Relocating Units in Robot Swarms with Uniform Control Signals is PSPACE-Complete*

David Caballero Angel A. Cantu Timothy Gomez Austin Luchsinger Robert Schweller Tim Wylie

Abstract

This paper investigates a restricted version of robot motion planning, in which particles on a board uniformly respond to global signals that cause them to move one unit distance in a particular direction on a 2D grid board with geometric obstacles. We show that the problem of deciding if a particular particle can be relocated to a specified location on the board is PSPACE-complete when only allowing 1×1 particles. This shows a separation between this problem, called the *relocation problem*, and the *occupancy problem* in which we ask whether a particular location can be occupied by any particle on the board, which is known to be in P with only 1×1 particles. We then consider both the occupancy and relocation problems for the case of extremely simple *rectangular* geometry, but slightly more complicated pieces consisting of 1×2 and 2×1 *domino* particles, and show that in both cases the problems are PSPACE-complete.

1 Introduction

The advanced development of microbots and nanobots has quickly become one of the most significant frontiers of our time. However, power and computation limitations at these scales often make autonomous robots infeasible and individually-controlled robots impractical. Thus, recent attention has focused on controlling large numbers of relatively simple robots. Many examples of large population robot swarms exist, ranging from naturally occuring magnetotactic bacteria [9, 11, 12] to manufactured light-driven "nanocars" [7, 13]. These particular microrobot swarms are manipulated uniformly through the use of external inputs such as light or a magnetic field. That is, all of the agents in the system react identically to the same global signal.

First proposed by Becker et al. [5], this model consists of movable polyominoes (as an abstraction of these nanorobots) that exist on a 2D grid board with "open" and "blocked" spaces. These polyominoes may be affected by global signals and step one unit distance when given a move command. Similar work has been shown in [4], where instead of moving one unit distance they travel maximally (referred to as "tilts"), which causes them to move linearly from one open location to another.

Previous Work. Before the tilt model was formally defined, there was research studying uniform control of particle swarms with precise movement [5]. Shortly after, investigation began on a version of particle swarm control where commands became limited and caused particles to move maximally [4]. In this work, the authors ask if any particle within a system can be moved to occupy a specified location. We refer to this problem as the *occupancy* problem. They prove that deciding the minimum number of moves needed to reconfigure one configuration of robots to another is PSPACE-complete. Recently in [2, 3], two additional natural questions for the model were proposed: the *relocation* and *reconfiguration* problems. The first asks whether a specified particle can be moved to a specified location. The second problem is to determine whether or not every particle in the system can be moved to its own specified location. In the later work the authors proved all of these problems to be PSPACE-complete even when limited to 1×1 tiles. These problems have also been investigated in the single-step model when considering limited directions. Recent work in [1, 6] shows that the relocation problem when limited to two or three directions and the reconfiguration problem when limited to two directions are both NP-complete. It was also shown that the *occupancy* problem is solvable in polynomial time in the single-step model even when all four directions are allowed.

Our Contributions. Our contributions are outlined in Table 1. We first show the *relocation problem* is PSPACE-complete with only 1×1 tiles by way of a reduction from a restricted version of the relocation problem within the *full-tilt* model, recently shown to be PSPACE-complete in SODA 2020 [2]. We then consider the case of domino shaped pieces, but with board geometry limited to being a single rectangle, and show that in this case both the *relocation* and *occupancy* problems are PSPACE-complete by a reduction from the problem of traversing a toggle-lock maze, shown to be PSPACE-complete in [8]. Videos of the constructions can be found at `https://asarg.hackresearch.com/main/cccg2020-Complexity`

*This research was supported in part by National Science Foundation Grant CCF-1817602.

Problem	Tile Size	Geometry	Result	Theorem
Occupancy	1×1	All	P	In [6]
Relocation	1×1	Connected	PSPACE-complete	Thm. 2
Occupancy/Relocation	1×1, 1×2	Rectangular	PSPACE-complete	Thms. 4,5

Table 1: An overview of the complexity results. For 1×1 polyominoes, the occupancy problem is in P, but the related problem of relocation is PSPACE-complete. We show that if 1×2 and 2×1 polyominoes (dominoes) are allowed, both of the problems are PSPACE-complete even with rectangular geometry.

