed the path downward using $10 \leq 10 \cdot (3i + j - 3) \leq 30 \cdot |Y|$ grid cells. The rest of the embedding is analogous. ◀

Now we have all ingredients to prove Theorem 2.

Proof of Theorem 2. By Lemma 20 we have that the graph produced by Construction 3 is a grid. Clearly, there are only two main robots. The correctness can be shown analogously to Lemmas 18 and 19 by adjusting the movement steps needed in the gadgets and to traverse the gadgets. We omit the details. ◀

5 Collision Free Movement on Grids

In this section, we present parameterized algorithms for COLLISION-FREE MOVEMENT TO Π parameterized by $k + c$ on grids. Specifically, we give kernelization algorithms that produce problems kernels of size $O(k \cdot c^2)$ for COLLISION-FREE MOVEMENT TO Π and $O(\min\{k \cdot c^2, k^2 + c^2\})$ for COLLISION-FREE MOVEMENT TO CONNECTIVITY. This means, if $c \in \omega(\sqrt{k})$, we have a strictly better kernel size for COLLISION-FREE MOVEMENT TO CONNECTIVITY. These kernelization algorithms can then e.g. be combined with Theorem 16 to obtain an algorithm with a running time in $(k \cdot c)^{O(c)} + n^{O(1)}$.

Kernelization. To obtain the kernelization results, we give a number of straightforward reduction rules. A *reduction rule* is a description of a procedure that takes a problem instance as input and outputs an instance of the same problem. We say that a reduction rule is *safe*, if it holds that the input instance is a yes-instance if and only if the output instance is a yes-instance. First, we prove Theorem 3 (restated below).

► **Theorem 3.** *COLLISION-FREE MOVEMENT TO Π parameterized by $k+c$ admits a polynomial kernel of size $O(k \cdot c^2)$ if the input graph is a grid.*

To prove Theorem 3, we present the following simple reduction rule. Essentially, it allows us to remove vertices that are too far away from all main robots. The intuition here is that if a vertex v has distance more than c to any main robot starting location, then it cannot be visited by any main robot, and it also cannot be visited by any obnoxious robot that *needs* to move in order to avoid collision.

► **Reduction Rule 1.** Let $(G = (V, E), R, O, c)$ be an instance of COLLISION-FREE MOVEMENT TO Π . If there is a $v \in V$ such that for all $r \in R$ it holds that $\text{dist}_G(v, r) > c$, then remove v from G (and from O , if it is contained in O).

Reduction Rule 1 can clearly be applied in polynomial time and it will also be useful in the algorithms we present in later sections. Next, we prove that it can be safely applied.

► **Lemma 21.** *Reduction Rule 1 is safe.*

Proof. Assume for contradiction that there is a vertex v that Reduction Rule 1 would remove and to or from which a robot is moved in a solution to the instance. Assume that the solution has a minimum number of movements. Note that this cannot be a main robot, since all of them have distance more than c to v and hence cannot be moved to v . It follows that an obnoxious robot is moved to v (or moved away from v). In a solution with a minimum number of movements, an obnoxious robot is only moved to avoid a collision with a main robot or another obnoxious robot. In the latter case, this property holds recursively for the other obnoxious robot. We can conclude that in the end-configuration, there is a path from v to a vertex v' with a main robot such that all vertices of the path (except v') have obnoxious robots on them and each one has been moved by at least one step. Let ℓ' be the number of obnoxious robots that have been moved. Then the distance of v' to any main robot starting location is at least $c - \ell' + 1$. It follows that moving a main robot to v' and moving the ℓ' obnoxious robots by one step each requires more than c movements in total, a contradiction. ◀

Now we can prove Theorem 3.

Proof of Theorem 3. We apply Reduction Rule 1 exhaustively, which can clearly be done in polynomial time. By Lemma 21 we have that the rule is safe. The definition of Reduction Rule 1 implies that we keep a ball of radius c around every vertex that has a main robot. In a grid, a ball of radius c around a vertex contains $O(c^2)$ vertices. It follows that after exhaustively applying Reduction Rule 1, the remaining graph G has $O(k \cdot c^2)$ vertices. ◀

Now we refine the kernelization algorithm for COLLISION-FREE MOVEMENT TO CONNECTIVITY by introducing some additional reduction rules with which we can achieve a smaller kernel if $k < c^2$. This allows us to prove Theorem 4 (restated below).

► **Theorem 4.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by $k + c$ admits a polynomial kernel of size $O(\min\{k \cdot c^2, k^2 + c^2\})$ if the input graph is a grid.*

To show Theorem 4, we introduce an additional reduction rule. Informally, this rule says that if a vertex v is too far away (distance more than $c + k$) from *some* main robot r 's starting location, then it is irrelevant.

► **Reduction Rule 2.** Let $(G = (V, E), R, O, c)$ be an instance of COLLISION-FREE MOVEMENT TO CONNECTIVITY. If there is a $v \in V \setminus R$ such that there exists $r \in R$ such that $\text{dist}_G(v, r) > c + k$, then remove v from G (and from O , if it is contained in O).

Reduction Rule 2 can clearly be applied in polynomial time. Next we prove that it is safe.

► **Lemma 22.** *Reduction Rule 2 is safe.*

Proof. Assume for contradiction that there is a vertex v that Reduction Rule 2 would remove and to or from which a robot is moved in a solution to the instance. Assume that the solution has a minimum number of movements. Let $r \in R$ such that $\text{dist}_G(v, r) > c + k$. Assume that an obnoxious robot is moved to v (or moved away from v). In a solution with a minimum number of movements, an obnoxious robot is only moved to avoid a collision with a main robot or another obnoxious robot. In the latter case, this property holds recursively for the other obnoxious robot. We can conclude that in the end-configuration, there is a path from v to a vertex v' with a main robot such that all vertices of the path (except v') have obnoxious robots on them and each one has been moved by at least one step. Let ℓ' be the number of obnoxious robots that have been moved. If a main robot is moved to or from v , set $\ell' = 0$ and $v = v'$ in the following argument. Then the distance of v' to r is at least $c + k - \ell' + 1$. Note that at most k vertices on any path from v' to r can have main robots on them. It follows that connecting the main robot with starting location r to the main robot on v' and moving the ℓ' obnoxious robots by one step each requires more than c movements in total, a contradiction. ◀

Now we can prove Theorem 4.

Proof of Theorem 4. We apply Reduction Rules 1 and 2 exhaustively, which can clearly be done in polynomial time. By Lemmas 21 and 22 we have that the rules are safe. By Theorem 3 we know that after exhaustively applying Reduction Rule 1, the remaining graph G has $O(k \cdot c^2)$ vertices. After applying Reduction Rule 2 we have that in particular, for any fixed $r \in R$, we have that for all $v \in V \setminus R$ it holds that $\text{dist}_G(v, r) \leq c + k$. It follows that we keep a ball of radius $c + k$ around r that contains all vertices except possibly some vertices in R . In a grid, a ball of radius $c + k$ around a vertex contains $O(c^2 + k^2)$ vertices. It follows that after exhaustively applying Reduction Rule 2, the remaining graph G has $O(k^2 + c^2)$ vertices. The theorem statement follows. ◀

Kernelization Lower Bounds. Now we present several kernelization lower bounds. We use the popular OR-cross-compositions framework [4, 25] to refute the existence of a polynomial kernel for a parameterized problem under the assumption that $\text{coNP} \not\subseteq \text{NP/poly}$, the negation of which would cause a collapse of the polynomial-time hierarchy to the third level. In order to formally define OR-cross-compositions, we introduce equivalence relations first. An equivalence relation R on the instances of some problem L is a *polynomial equivalence relation* if

1. one can decide for every two instances in time polynomial in their sizes whether they belong to the same equivalence class, and
2. for each finite set S of instances, R partitions the set into at most $(\max_{x \in S} |x|)^{\mathcal{O}(1)}$ equivalence classes.



■ **Figure 14** Illustration of the OR-cross-composition from Theorem 5. Blue filled square vertices have main robots on them.