2 Preliminaries

Board. A *board* (or *workspace*) is a rectangular region of the 2D square lattice in which specific locations are marked as *blocked*. Formally, an $m \times n$ board is a partition $B = (O, W)$ of $\{(x, y) | x \in \{1, 2, \ldots, m\}, y \in \{1, 2, \ldots, n\}\}$ where O denotes a set of *open* locations, and W denotes a set of *blocked* locations- referred to as "concrete." We classify the different board geometries according to the following hierarchy:

- Connected: A board where the set of open spaces O is a connected shape.
- Simple: A connected board is said to be *simple* if O has genus-0.
- Monotone: A simple board where O is either horizontally monotone or vertically monotone.
- Convex: A monotone board where O is both horizontally and vertically monotone.
- Rectangular: A convex board is *rectangular* if O is a rectangle.

Tile and Polyomino. A tile is a unit square centered on a non-blocked point on a given board. Formally a tile stores a coordinate on the board c and is said to occupy c. A *polyomino* is a finite set of tiles $P = \{t_1, \ldots t_k\}$ that is connected with respect to the coordinates occupied by the tiles in the polyomino. A polyomino that consists of a single tile is informally referred to as a "tile." In this work we only use single tiles and dominos which are polyominos consisting of two tiles.

Configurations. A configuration is an arrangement of polyominoes on a board such that there are no overlaps among polyominoes, or with blocked board spaces. Formally, a configuration $C = (B, P = \{P_1 \ldots P_k\})$ consists of a board B and a set of non-overlapping polyominoes P that each do not overlap with the blocked locations of board B.

Step. A *step* is a way to turn one configuration into another by way of a global signal that moves all tiles in a configuration one unit in a direction $d \in \{N, E, S, W\}$ when possible without causing an overlap with a blocked position, or another tile. Formally, for a configuration $C = (B, P)$, let P' be the maximal subset of P such that translation of all tiles in P' by 1 unit in the direction d induces no overlap with blocked squares or other tiles. A step in direction d is performed by executing the translation of all tiles in P' by 1 unit in that direction.

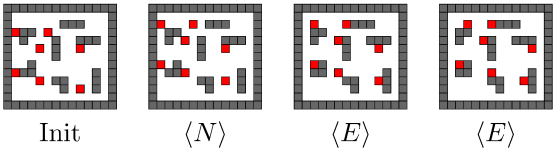

Figure 1: An example step sequence. The initial board configuration followed by the resulting configurations after an N step, E step, and then final E step.

We say that a configuration C can be *directly reconfigured* into configuration C' (denoted $C \rightarrow_1 C'$) if applying one step in some direction $d \in \{N, E, S, W\}$ to C results in C'. We define the relation $\rightarrow_*$ to be the transitive closure of $\rightarrow_1$ and say that C can be *reconfigured* into C' if and only if $C \rightarrow_* C'$, i.e., C may be reconfigured into C' by way of a sequence of step transformations. A related concept that is the focus of previous work is the *tilt* transformation in which a single direction d tilt consists of the repeated application of a direction d-step until the configuration is d-terminal. In this paper we focus on the step transition, but discuss connections to previous work using the tilt transformation.

Step Sequence. A *step sequence* is a series of steps which can be inferred from a series of directions $D = \langle d_1, d_2, \ldots, d_k \rangle$; each $d_i \in D$ implies a step in that direction. For simplicity, when discussing a step sequence, we just refer to the series of directions from which that sequence was derived. Given a starting configuration, a step sequence corresponds to a sequence of configurations based on the step transformation. An example step sequence $\langle N, E, E \rangle$ and the corresponding sequence of configurations can be seen in Fig. 1.