An *OR-cross-composition* of a problem $L \subseteq \{0, 1\}^*$ into a parameterized problem P (with respect to a polynomial equivalence relation R on the instances of L) is an algorithm that takes t R -equivalent instances $x_1, \dots, x_t$ of L and constructs in time polynomial in $\sum_{i=1}^t |x_i|$ an instance (y, k) of P such that

1. $k \leq (\max_{i \in [t]} |x_i| + \log t)^{O(1)}$, and
2. (y, k) is a yes-instance of P if and only if there is an $i \in [t]$ s.t. x_i is a yes-instance of L .

If an NP-hard problem L OR-cross-composes into a parameterized problem P , then P does not admit a polynomial kernel, unless $\text{coNP} \subseteq \text{NP/poly}$ [4, 25].

First, we show Theorem 5 (restated below), which implies that we cannot get similar kernelization results as Theorems 3 and 4 for planar graphs.

► **Theorem 5.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by c does not admit a polynomial kernel unless $\text{coNP} \subseteq \text{NP/poly}$, even if the input graph is planar and $k = 2$.*

Proof. We provide an OR-cross-composition from LINKED PLANAR 3-SAT [37] based on Construction 2 that is used in the proof of Theorem 2. We say that two instances of LINKED PLANAR 3-SAT are equivalent if the two formulas have the same number of variables and the same number of clauses. Assume we are given t equivalent instances $x_1, x_2, \dots, x_t$ of LINKED PLANAR 3-SAT. We apply Construction 2 to each instance of LINKED PLANAR 3-SAT. We identify the vertices r_1 and r_2 (the start positions of the main robots) of all instances. Then each edge incident with r_1 and r_2 we subdivide $s + 2 = |X| + |Y| + 2$ times, where $|X|$ and $|Y|$ are the number of variables and clauses, respectively. We set $c = 16 \cdot |X| + (s + 2) \cdot (|Y| + 3) + 1$. This finishes the construction. For an illustration see Figure 14.

The edge subdivisions ensure that it is too expensive to move obnoxious robots between the instances. Intuitively, the allowed movement amount forces r_1 and r_2 to meet through one of the instances. The formal correctness is analogous to Lemmas 18 and 19. ◀

Next, we prove Theorem 6 (restated below), that is, we presumably cannot drop the parameter k from the kernelization results in Theorems 3 and 4.

► **Theorem 6.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by c does not admit a polynomial kernel unless $\text{coNP} \subseteq \text{NP/poly}$, even if the input graph is a grid.*

Proof. We provide an OR-cross-composition from LINKED PLANAR 3-SAT [37] based on Construction 3 that is used in the proof of Theorem 2. We say that two instances of LINKED

PLANAR 3-SAT are equivalent if the two formulas have the same number of variables and the same number of clauses. Assume we are given t equivalent instances $x_1, x_2, \dots, x_t$ of LINKED PLANAR 3-SAT. We apply Construction 3 to each instance of LINKED PLANAR 3-SAT with one small modification: we do not place any obnoxious robots in the variable gadgets. Note that the created instances all have the same horizontal size when embedded into a grid using the description on Lemma 20. We place the instances on top of each other (aligned) such that they do not overlap, see Figure 15 for an illustration. Then in each instance, we subdivide the edges incident with r_1 and r_2 , respectively, $c+1$ times, where we specify c later, respectively. We put a main robot on each newly created vertex. We embed the new vertices produced by the edge subdivisions horizontally to the left and right, respectively, from the original positions of r_1 and r_2 , respectively. Now we create a vertical path connecting all r_1 -vertices of all instances and a vertical path connecting all r_2 -vertices of all instances. We place main robots on all vertices of those two paths. For an illustration see Figure 15. Now, for each instance, we create one additional path with main robots on each vertex as follows. Let (x, y) be the coordinates of the original grid point of the r_1 -vertex of an instance (before the edge subdivisions). Then we start a path at grid point $(x-1, y-1)$ of length $k' < c-1$ horizontally to the left, where we specify k' later. Furthermore, we connect the vertex on grid point $(x-1, y-1)$ to the one on grid point $(x-1, y)$.

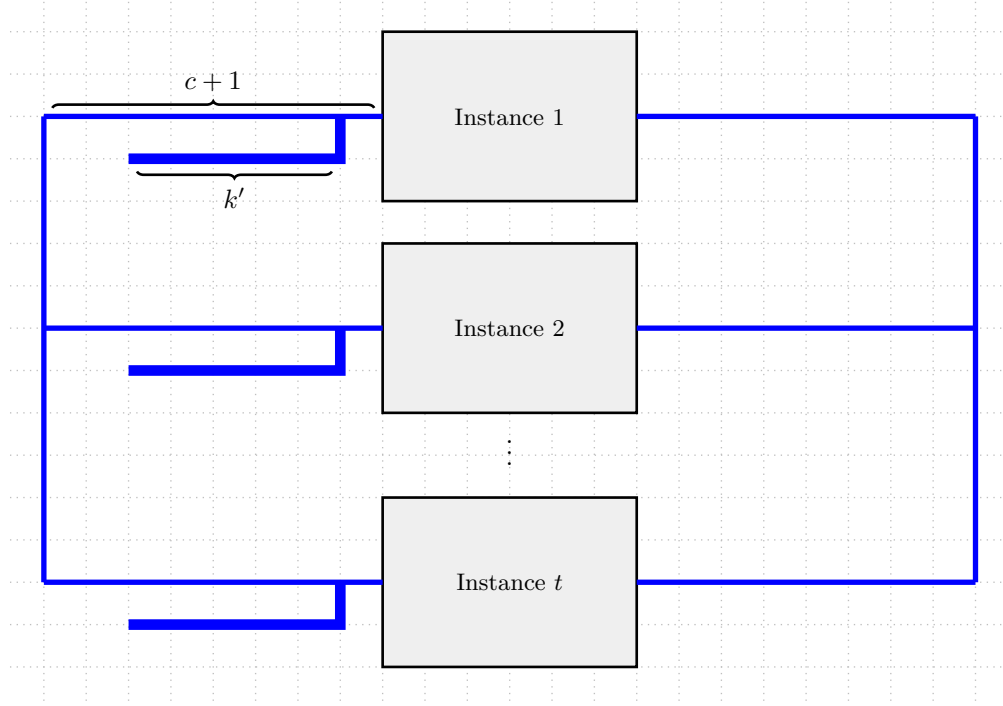
The intuition here is that instead of having two main robots meet from each side of an instance constructed by Construction 3, we create a connection between the main robots on the left side of the instances and the main robots on the right side of the instances by moving k' robots into one of the instances. This is why the obnoxious robots in the variable gadgets are removed. The main robots will play the role of the obnoxious robots instead. This also means that k' essentially needs to equal the number of steps made by main robots in the reduction of Theorem 2. Since the other paths on both sides of the instances have length $c+1$, main robots on those paths cannot be moved, since broken connections cannot be repaired with less than c moves.

To finish the construction, we set $k' = 12 \cdot |X| + 6 \cdot |Y| + s + 5$, where $|X|$ and $|Y|$ are the numbers of variables and clauses, respectively, and $s = 90 \cdot (|X| + |Y|) - 4$. We set $c = (k')^2 + (s+2) \cdot |Y|$.

Note that for the main robots on the left and the right side of the instances to be connected, a connection through one of the instances has to be established. The formal correctness is analogous to Theorem 2 with a few minor differences: If one of the instances is a yes-instance, then we can move the main robots on the path of length k' next to the instance through the instance using $(k')^2$ moves. In the variable gadget, the main robots assume the role of the obnoxious robots in the original reduction, that is, they leave the same vertices in the variable gadgets unoccupied as the obnoxious robots, such that the obnoxious robots on the paths from clause gadgets to variable gadgets can move if the variable is set to a truth value that satisfied the corresponding literal of the clause. Those are $(s+2) \cdot |Y|$ moves in total.