3 Hardness Results for Occupancy and Relocation

In this section we present our two PSPACE-completeness results. We first show the relocation problem is PSPACE-complete when allowing only 1×1 tiles by reducing from a restricted form of reloca-

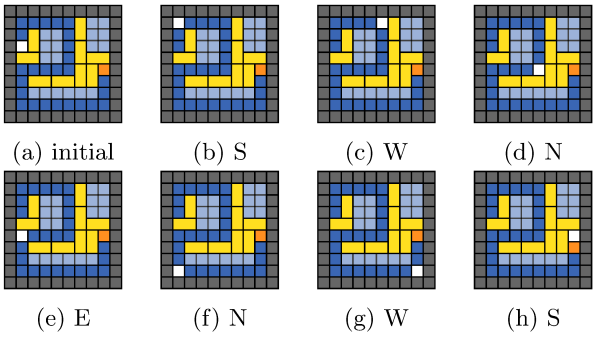

Figure 2: An example of an empty space moving through a configuration. The board geometry is just a rectangular frame. The dominoes along with many of the tiles (shown in lighter blue) are gridlocked and cannot move. We can see that through a sequence of tilts the space can move through the configuration and eventually allow the orange tile to change position.

tion within the full-tilt model, shown to be PSPACE-complete in [2]. Then, we show that both relocation and occupancy problems are PSPACE-complete when allowing 1×2 and 2×1 polyominoes even when restricted to a board with rectangular geometry. We show this by a reduction from the problem of moving a single robot through a *toggle-lock* maze [8]. Both of our PSPACE-hardness reductions utilize a common technique in which we consider an empty space in a mostly-full board as an agent. With this technique, isolated spaces now travel maximally across the board per step, similar to a single tile in the full-tilt model. This method is demonstrated in Figure 2.

3.1 Problem Definitions

Occupancy. The occupancy problem asks whether or not a given location can be occupied by any tile on the board. Formally, given a configuration $C = (B, P)$ and a coordinate $e \in B$, does there exist a step sequence such that $C \rightarrow_* C'$ where $C' = (B, P')$ and $\exists p \in P'$ that contains a tile that occupies coordinate e?

Relocation. The relocation problem asks whether a specified polyomino can be relocated to a particular position. Formally, given a configuration $C = (B, P)$, a polyomino $p \in P'$, and a coordinate $e \in B$, does there exist a step sequence such that $C \rightarrow_* C'$ where $C' = (B, P')$ and a tile in p occupies coordinate e?

3.2 Relocation with 1×1s

Recently, [2] proved that occupancy and relocation in the full-tilt model are PSPACE-complete with only 1×1 tiles. We can reduce directly from a modified version of the occupancy problem in full-tilt. The key idea in the reduction to invert the construction from [2] so that every space is replaced by a single tile, and vice versa. Now, the empty spaces act as the tiles in the original reductions and behave similarly to that in Figure 2 (although, this results in a board with no dominoes).

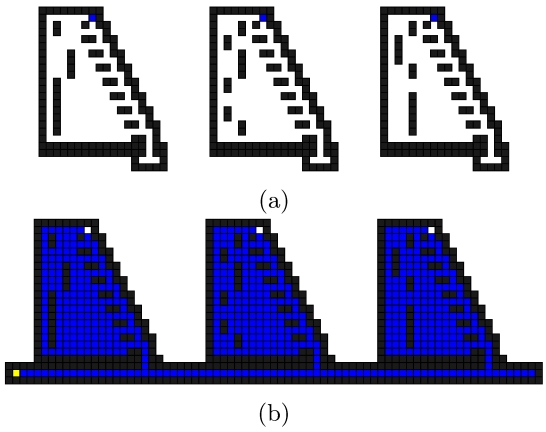

Figure 3: (a) An example input for the k-region relocation problem. (b) Reducing the k-region relocation problem to the relocation problem.

Lemma 1 *The relocation problem in the single-step model is in PSPACE.*

Proof. The problem can be solved by non-determinisitically selecting a movement from the available current movements until a tile is in the correct position. We only need to keep track of the current configuration between each move so the problem can be solved in NPSPACE which is known to equal PSPACE. □

Theorem 2 *The relocation problem in the single-step model is PSPACE-complete even when limited to only* 1×1 *tiles and connected geometry.*

Proof. To show hardness we reduce from a restricted version of the relocation problem under the *full-tilt* operation. The full-tilt model simply moves all pieces maximally in a given direction until colliding with a wall or other obstructed unit. In [2] the following restricted version of this problem, which we will call the *k-region relocation* problem, was shown to be PSPACE-complete [1] by way of a reduction from non-deterministic constraint logic [10]. In this problem we consider an input board configuration consisting of k disjoint regions, each with a single particle within each region. Further, we append a 1×3 enclosed region to the bottom row of each of these regions that includes a single central opening at

[1] This version of the problem was not explicitly formulated within the conference version of this paper, but this subproblem represents the key portion from which the hardness is derived. Key details and a formal proof is provided in Section 4.

the center leading the next higher row. See Figure 3a for an example. Given such an input, the *k-region relocation* problem asks if it is possible to move all k pieces into their corresponding 1×3 enclosed regions.

Given the PSPACE-hardness of *k-region relocation*, we now show the PSPACE-hardness of the relocation problem within our single-step model. The key idea is to apply the technique of filling each of the k disjoint regions with tiles, with the exception of the location of the given region's single particle. In this way, each step transition moves the empty particle in the same manner a full-tilt transition would maximally move a single particle (but in the opposite direction). Next, we connect the 1×3 output regions as shown in Figure 3b. In this way, the k empty spaces are able to reach the bottom-most row of the configuration if and only if the original *k-region relocation* input can relocate it's k pieces to the k output regions. With a final additional step the k spaces combine to create enough space for the target particle (shown in yellow) to move exactly k spaces to a designated relocation point. □

3.3 Complexity with Rectangular Board Geometry and Dominoes

In this section we relax the restriction on tile size and show both the occupancy and relocation problems are both PSPACE-complete even when restricted to rectangular board geometry, and with particles of size at most 2. We show this by reducing from a simple gadget model proposed in [8]. The authors show that the problem of relocating a single agent in a connected system of these gadgets is PSPACE-complete.

Gadget Basics. The gadgets used follow simple rules. They have two states, and contain tunnels that allow traversal through the gadgets. These tunnels exist in different types, such as the lock and toggle. A toggle tunnel can always be traversed in one direction, and on a state change that direction is reversed. The lock tunnel can be traversed in either direction when it is unlocked, and neither when it is locked. On a state change the lock tunnel will either lock or unlock. A gadget can contain multiple tunnels, each affected by the gadget's state changes. For our purpose we will use a crossing toggle lock, as shown in Figure 4a.

Crossing Toggle-Lock Domino Gadget. The Crossing Toggle-Lock Domino Gadget, shown in Figure 4b, enforces the same rules for traversal with two dominoes. When in the unlocked state the horizontal tunnel contains only 1×1 tiles and allows for the space to travel through it unblocked. When in the locked state there is a domino blocking the horizontal path. When a space attempts to pass through that path it is blocked by the domino and cannot continue through the gadget.

The vertical tunnel only allows traversal in one direction based on the state of the gadget. When in the unlocked state, traversal is allowed from south to north and if attempting to enter from the north, it is blocked by a domino. When in the locked state traversal is only allowed from north to south. Any complete traversal through the vertical tunnel will change the location of the dominoes in the tunnel and the state of the gadget.

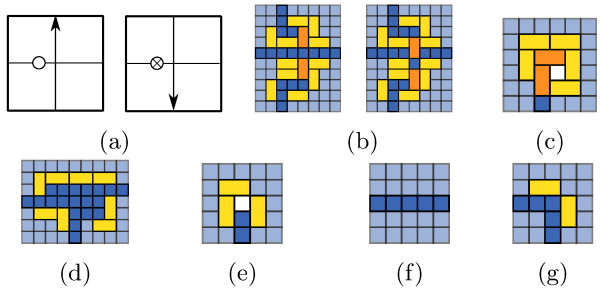

Figure 4: (a) Crossing Toggle-Lock (CTL) gadget in state 1 *left* and in state 2 *right*. The ⊗ represents the locked state of the lock tunnel. (b) The crossing toggle-lock gadget implemented in the single-step tilt model. The left image is in an open position, and the right is the closed position. (c) The goal gadget for occupation. The space can only be covered by a polyomino if another space is in the gadget. (The light blue tiles are used to fill up the board and keep polyominoes in the gadgets from moving. These tiles will never move) (d) The 3-way branching gadget that allows the space to enter at any of the 3 locations and exit at any other. (e) The Start Gadget. Intially contains the space that acts as the agent and has dominos to enforce that the space can only exit at one location. (f) The wire gadget, a group of tiles act as a medium for the space to travel through. (g) Corner Gadget used to allow the space to change directions.

Other Gadgets. In order to fully implement a CTL puzzle, we need a few other gadgets shown in Figures 4(d-g).