For the other direction, also analogous arguments hold with the following modification. We argue that the main robots cannot take “short-cuts” through paths connecting variable gadgets and clause gadgets since moving all obnoxious robots out of the way and moving main robots into the path costs at least $2 \cdot (s+2)^2$ moves. For large enough $|X|$ and $|Y|$ we have that $2 \cdot (s+2)^2 > c$, and hence this is not possible. It follows that we can use the same arguments for the correctness as in the proof of Theorem 2. ◀

Finally, we prove Theorem 7 (restated below), that is, we presumably cannot improve the kernel size from Theorem 4. For this kernelization lower bound, we use the weak cross-



■ **Figure 15** Illustration the OR-cross-composition from Theorem 6. The grid points on blue paths have main robots on them. The thick blue paths have the main robots on them that would move through the corresponding instance.

compositions framework [25, 30] to refute the existence of a polynomial kernel of a certain size for a parameterized problem under the assumption that $\text{coNP} \not\subseteq \text{NP/poly}$. A *weak cross-composition* of a problem $L \subseteq \{0, 1\}^*$ into a parameterized problem P (with respect to a polynomial equivalence relation R on the instances of L) is an algorithm that takes t R -equivalent instances $x_1, \dots, x_t$ of L and constructs in time polynomial in $\sum_{i=1}^t |x_i|$ an instance (y, k) of P such that

1. $k \leq (\max_{i \in [t]} |x_i|)^{O(1)} \cdot t^{\frac{1}{d}}$ for some $d \in \mathbb{N}$, and
2. (y, k) is a yes-instance of P if and only if there is an $i \in [t]$ s.t. x_i is a yes-instance of L .

If an NP-hard problem L weakly cross-composes into a parameterized problem P , then P does not admit a polynomial kernel of size $O(k^{d-\varepsilon})$ for any $\varepsilon > 0$, unless $\text{coNP} \subseteq \text{NP/poly}$ [25, 30].

► **Theorem 7.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by $k + c$ does not admit a polynomial kernel of size $O((k + c)^{2-\varepsilon})$ for any $\varepsilon > 0$ unless $\text{coNP} \subseteq \text{NP/poly}$, even if the input graph is a grid.*

Proof. We provide a weak cross-composition from LINKED PLANAR 3-SAT [37] based on Construction 3 that is used in the proof of Theorem 2. We say that two instances of LINKED PLANAR 3-SAT are equivalent if the two formulas have the same number of variables and the same number of clauses. Assume we are given t equivalent instances $x_1, x_2, \dots, x_t$ of LINKED PLANAR 3-SAT. Furthermore, assume that $\sqrt{t} = t'$ for some $t' \in \mathbb{N}$ (if not, we can copy some of the instances sufficiently many times). We apply Construction 3 to each instance of LINKED PLANAR 3-SAT. We remove the main robots from vertices r_1 and r_2

of all instances. Then in each instance, we subdivide the edges incident with r_1 and r_2 , respectively, $s + 4 = 90 \cdot (|X| + |Y|)$ times, where $|X|$ and $|Y|$ are the number of variables and clauses, respectively. We use the embedding for the instances described in Lemma 20 and embed the new vertices produced by the edge subdivisions horizontally to the left and right, respectively, from the original positions of r_1 and r_2 , respectively.

Let $x_{\max} = \max_{i \in [t]} |x_t|$. We arrange the t instances in a $(\sqrt{t} \times \sqrt{t})$ -“super grid” where each cell has size $(x_{\max} + 4) \times (x_{\max} + 4)$ and place one of the instances into each super grid cell so that there are at least 2 empty rows / columns of grid cells between each instance and the boundaries of the super grid cells. For an illustration see Figure 16. Now we place two vertices r_1^* and r_2^* at the bottom-left and top-right corner of the super grid, respectively, and place main robots on them.

We create paths from r_1^* horizontally (to the right) and vertically (to the top) of length $\sqrt{t} \cdot (x_{\max} + 4) - 1$, that is, they reach just short of the bottom-right and top-left corner of the super grid. From the vertical path, we create horizontal paths (to the left) of length $\sqrt{t} \cdot (x_{\max} + 4) - 1$ along the bottom (plus one grid cell up) of each super grid cell. We connect this path (via vertical paths up) to the r_1 -vertex of each instance in each super grid cell that it traverses. Now we do an analogous construction from r_2^* . We create paths from r_2^* horizontally (to the left) and vertically (to the bottom) of length $\sqrt{t} \cdot (x_{\max} + 4) - 1$, that is, they also reach just short of the bottom-right and top-left corner of the super grid and do not touch the paths starting at r_1^* . From the vertical path, we create horizontal paths (to the left) of length $\sqrt{t} \cdot (x_{\max} + 4) - 1$ along the top (minus one grid cell up) of each super grid cell. We connect this path (via vertical paths down) to the r_2 -vertex of each instance in each super grid cell that it traverses. The path just described are depicted in red in Figure 16.

We set $c = 2 \cdot \sqrt{t} \cdot (x_{\max} + 4) + 2 \cdot (s + 4) + c'$, where c' is the allowed movement amount of an instance x_i with $i \in [t]$ (note that since the instances are equivalent, they all have the same allowed movement amount). This finishes the construction.

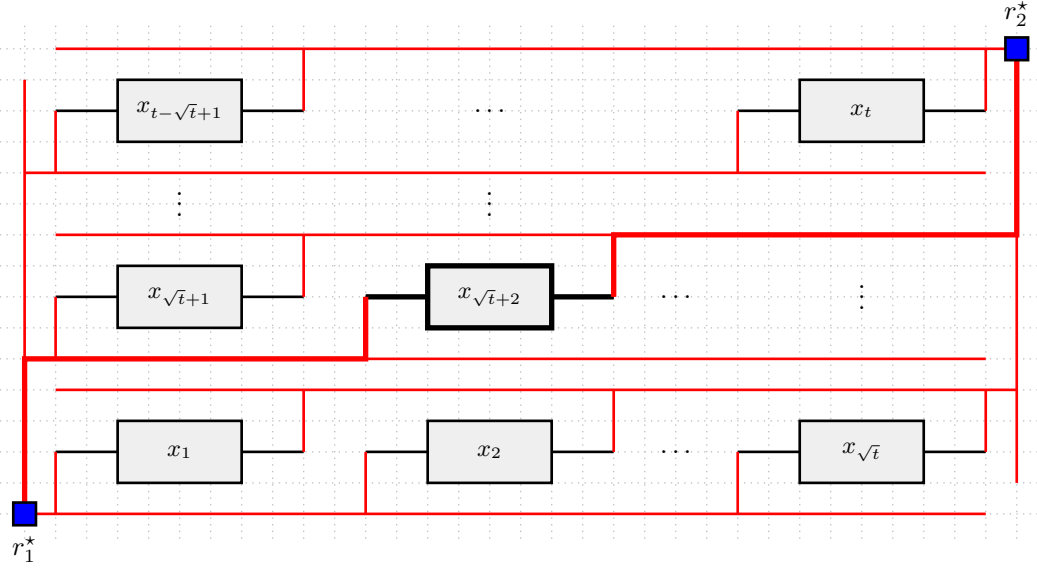
Note that for the main robots on r_1^* and r_2^* , respectively, to meet, they have to traverse at least one of the instances. Assume w.l.o.g. that the two robots meet in an instance (if they do not, we can move the meeting point to r_1 or r_2 with a respective neighbor of one of the instances). In order to reach said instance, they have to travel at least $2 \cdot \sqrt{t} \cdot (x_{\max} + 4)$ along red paths in Figure 16. Furthermore, they have to traverse the edge-subdivision after r_1 and r_2 of said instance, which takes another $2 \cdot (s + 4)$ steps. There are now c' steps remaining to traverse at least one instance. The edge subdivisions ensure that it is too expensive to move oblivious robots between the instances. Intuitively, the allowed movement amount forces r_1 and r_2 to meet through one of the instances. The formal correctness is analogous to Theorem 2. $\blacktriangleleft$

6 Collision Free Movement on Planar Graphs

In this section, we prove Theorem 8 (restated below), that is, we show that COLLISION-FREE MOVEMENT TO Π is fixed-parameter tractable when parameterized by $k + c$ if the input graph is planar.