Branching Gadget. The other gadget required in the motion planning problem is a 3-way branching gadget. The gadget is shown in Figure 4d and connects all three locations and allows for movement between them. The way the gadget is set up is when entering from any point the space will be able to cycle around the edges of the gadget. At certain positions in the gadget the space will be able to exit out one of the locations.

Wire Gadget. The wire gadget shown in Figure 4f is just a group of single tiles. These tiles connect the other gadgets and allow the agent to travel through them.

Corner Gadget. The puzzle solvabilty problem allows for wires that turn. Since the agent travels the maximum distance possible before reaching a domino or the edge of the board we create a corner gadget (Figure 4g)

to allow the agent to stop and change direction.

Start Gadget. The start gadget (Figure 4e) is where the agent starts. When constructing the reduction the gadget contains the space that acts as the agent. The open position is surrounded on three sides by dominoes so the space can only exit from one side.

Goal Gadget. The goal gadget (Figure 4c) is the objective for the agent to reach. The gadget contains a second empty space that is surronded by dominoes. This space is the goal location. There is a horizontal domino that can be moved into this space if the agent reaches the goal gadget. The horizontal domino can only fill the goal location if the agent reaches the goal location.

Lemma 3 *The occupancy problem in single-step is PSPACE-hard with a rectangular board.*

Proof. Given an instance of a CTL puzzle we create a configuration by replacing each element of the CTL puzzle with one of our gadgets. We replace each CTL gadget with a crossing toggle-lock domino gadget and every 3-way intersection with a branching gadget. We also replace the start location with the start gadget and the goal location with the goal gadget. We finally connect these with wire gadgets and corner gadgets.

Our crossing toggle-lock domino gadget must behave the same as the CTL gadget. We can see that the space can only traverse the crossing toggle-lock domino gadget when an agent can traverse a CTL gadget in the same state. Observe that a space can travel through the horizontal tunnel when the gadget is in the unlocked state. While in this state observe that the space can only traverse the vertical tunnel from south to north since the north entrance is blocked by a domino in all directions. When traversing from south to north in this state we can see that the dominoes are able to move downward one step changing the state of the gadget to the locked state. Observe that in the locked state the horizontal tunnel is blocked by a vertical domino so a horizontal traversal in either direction is not possible. Also, observe that the space cannot traverse the vertical tunnel when entering from the south since it is blocked by a domino. When entering from the north in this state the space can traverse and changes the locations of the dominoes.

There exists a solution to the given instance of the CTL puzzle if and if only if there exists a solution to the occupancy problem on the given configuration. Since the crossing toggle-lock domino gadget has the same behavior of the CTL gadget, and the branching gadget allows a tile to enter and exit at any location, we can see that if the CTL puzzle is solvable then there exists a move sequence that solves the occupancy problem. Also since our gadgets behave the same as the gadgets in the CTL puzzle if there does not exist a solution to the CTL puzzle then there is no way for the space to reach the goal gadget and no way to solve the occupancy problem. □

Theorem 4 *The occupancy problem in single-step is PSPACE-complete when limited to rectangular board geometry if both* 1×1 *tiles and* 1×2 */* 2×1 *dominoes are included.*

Proof. We can see that the occupancy problem is in PSPACE in the same way as in Lemma 1 since we can non-deterministically select a valid move sequence. Through the reduction in Lemma 3 we show the problem is PSPACE-hard so the occupancy problem with the paramaters shown is PSPACE-complete. □

Corollary 5 *The relocation problem in single-step is PSPACE-complete even when limited to a rectangular board geometry when allowing* 1×1 *tiles and* 1×2 */* 2×1 *dominoes.*

Proof. We can see from Lemma 1 that the relocation problem is in PSPACE. The reduction from above can be extended to show the relocation problem is PSPACE-hard by asking if the horizontal domino in the goal gadget can reach the positon directly below it. □

4 Relocation Complexity in Full Tilt

This section is taken from [2] with the additional proof of *k-region relocation* hardness. To achieve this result we provide a polynomial time reduction from Non-Deterministic Constraint Logic [10]. We explain high level details of this construction along with key lemmas.