► **Theorem 8.** *COLLISION-FREE MOVEMENT TO Π is fixed-parameter tractable when parameterized by $k + c$ if the input graph is planar.*

To prove Theorem 8, we show that COLLISION-FREE MOVEMENT TO Π is expressible in monadic second-order logic (MSO). This allows us to employ Courcelle’s famous theorem [2, 5, 6] to obtain the result.



■ **Figure 16** Illustration of the weak cross-composition from Theorem 7. The drawing is not up to scale but rather should give the rough idea on how the embedding looks like. Black boxes correspond to the instances and red lines to the connecting paths. Blue filled square vertices have main robots on them. The thick lines correspond to the paths visited by the main robots when instance $x_{\sqrt{t}+2}$ is selected.

We begin by recalling some basic notation and terminology for relational structures. A *relational vocabulary* is a finite set τ of relation symbols, each of which is associated with a natural number, known as its *arity*. Given any relational vocabulary τ , a τ -*structure* is a pair $\mathcal{A} = (A, \{R^A \mid R \in \tau\})$; A is said to be the *universe* of $\mathcal{A}$ while, for each $R \in \tau$, the *interpretation* R^A of R in $\mathcal{A}$ is a subset of A^r , where r is the arity of R . Here, we are interested only in relational structures with finite universes, and where the maximum arity of any relation is two.

For any vocabulary τ , the set of *first-order formulas* is built up from a countably infinite set of individual variables $x_1, x_2, \dots$, the relation symbols $R \in \tau$, the connectives $\wedge, \vee, \neg$ and the quantifiers $\forall x, \exists x$ ranging over elements of the universe of the structure (for notational convenience we will also use standard shorthand such as $\Rightarrow, \Leftrightarrow, =$, and $\in$). *Monadic second-order logic* (MSO) additionally allows quantification over subsets of the universe via unary relation variables (which we will call *set variables*). Given an MSO-formula ϕ , an individual variable x (respectively a unary relation variable X) appearing in ϕ is said to be a *free variable* if x (respectively X) is not in the scope of a quantifier $\exists x$ or $\forall x$ (respectively $\exists X$ or $\forall X$). We write $\phi(X_1, \dots, X_n, x_1, \dots, x_m)$ for a formula ψ with free relation variables $X_1, \dots, X_n$ and individual variables $x_1, \dots, x_m$. Given subsets $A_1, \dots, A_n \subseteq A$ (formally these define interpretations of unary relation variables over A) and elements $a_1, \dots, a_m \in A$, we write $\mathcal{A} \models \phi(A_1, \dots, A_n, a_1, \dots, a_m)$ to mean that $\mathcal{A}$ satisfies ϕ if the variables $X_1, \dots, X_n, x_1, \dots, x_m$ are interpreted as $A_1, \dots, A_n, a_1, \dots, a_m$ respectively.

A *tree decomposition* of a τ -structure $\mathcal{A}$ with universe A is a pair $(T, \mathcal{B})$, where $T = (V_T, E_T)$ is a tree and $\mathcal{B} = (B_v)_{v \in V_T}$ is a collection of subsets of A such that:

1. for all $a \in A$, the set $\{v \in V_T : a \in B_v\}$ is non-empty and induces a connected subtree in T , and

2. for every relation symbol $R \in \tau$ and every tuple $(a_1, \dots, a_r) \in R^{\mathcal{A}}$, there exists $B_v \in \mathcal{B}$ such that $a_1, \dots, a_r \in B_v$.

As for graphs, the *width* of the tree decomposition $(T, \mathcal{B})$ is $\max_{B_v \in \mathcal{B}} |B_v| - 1$, and the *treewidth* of $\mathcal{A}$ (denoted $\text{tw}(\mathcal{A})$) is the minimum width over all tree decompositions of $\mathcal{A}$.

We now give the formal definition of the MSO MODEL CHECKING problem.

MSO Model Checking

Input: A relational structure $\mathcal{A}$ and an MSO-formula ϕ .

Question: Are there subsets $A_1, \dots, A_n \subseteq A$ and elements $a_1, \dots, a_m \in A$ such that $\mathcal{A} \models \phi(A_1, \dots, A_n, a_1, \dots, a_m)$?

Courcelle's theorem [2, 5, 6] implies the following.

► **Theorem 23** ([2, 5, 6]). *MSO MODEL CHECKING is fixed-parameter tractable when parameterized by $\text{tw}(\mathcal{A}) + |\phi|$.*

We begin by defining the relational structure $\mathcal{A}$ that will encode our instance of COLLISION-FREE MOVEMENT TO II. Assume we are given an instance $(G = (V, E), R, O, c)$ of COLLISION-FREE MOVEMENT TO II. The universe A of $\mathcal{A}$ consists of the following elements:

- The set V of *vertices*.
- The set E of *edges*.
- The set $[c]$ of *steps*.
- The set $S_v = V \times [c]$ of *vertex-step pairs*.
- The set $S_e = V \times [c]$ of *edge-step pairs*.

The structure $\mathcal{A}$ has eight relation symbols (two unary and six binary relations), with the following interpretations:

- The unary relation *main* on V , where $\text{main}(v) \Leftrightarrow v \in R$.
- The unary relation *robot* on V , where $\text{robot}(v) \Leftrightarrow v \in R \cup O$.
- The binary relation *inc* on $V \times E$, where $\text{inc}(v, e) \Leftrightarrow v \in e$.
- The binary relation *vertex* on $S_v \times V$, where $\text{vertex}((v, i), w) \Leftrightarrow v = w$.
- The binary relation *edge* on $S_e \times E$, where $\text{edge}((e, i), f) \Leftrightarrow e = f$.
- The binary relation *step_v* on $S_v \times [c]$, where $\text{step}_v((v, i), j) \Leftrightarrow i = j$.
- The binary relation *step_e* on $S_e \times [c]$, where $\text{step}_e((e, i), j) \Leftrightarrow i = j$.
- The binary relation *nextstep* on $[c] \times [c]$, where $\text{nextstep}(i, j) \Leftrightarrow i = j + 1$.

We now bound the treewidth of $\mathcal{A}$. Let $(T, \mathcal{B})$ be a tree decomposition for G of width $\text{tw}(G)$. We construct a tree decomposition $(T, \mathcal{B}')$ for $\mathcal{A}$, indexed by the same tree $T = (V_T, E_T)$, as follows. Fix a vertex $v \in V_T$, and let B_v be the corresponding element of $\mathcal{B}$. We define the corresponding element of $\mathcal{B}'$ to be

$$B'_v := B_v \cup \{e = \{u, w\} \in E : v, w \in B_v\} \cup \{s = (u, i) \in S_v : u \in B_v\} \cup \{s = (\{u, w\}, i) \in S_e : u, w \in B_v\} \cup [c].$$

It is straightforward to verify that, with this definition, $(T, \mathcal{B}')$ is indeed a tree decomposition for $\mathcal{A}$. Furthermore, it is immediate that $|B'_v| \leq |B_v| + \binom{|B_v|}{2} + c \cdot \binom{|B_v|}{2} + c \cdot |B_v| + c$. Hence, we have the following.

► **Lemma 24.** *It holds that $\text{tw}(\mathcal{A}) \in O(c \cdot \text{tw}(G)^2)$.*

We now proceed to define the formula ϕ . The main idea (intuitively) is as follows.

- For each main robot, we try to find a movement of length c .
- If an obnoxious robot is encountered in a movement, then we try to find a (non-trivial) movement of length c for the obnoxious robot.
- We know that in total, we can only have c non-trivial movements in the movement plan.
- The movement plan should be collision-free and the end-configuration of the main robots should induce a subgraph with property II.
- The length of the movement plan should be at most c .