4.1 Non-Deterministic Constraint Logic

A constrant logic graph is a weighted directed graph with a constraint on each of the vertices [10]. The constraint specifies the minimum weight required from the edges directed in (the sum of the inflow) to any vertex. When given a graph, the usual problem studied is whether a particular edge can be "flipped"- the direction of the edge changed, i.e., is there a sequence of edge flips that maintain the constraints on all vertices, and allows the target edge to be flipped? This is a one-player unbounded game. The problem is still PSPACE-Complete when the edge weights are all strength 1 or 2, and vertices have max degree 3. We address the following equivalent problem.

Configuration-to-Configuration Problem. Given two states of a constraint graph G and G', does there exist a sequence of edge flips starting with G that results in G' [10].

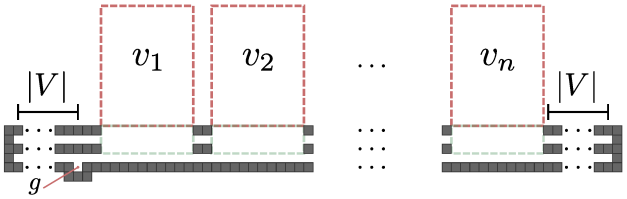

Figure 5: An overview of the layout of the different components for the reduction. The dotted red lines represent where each of the vertex gadgets go (not to scale), the dotted green boxes below denote the geometry specific to each vertex to force the state tile into the top row (the vertex was in the wrong state) unless the vertex is in the state specified by the target configuration. The bottom row requires all $|V|$ state tiles in order for a tile to get into the goal location g.

4.1.1 Vertex Gadget

Assuming a max degree of three, there are 8 possible arrangements of in/out edges. Define the vertex *state* as a label from 0 to 7 determined by the directions of its incident edges. A vertex gadget contains a single 1×1 tile refered to as the *state tile*, a *transition area*, and a number of *state gadgets* equal to the number of legal states of that vertex. Since there are eight states, there are eight basic paths in the gadget that the state tile could be in representing the vertex's state.

Flipping an edge is represented by a move sequence performed while in a valid state that moves the state tile from one state path to another, which happens simultaneously in two vertex gadgets since an edge connects two vertices. This edge flip happens in all vertex gadgets, but if the edge is not incident to that vertex, there is no effect on the path of the state tile.

4.1.2 Goal Area

An overview of the reduction layout is in Figure 5 where the goal area is shown at the bottom of all the vertex gadgets. Once all the tiles are in positions that represent the target configuration, the tiles can be extracted into the goal area though the bottom of a state gadget. After extraction the tiles enter the goal area. The goal area consists of two rows. The valid row and the invalid row. The invalid row (top row) traps any tiles that enter when a vertex was not in the specified (in the target configuration) state. If there exists a solution to the Configuration-to-Configuration Problem then all tiles will be able to reach the valid row.

Lemma 6 *After performing a move sequence to flip an edge, only the two vertex gadgets representing vertices incident to that edge will have their state tile change state paths. All other vertex gadgets will have their state tile stay in the same state path.*

Lemma 7 *If a vertex enters an illegal state, the representative vertex gadget's state tile will be trapped in an 'illegal' state path and cannot be extracted.*

4.2 Hardness of k-region Relocation

In this section we will describe how to modify the reduction from [2] to show hardness for the *k-region relocation problem.*

k-region relocation. The k-region relocation problem asks: given a board with k disjoint regions each containing a single tile, and a set of positions in each region called goal areas, does there exist a move sequence that relocates all tiles to their goal area?

Theorem 8 *The k-region relocation problem in the full-tilt model is PSPACE-hard.*

Proof. First, note in the orginal reduction that the goal location may be filled if and only if each tile is extracted from its vertex gadget and enters the goal row. This means that the problem of "Can each tile be extracted from it's vertex gadget?" is PSPACE-hard. Now consider the board used for the proof of hardness for the occupancy problem in [2]. Each vertex gadget is only connected to the others through the two rows at the bottom of the construction. Both of these rows can be removed and replaced with the 1×3 regions described in Theorem 2. The k-many 1×3 rows (which replaced the goal row) can now be reached if and only if each tile can be extracted from it's goal gadget. □

5 Future Work

There are a number of directions for future work. We show that with only 1×1 tiles the *relocation* problem is PSPACE-complete with a connected board. Relocation and occupancy become PSPACE-complete when restricted to a rectangular board but allowing for larger pieces. How much power do these constraints remove? Do these problems become easier when only restricting either the board geometry or the number of larger pieces (i.e., constant number of dominoes), or are they still hard?