We introduce several subformulas:

- $\text{movement}(X)$ has as free variable X and checks whether these are exactly c elements from S_v that correspond to a movement.

$$\begin{aligned} \text{movement}(X) = \exists x_1, \dots, x_c \in S_v \Big(\bigwedge_{i \in [c]} (x_i \in X \wedge \text{step}_v(x, i)) \wedge \forall x \in X \bigvee_{i \in [c]} (x = x_i) \wedge \\ \bigwedge_{i \in [c-1]} (\exists v \in V (\text{vertex}(x_i, v) \wedge \text{vertex}(x_{i+1}, v))) \vee \\ \exists e \in E \exists v, w \in V (\text{inc}(v, e) \wedge \text{inc}(w, e) \wedge \text{vertex}(x_i, v) \wedge \text{vertex}(x_{i+1}, w)) \Big) \end{aligned}$$

Note that the size of $\text{movement}(X)$ is in $O(c)$. The formula checks whether there are c elements in S_v with c different steps (note that this implies that the elements are different) and that each element in X is equal to one of these c elements from S_v . This implies that X contains exactly c elements from S_v . Then the formula checks for every pair of vertex-step pairs in X with consecutive time steps whether they have the same vertex or adjacent vertices.

- $\text{collision-free}(X_1, X_2)$ has as free variables X_1 and X_2 , which are expected to be movements, and checks whether they are collision-free.

$$\begin{aligned} \text{collision-free}(X_1, X_2) = \forall x_1 \in X_1 \forall x_2 \in X_2 \forall i \in [c] \exists v, w \in V \Big((\text{step}_v(x_1, i) \wedge \text{step}(x_2, i)) \Rightarrow \\ (v \neq w \wedge \text{vertex}(x_1, v) \wedge \text{vertex}(x_2, w)) \Big) \wedge \\ \forall x_1, x_2 \in X_1 \forall x_3, x_4 \in X_2 \forall i, j \in [c] \forall v, w \in V \Big((\text{nextstep}(i, j) \wedge \\ \text{step}_v(x_1, i) \wedge \text{step}(x_2, j) \wedge \text{step}_v(x_3, i) \wedge \text{step}(x_4, j)) \Rightarrow \\ \neg(\text{vertex}(x_1, v) \wedge \text{vertex}(x_2, w) \wedge \text{vertex}(x_3, w) \wedge \text{vertex}(x_4, v)) \Big) \end{aligned}$$

Note that the size of $\text{collision-free}(X_1, X_2)$ is constant.

- $\text{move}(X, x)$ has as free variables X and x where X is expected to be a movement and x is expected to be an element from S_e . It checks whether the edge-step pair $x = (e, i)$ is a move in the movement, that is, whether at step i there is a move along edge e in the movement X .

$$\begin{aligned} \text{move}(X, x) = \exists x_1, x_2 \in X \exists v, w \in V \exists e \in E \exists i, j \in [c] \Big(\text{nextstep}(i, j) \wedge \text{step}_v(x_1, i) \wedge \text{step}_v(x_2, j) \wedge \\ v \neq w \wedge \text{vertex}(x_1, v) \wedge \text{vertex}(x_2, w) \wedge \text{inc}(v, e) \wedge \text{inc}(w, e) \wedge \text{edge}(x, e) \wedge \text{step}_e(x, i) \Big) \end{aligned}$$

Note that the size of $\text{move}(X, x)$ is constant.

- $\text{start}(X, x)$, $\text{end}(X, x)$, and $\text{visit}(X, x)$ have as free variables X and x , respectively, where X is expected to be a movement and x is expected to be an element from V . $\text{start}(X, x)$ checks whether the movement X starts at vertex x . $\text{end}(X, x)$ checks whether the

movement X ends at vertex x . $\text{visit}(X, x)$ checks whether the movement X visits vertex x at some step.

$$\begin{aligned}\text{start}(X, x) &= \exists x_1 \in X (\text{vertex}(x_1, x) \wedge \text{step}_v(x, 1)) \\ \text{end}(X, x) &= \exists x_1 \in X (\text{vertex}(x_1, x) \wedge \text{step}_v(x, c)) \\ \text{visit}(X, x) &= \exists x_1 \in X (\text{vertex}(x_1, x))\end{aligned}$$

Note that the sizes of $\text{start}(X, x)$, $\text{end}(X, x)$, and $\text{visit}(X, x)$ are each constant.

Now we are ready to give the formula ϕ . Denote with $c^* = \min(k + \ell, c)$.

$$\begin{aligned}\phi(X_1, \dots, X_{c^*}, x_1, \dots, x_c) &= \bigwedge_{i \in [c^*]} \text{movement}(X_i) \wedge \bigwedge_{i, j \in [c^*]} ((i \neq j) \Rightarrow \text{collision-free}(X_i, X_j)) \wedge \\ &\quad \forall v \in V \left((\text{robot}(v) \wedge (\bigvee_{i \in [c^*]} \text{visit}(X_i, v))) \Rightarrow (\bigvee_{i \in [c^*]} \text{start}(X_i, v)) \right) \wedge \\ &\quad \forall x \in S_e \left((\bigvee_{i \in [c^*]} \text{move}(X_i, x)) \Rightarrow (\bigvee_{i \in [c]} (x = x_i)) \right) \wedge \\ &\quad \exists X \subseteq V \left((\forall v \in V ((\text{main}(v) \wedge \bigwedge_{i \in [c^*]} \neg \text{start}(X_i, v))) \vee \right. \\ &\quad \left. \exists w \in V (\text{main}(w) \wedge (\bigvee_{i \in [c^*]} (\text{start}(X_i, w) \wedge \text{end}(X_i, w))) \Leftrightarrow (v \in X))) \wedge \Pi(X) \right)\end{aligned}$$

Note that the size of the formula ϕ is in $O(c^2 + |\Pi|)$. Since the graph property Π is recursively enumerable, we can conclude $|\Pi(X)| = f(k)$ for $|X| = k$ and some computable function f . Hence, we have the following.

► **Lemma 25.** *It holds that $|\phi| \in O(c^2) + f(k)$, for some computable function f .*

It remains to show that the formula is correct.

► **Lemma 26.** *The formula ϕ is satisfiable if and only if $(G = (V, E), R, O, c)$ is a yes-instance of COLLISION-FREE MOVEMENT TO Π .*

Proof. First, assume that ϕ is satisfiable. We show that then $(G = (V, E), R, O, c)$ is a yes-instance of COLLISION-FREE MOVEMENT TO Π . Let $X_1, \dots, X_{c^*}, x_1, \dots, x_c$ denote a satisfying assignment for ϕ . We construct a movement plan $\mathcal{M}$ as follows. We have that the sets X_i with $i \in [c^*]$ satisfy $\text{movement}(X_i)$. It follows that $X_i = \{(v_1, 1), (v_2, 2), \dots, (v_c, c)\}$ (see the first line of the definition of $\text{movement}(X)$). We claim that $M_i = (v_1, v_2, \dots, v_c)$ is a movement. The second and third line of the definition of $\text{movement}(X)$ give us exactly that for all v_j, v_{j+1} with $j \in [c - 1]$ we have that $v_j = v_{j+1}$ or $\{v_j, v_{j+1}\} \in E$. Hence, we can conclude that M_i is a movement. Now for each M_i that is a movement for a robot $r \in R \cup O$, we add M_i to the movement plan $\mathcal{M}$. For each $r \in R \cup O$ such that none of the M_i with $i \in [c^*]$ is a movement for r , we add a trivial movement to the movement plan $\mathcal{M}$. Now we show the following, from which clearly follows that $(G = (V, E), R, O, c)$ is a yes-instance of COLLISION-FREE MOVEMENT TO Π .

1. For each $r \in R \cup O$ there is at most one $i \in [c^*]$ such that M_i is a movement for i , that is, the movement plan $\mathcal{M}$ is well-defined.
2. The movement plan $\mathcal{M}$ is collision-free.
3. $\text{len}(\mathcal{M}) \leq c$ and $G[R^*(\mathcal{M})] \in \Pi$.

Consider the first statement and assume for contradiction that there are two movements M_i and M_j with $i \neq j$ for some $r \in R \cup O$. Note that in ϕ , the subformula $\text{collision-free}(X_i, X_j)$ is checked and evaluates to true, since otherwise ϕ would evaluate to false. In $\text{collision-free}(X_i, X_j)$ it is, in particular, checked, that $(v_1, 1) \in X_i$ and $(v'_1, 1) \in X_j$ implies $v_1 \neq v'_1$. It follows that M_i and M_j cannot both be a movement for the same $r \in R \cup O$.

Consider the second statement and assume for contradiction that there are two movements M_i and M_j with $i \neq j$ for robots $r, r' \in R \cup O$, respectively, that are not collision-free. First, consider that both M_i and M_j are non-trivial. Then in ϕ , the subformula $\text{collision-free}(X_i, X_j)$ is checked and evaluates to true, since otherwise ϕ would evaluate to false. However, if M_i and M_j then there exists some vertex v and some step t such that both r and r' occupy v in step t or there exist an edge $\{v, w\} \in E$ and some step t such that r occupies v in step t and w in step $t + 1$ and r' occupies w in step t and v in step $t + 1$. In this case we can verify that $\text{collision-free}(X_i, X_j)$ evaluates to false: In the former case the first two lines in the definition of $\text{collision-free}(X_1, X_2)$ evaluate to false, and in the latter case, the last three lines in the definition of $\text{collision-free}(X_1, X_2)$ evaluate to false. In each case we reach a contradiction. Now consider the case that w.l.o.g. M_i is a non-trivial movement and M_j is a trivial movement. If M_i and M_j are not collision-free, then there must be a step t such that robot r occupies r' in step t . However, then $\text{robot}(r')$ and $\text{visit}(X_i, r')$ both evaluate to true. It follows that $\bigvee_{j \in [c^*]} \text{start}(X_j, r')$ must also evaluate to true. It follows that M_j corresponds to some X_j and $\text{collision-free}(X_i, X_j)$ evaluates to true. By the same arguments as earlier we can conclude that M_i and M_j are collision-free.

Finally, consider the third statement. Let M_i be a non-trivial movement for some $r \in R \cup O$. First, it is straightforward to verify that if robot r moves along edge $\{v, w\}$ in step t , then $\text{move}(X_i, (\{v, w\}, t))$ evaluates to true. Since the movement plan $\mathcal{M}$ is collision-free according to the previous arguments, we have that for a fixed $(\{v, w\}, t)$, the subformula $\text{move}(X_i, (\{v, w\}, t))$ evaluates to true for at most one X_i . In the third line of the definition of ϕ it is checked that whenever $\text{move}(X_i, (\{v, w\}, t))$ evaluates to true, then $(\{v, w\}, t) = x_j$ for some $j \in [c]$. It follows that the set $\{x_1, x_2, \dots, x_c\}$ contains all moves made by robots according to movement plan $\mathcal{M}$. It follows that there are at most c different moves, and hence $\text{len}(\mathcal{M}) \leq c$. Next, consider the last two lines of the definition of ϕ . We show that the set X here corresponds to $R^*(\mathcal{M})$: The formula ensures that a vertex v is contained in X if and only if v is a main robot and not the start of any X_i (in this case v has a trivial movement), or there is a main robot w and an X_i starting with w that ends in v , that is, X_i corresponds to a movement for a main robot w that ends in v . Finally ϕ checks whether $\Pi(X)$ evaluates to true. It follows that $G[R^*(\mathcal{M})] \in \Pi$.

Now assume that $(G = (V, E), R, O, c)$ is a yes-instance of COLLISION-FREE MOVEMENT TO Π . Then there is a collision-free movement plan $\mathcal{M}$ that contains at most c^* non-trivial movements which make at most c moves. Let $X_1, X_2, \dots, X_{c^*}$ correspond to the non-trivial movements (if there are strictly less than c^* non-trivial movements, we can add sets X that only contain a vertex that is never visited (we can easily ensure that such a vertex always exists)). Let $x_1, x_2, \dots, x_c$ correspond to the moves. If there are strictly less than c moves, then we can duplicate arbitrary moves. By analogous arguments to the first part of this proof, we can argue that $\phi(X_1, X_2, \dots, X_{c^*}, x_1, x_2, \dots, x_c)$ evaluates to true. $\blacktriangleleft$

Now we have all ingredients to prove Theorem 8.

Proof of Theorem 8. Given an instance $(G = (V, E), R, O, c)$ of COLLISION-FREE MOVEMENT TO Π , we first apply Reduction Rule 1. By Lemma 21 this creates an equivalent

instance $(G' = (V', E'), R, O', c)$. We can observe that $\text{diam}(G') \in O(k \cdot c)$ (where if G' is disconnected, we assume that $\text{diam}(G')$ is the maximum diameter of the connected components of G'). This follows from the fact that after Reduction Rule 1 is applied, every vertex in G' has distance at most c from some $r \in R$, and for every $r, r' \in R$ such that r and r' are in the same connected component of G' , we have that $\text{dist}(r, r') \in O(k \cdot c)$. Since G' is planar, it is well-known that this implies that $\text{tw}(G') \in O(k \cdot c)$. Theorem 8 now follows from Lemmas 24–26, and Theorem 23. $\blacktriangleleft$

We remark that the described algorithm does not exploit planarity. It is only used to upper-bound the treewidth with the diameter of the input graph. Hence, the algorithm also implies that COLLISION-FREE MOVEMENT TO Π is fixed-parameter tractable when parameterized by $k + c$ and the treewidth of the input graph combined.

7 Collision Free Movement on Unit Disk Graphs

In this section, we consider the canonical optimization version of the problem COLLISION-FREE MOVEMENT TO CONNECTIVITY, where the objective is to minimize the length of the movement plan. We prove Theorem 9 (restated below), that is, we give a fixed-parameter approximation algorithm for COLLISION-FREE MOVEMENT TO CONNECTIVITY parameterized by $k + c$ on unit disk graphs, that produces a solution with energy at most $2 \cdot \text{OPT} + 3k$, where OPT is the energy of an optimal solution (Theorem 9). Here, the parameter c is the number of moves in an optimal solution. In fact, the algorithm we present also works on a slightly more general graph class, so-called *clique grids*, which are generalizations of grids and defined as follows.

► **Definition 27** (Clique Grid) ([24]). *A graph $G = (V, E)$ is a clique grid graph if there exists a function $f : V \rightarrow [t] \times [t']$ for some $t, t' \in \mathbb{N}$ such that*

- *for all $(i, j) \in [t] \times [t']$ it holds that $f^{-1}(i, j)$ is a clique in G , and*
- *for all $\{u, v\} \in E$ it holds that if $f(u) = (i, j)$ and $f(v) = (i', j')$, then $|i - i'| \leq 2$ and $|j - j'| \leq 2$.*

The function f is a representation of G .

It is known that unit disk graphs are clique grids and that, given the point set of the unit disk graph, a representation can be computed in polynomial time.

► **Lemma 28** ([24]). *Let G be a unit disk graph with point set D . Then G is a clique grid graph and a representation f of G can be computed from D in polynomial time.*

In this section, we will assume that a representation of the input graph in form of a point set or a clique grid representation is given as part of the input. Now we prove the following result.

► **Theorem 9.** *COLLISION-FREE MOVEMENT TO CONNECTIVITY admits a fixed-parameter approximation when parameterized by $k + c$ if the input graph is a unit disk graph, that produces a solution with energy at most $2 \cdot \text{OPT} + 3k$, where OPT is the energy of an optimal solution.*

We present an algorithm to prove Theorem 9 with the following steps. Assume we are given an instance $(G = (V, E), R, O)$ of (the optimization version) of COLLISION-FREE MOVEMENT TO CONNECTIVITY and we have a clique grid representation of G . We first start with some basic “guessing” steps, which in this context means that we exhaustively enumerate and check all possibilities for the guess.

- We first guess the optimal value for c .
- We exhaustively apply Reduction Rule 1, that is, we remove each vertex that has distance more than c from every main robot.
- We guess a tree T with k vertices that witnesses the connectivity in an optimal solution of the instance.
- We guess a bijection between the main robots and the vertices of T .