References

[1] Jose Balanza-Martinez, David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert Schweller, and Tim Wylie, *Relocation with uniform external control in limited directions*, The 22^{nd} Japan Conference on Discrete and Computational Geometry, Graphs, and Games, JCDCGGG, 2019, pp. 39–40.

[2] Jose Balanza-Martinez, Timothy Gomez, David Caballero, Austin Luchsinger, Angel A. Cantu, Rene Reyes, Mauricio Flores, Robert T. Schweller, and Tim Wylie, *Hierarchical shape construction and complexity for slidable polyominoes under uniform external forces*, Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA'20, SIAM, 2020, pp. 2625–2641.

[3] Jose Balanza-Martinez, Austin Luchsinger, David Caballero, Rene Reyes, Angel A. Cantu, Robert Schweller, Luis Angel Garcia, and Tim Wylie, *Full tilt: Universal constructors for general shapes with uniform external forces*, Proceedings of the 30^{th} Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'19, 2019, pp. 2689–2708.

[4] Aaron T. Becker, Erik D. Demaine, Sándor P. Fekete, Golnaz Habibi, and James McLurkin, *Reconfiguring massive particle swarms with limited, global control*, Algorithms for Sensor Systems (Berlin, Heidelberg) (Paola Flocchini, Jie Gao, Evangelos Kranakis, and Friedhelm Meyer auf der Heide, eds.), Springer Berlin Heidelberg, 2014, pp. 51–66.

[5] Aaron T. Becker, Golnaz Habibi, Justin Werfel, Michael Rubenstein, and James McLurkin, *Massive uniform manipulation: Controlling large populations of simple robots with a common input signal*, 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nov 2013, pp. 520–527.

[6] David Caballero, Angel A. Cantu, Timothy Gomez, Austin Luchsinger, Robert Schweller, and Tim Wylie, *Hardness of reconfiguring robot swarms with uniform external control in limited directions*, ArXiv e-prints (2020), arxiv:2003.13097.

[7] Pinn-Tsong Chiang, Johannes Mielke, Jazmin Godoy, Jason M. Guerrero, Lawrence B. Alemany, Carlos J. Villagómez, Alex Saywell, Leonhard Grill, and James M. Tour, *Toward a light-driven motorized nanocar: Synthesis and initial imaging of single molecules*, ACS Nano **6** (2012), no. 1, 592–597, PMID: 22129498.

[8] Erik D. Demaine, Isaac Grosof, Jayson Lynch, and Mikhail Rudoy, *Computational complexity of motion planning of a robot through simple gadgets*, 9th International Conference on Fun with Algorithms, FUN 2018, June 13-15, 2018, La Maddalena, Italy, 2018, pp. 18:1–18:21.

[9] Ouajdi Felfoul, Mahmood Mohammadi, Louis Gaboury, and Sylvain Martel, *Tumor targeting by computer controlled guidance of magnetotactic bacteria acting like autonomous microrobots*, 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sep. 2011, pp. 1304–1308.

[10] Robert A. Hearn and Erik D. Demaine, *The nondeterministic constraint logic model of computation: Reductions and applications*, Proceedings of the 29th International Colloquium on Automata, Languages and Programming (London, UK, UK), ICALP '02, Springer-Verlag, 2002, pp. 401–413.

[11] Sylvain Martel, *Bacterial microsystems and microrobots*, Biomedical Microdevices, vol. 14, 2012, pp. 1033–1045.

[12] Sylvain Martel, Samira Taherkhani, Maryam Tabrizian, Mahmood Mohammadi, Dominic de Lanauze, and Ouajdi Felfoul, *Computer 3d controlled bacterial transports and aggregations of microbial adhered nano-components*, Journal of Micro-Bio Robotics **9** (2014), no. 1, 23–28.

[13] Yasuhiro Shirai, Andrew J. Osgood, Yuming Zhao, Kevin F. Kelly, and James M. Tour, *Directional control in thermally driven single-molecule nanocars*, Nano Letters **5** (2005), no. 11, 2330–2334, PMID: 16277478.