Next, we define a *continuation type* $t_i(v)$ of a vertex v for step i and we define a *type* of a vertex v as follows:

- For step c , the *continuation type* $t_c(v)$ of vertex v is an element from $\{F, O, r_1, r_2, \dots, r_k\}$ and set as follows. We set $t_c(v) = F$ meaning the vertex is *free* if $v \notin O \cup R$, we set $t_c(v) = O$ meaning the vertex is *occupied by an obnoxious robot* if $v \in O$, and we set $t_c(v) = r_i$ meaning the vertex is *occupied by a main robot* if $v = r_i \in R$.
- For step $i < c$, the *continuation type* $t_i(v)$ of a vertex v is defined as the following set: For each clique C that is neighboring the clique of i , and each v' that is a neighbor of c in C , add the pair $(C, t_{i+1}(v'))$ to $t_i(v)$.
- The *type* of v is a triple consisting of the clique C of v , an element from $\{F, O, r_1, \dots, r_k\}$ indicating whether v is free or occupied (as above), and the list of all continuation types of v for all steps $i \leq c$.
- If there is a vertex type such that there are at most $2k + c + 2$ vertices of that type, we additionally add the identity of the vertices to their type.

Note that since we apply Reduction Rule 1, we have $O(k \cdot c^2)$ different cliques (this follows from an analogous argument as the proof of Theorem 3). Since each clique only has constantly many neighboring cliques, the number of different continuation types is in $2^{O(k \cdot c^2)}$. We can iteratively compute the continuation types of all vertices for all steps in $2^{O(k \cdot c^2)} \cdot n^{O(1)}$ time, starting with step c , in a straightforward way.

The intuition behind the continuation types is as follows. If a robot arrives at a vertex in step c it cannot travel any further afterwards. So the only relevant information is whether the vertex is free or occupied by another robot. In the latter case, the other robot needs to be moved before step c to avoid a collision. If a robot arrives at a vertex at some step $i < c$, it can travel further, so it is relevant which types of vertices it can reach. This is encoded in the list of clique-vertex type pairs. Of course, it is also still relevant whether the vertex is free or occupied by another robot. Note that if a clique-neighbor type pair is in the list of the type of a vertex, it only means that the vertex can reach *some* neighbor in the clique with the type (not all). This, intuitively, is the reason for the approximation factor two, since we can only guarantee that we move a robot to a vertex of the “correct” vertex type in the “correct” clique at each step, but we cannot guarantee that we move the robot to the “correct” vertex. So the solution we construct may need an extra step in each clique to move to the “correct” vertex.

The algorithm now continues with a second guessing step, where we guess up to $\max\{c, k\}$ vectors M_i of vertex types that each have length up to c (possibly including duplicates). Whenever there is a type switch from one position to the next in the vector, we consider this to be a move. We reject all guesses where the total number of moves is greater than c . These vectors will be templates for the movements in the movement plan we construct. In a later step, we will replace the types with actual vertices. The main goal is to first move the main robots into the “correct” cliques, and then afterwards to the “correct” vertices inside the cliques. For a given guess, we make the following checks:

- For each vector M_i , we check whether it is a movement. More specifically, whenever two subsequent entries in the vector are two different types t, t' we check whether the types are *neighboring types*, that is, whether there are two vertices u, v which are neighbors and u has type t , and v has type t' . Furthermore, we check whether the first entry of the vector is a vertex type for a vertex that is occupied by a robot.
- For each vector M_i , we check whether it is collision-free, that is, whenever there is a vertex type t in the vector for a vertex that is occupied by a robot, there is another movement vector M_j starting with type t that moves the robot away (sufficiently early). If several vectors (say ℓ) move a robot to a vertex of type t for a vertex that is occupied by another robot, such that the time steps where the moved robots occupy the vertex of type t intersect, we check whether there are ℓ other movements vector starting with type t that move the robots away (sufficiently early).
- For each pair of vectors M_i, M_j we check whether they are collision-free. This is only relevant if the vectors contain types that have vertex identities.
- We want to find vectors where the number of moves in each step is minimum, that is, moves only happen simultaneously if they have to, that is, when robots move around a cycle of length at least 3.
We check whether at each step across all vectors, there is either at most one move, or at most one cyclic set of moves.

If none of the above checks fail, we check whether T is a subgraph of G such that the location of the vertices of T in G complies with our guess, that is, they have the correct type. To do this, we use a subgraph finding algorithm that allows us to define a cost function for possible mappings from the vertices of T to the vertices of G . We use this to make sure that the subgraph embedding complies with our guesses.

► **Theorem 29** ([12]). *Let F be an undirected graph with s vertices and treewidth t . Let G be an undirected graph with n vertices and let $c : V(F) \times V(G) \rightarrow \mathbb{N}$ be a cost function of mapping a vertex of F to a vertex of G . If F is a subgraph of G , then it is possible to find in time $2^{O(s)} \cdot n^{O(t)}$ a subgraph embedding ϕ that minimizes $c(\phi)$.*

We define a cost function c as follows: For $v \in V(T)$ such that we guessed that v is mapped to main robot r , and the movement for r moves the robot to a vertex of type t , and $v' \in V(G)$, we set

$$c(v, v') = \begin{cases} 1, & \text{if } v' \text{ has type } t, \\ n, & \text{otherwise.} \end{cases}$$

We use the algorithm behind Theorem 29 to find subgraph embedding ϕ for T into G of minimal cost. If the cost is larger than k , we reject the guess.

Now we have all ingredients to construct a movement plan $\mathcal{M}$ for the COLLISION-FREE MOVEMENT TO CONNECTIVITY instance. For each movement $M \in \mathcal{M}$ we have that the first vertex type for a vertex that is occupied by a robot, otherwise the checks described above fail. We pick an arbitrary vertex of that type (a different one for each movement) and identify the robot on that vertex as the one corresponding to the movement. We iterate through the steps from $i = 1$ to c :

- We perform the i th move of each move of each movement. That is, for each $M \in \mathcal{M}$, we pick a (different) vertex v of type $M[i + 1]$, and move the robot corresponding to M from its previous position to v .

When picking the vertices, we avoid collision where two robots use the same edge in opposite directions. This can always be done since there are strictly more than $2k$ vertices of each type that does not include the vertex identity. Recall that if such a collision happens due to types that contain vertex identities, then the third of the above-described checks fails.

- If after the i th move is performed for each movement, a robot r is moved to a vertex v of some type t that is occupied by another robot r' , then we moved robot r to the “wrong” vertex type t .

Since the second of the above-described checks did not fail, there must be another vertex v' of type t that is not occupied by any robot after the i th move. Furthermore, since the last of the above-described checks did not fail, we know that r' was already on v in the previous step.

We introduce an intermediate step (between steps $i - 1$ and i) in which we move robot r' from v to v' .

By construction, the above described movements are collision-free. Furthermore, the resulting movement plan has length at most $2c$ since for every move in the type vectors we add at most one additional move in the second of the above-described steps of the construction.

To ensure that the end-configuration is connected, we have to move the main robots to the vertices specified by the embedding ϕ of the tree T on the main robots. By construction, we have that $\mathcal{M}$ moves each main robot to a vertex that has the same type as its target vertex in the embedding ϕ . Note that all vertices of the same type are in the same clique. In a final step, we move each main robot onto their target vertex using at most three moves.

Assume that there is a main robot r which is not on its target vertex v . If the target vertex v is free, we can move r to that vertex with one additional move. If the target vertex v is occupied by some robot r' , then we pick another vertex v' of the same type that is not occupied by a main robot. Note that since there are at least $2k + c + 2$ vertices of each type (that does not include the vertex identity), such a vertex exists. If v' is free, then we move the robot r' to v' , and afterwards we move r to v . If v' is occupied by an oblivious robot r'' , we perform a cyclic move. Let v'' be the current position of r . We simultaneously move robot r to v , robot r' to v' , and robot r'' to v'' . All described movements are clearly collision-free, contain at most three moves, and reduce the number of main robots that are not on their target vertex by one. Hence, by iterating this procedure, we can move all main robots to their respective target vertices in a collision-free manner by using at most three additional moves per main robot.

Now we are ready to prove Theorem 9.

Proof of Theorem 9. It is straightforward to check that the described guesses create $f(k + c) \cdot n^{O(1)}$ cases, the described checks can be performed in $f(k + c) \cdot n^{O(1)}$ time, and the movement plan can be constructed in $f(k + c) \cdot n^{O(1)}$ time.

As argued during the description, the algorithm clearly produces a collision-free movement plan such that the end-configuration of the main robots is connected.

Now assume there is an optimal solution $\mathcal{M}^*$ of length c to the COLLISION-FREE MOVEMENT TO CONNECTIVITY instance. Then there is a guess such that the each type vector corresponds to one movement in $\mathcal{M}^*$ and the tree T corresponds to a spanning tree in the subgraph of the main robots in their end-configuration. As argued, the movement plan constructed by the algorithm then has length at most $2c + 3k$. It follows that we compute a 2-approximation with an additional additive error of $3k$. ◀

8 Conclusion

To the best of our knowledge, we give the first computational results for the problems COLLISION-FREE MOVEMENT TO Π and COLLISION-FREE MOVEMENT TO CONNECTIVITY. We leave several open questions. We believe that the following are of particular interest.

- Is COLLISION-FREE MOVEMENT TO CONNECTIVITY fixed-parameter tractable when parameterized by c if the input graph is planar?
- Is COLLISION-FREE MOVEMENT TO CONNECTIVITY fixed-parameter tractable when parameterized by $k + c$ if the input graph is a unit disk graph?

It would also be interesting to identify graph classes on which COLLISION-FREE MOVEMENT TO CONNECTIVITY becomes solvable in polynomial time.

- Is COLLISION-FREE MOVEMENT TO CONNECTIVITY solvable in polynomial time if the input graph is a tree?

It is also natural to consider other parameters, such as e.g. the number of obnoxious robots or structural parameters of the input graph such as the treewidth.

Apart from these relatively specific questions, our work opens several natural future work directions. Instead of minimizing the total movement (energy) one could ask to minimize the makespan of the movement plan instead. There are also many other natural graph properties that one could focus on, for example one could require that the target formation admits a perfect matching among the main robots (for an even number of robots). Notably, for two robots this coincides with connectivity, and hence some of our hardness results transfer.

References

- 1 Pankaj K. Agarwal, Mark de Berg, Benjamin Holmgren, Alex Steiger, and Martijn Struijs. Optimal motion planning for two square robots in a rectilinear environment. In *Proceedings of the 41st International Symposium on Computational Geometry (SoCG)*, volume 332 of *LIPIcs*, pages 5:1–5:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.
- 2 Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.
- 3 Piotr Berman, Erik D Demaine, and Morteza Zadimoghaddam. $O(1)$ -approximations for maximum movement problems. In *Proceedings of the 14th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 62–74. Springer, 2011.
- 4 Hans L Bodlaender, Bart MP Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.
- 5 Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and computation*, 85(1):12–75, 1990.
- 6 Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.
- 7 Gruia Cialinescu, Adrian Dumitrescu, and János Pach. Reconfigurations in graphs and grids. *SIAM Journal of Discrete Mathematics*, 22(1):124–138, 2008.
- 8 Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

- 9 Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, Dominik Leko, and M. S. Ramanujan. Routing few robots in a crowded network. *Journal of Computer and System Sciences*, 157:103753, 2026.
- 10 Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. Parameterized algorithms for coordinated motion planning: Minimizing energy. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 297 of *LIPICs*, pages 53:1–53:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.
- 11 Erik D. Demaine, Mohammad Taghi Hajiaghayi, Hamid Mahini, Amin S. Seyed-Roshkhar, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms*, 5(3):30:1–30:30, 2009.
- 12 Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Dániel Marx. Minimizing movement: Fixed-parameter tractability. *ACM Transactions on Algorithms*, 11(2):14:1–14:29, 2014.
- 13 Erik D. Demaine and Mikhail Rudoy. A simple proof that the $(n^2 - 1)$ -puzzle is hard. *Theoretical Computer Science*, 732:80–84, 2018.
- 14 Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer, 2016.
- 15 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.
- 16 Eduard Eiben, Robert Ganian, and Iyad Kanj. The Parameterized Complexity of Coordinated Motion Planning. In *Proceedings of the 39th International Symposium on Computational Geometry (SoCG)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
- 17 Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. A minor-testing approach for coordinated motion planning with sliding robots. In *Proceedings of the 41st International Symposium on Computational Geometry (SoCG)*, volume 332 of *LIPICs*, pages 44:1–44:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.
- 18 Sándor P Fekete, Phillip Keldenich, Dominik Krupke, and Joseph SB Mitchell. Computing coordinated motion plans for robot swarms: The CG: SHOP challenge 2021. *ACM Journal of Experimental Algorithmics (JEA)*, 27:1–12, 2022.
- 19 Michael R. Fellows, Danny Hermelin, Frances Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009.
- 20 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 (CCCG)*, pages 6–9, 2003.
- 21 Foivos Fioravantes, Dusan Knop, Jan Matyáš Kristan, Nikolaos Melissinos, and Michal Opler. Exact algorithms and lowerbounds for multiagent path finding: Power of treelike topology. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI)*, pages 17380–17388. AAAI Press, 2024.
- 22 Gary William 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.
- 23 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, 2006.

- 24 Fedor V Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. Finding, hitting and packing cycles in subexponential time on unit disk graphs. *Discrete & Computational Geometry*, 62(4):879–911, 2019.
- 25 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.
- 26 Zachary Friggstad and Mohammad R Salavatipour. Minimizing movement in mobile facility location problems. *ACM Transactions on Algorithms (TALG)*, 7(3):1–22, 2011.
- 27 Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified NP-complete problems. In *Proceedings of the 6th annual ACM Symposium on Theory of Computing (STOC)*, pages 47–63, 1974.
- 28 Oded Goldreich. Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard. In *Studies in complexity and cryptography. Miscellanea on the interplay between randomness and computation*, pages 1–5. Springer, 2011.
- 29 Siddharth Gupta, Guy Sa’ar, and Meirav Zehavi. The parameterized complexity of motion planning for snake-like robots. *Journal of Artificial Intelligence Research*, 69:191–229, 2020.
- 30 Danny Hermelin and Xi Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *Proceedings of the 23rd annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 104–113. SIAM, 2012.
- 31 Iyad Kanj and Salman Parsa. On the parameterized complexity of motion planning for rectangular robots. In *Proceedings of the 40th International Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 65:1–65:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.
- 32 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.
- 33 Subhash Khot and Venkatesh Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289(2):997–1008, 2002.
- 34 Nina Klobas, George B. Mertzios, Hendrik Molter, Rolf Niedermeier, and Philipp Zschoche. Interference-free walks in time: temporally disjoint paths. *Autonomous Agents Multi Agent Systems*, 37(1):1, 2023.
- 35 Pascal Kunz, Hendrik Molter, and Meirav Zehavi. In which graph structures can we efficiently find temporally disjoint paths and walks? In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 180–188. ijcai.org, 2023.
- 36 Christos H Papadimitriou, Prabhakar Raghavan, Madhu Sudan, and Hisao Tamaki. Motion planning on a graph. In *Proceedings 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 511–520. IEEE, 1994.
- 37 Alexander Pilz. Planar 3-SAT with a clause/variable cycle. *Discrete Mathematics & Theoretical Computer Science*, 21(3), 2019.
- 38 Daniel Ratner and Manfred Warmuth. The $(n^2 - 1)$ -puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.
- 39 Oren Salzman and Roni Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AMAAS ’20)*, pages 1711–1715, 2020.
- 40 Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

- 41 Roni Stern. Multi-agent path finding - an overview. In *Artificial Intelligence - 5th RAAI Summer School*, pages 96–115, Dolgoprudny, Russia, 2019. Springer.
- 42 Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Barták. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the 12th International Symposium on Combinatorial Search (SOCS)*, pages 151–159, 2019.
- 43 Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI '13)*, pages 1443–1449, 2013.