8-2020

Building Patterned Shapes in Robot Swarms with Uniform Control Signals

David Caballero
The University of Texas Rio Grande Valley

Angel A. Cantu

Timothy Gomez

Austin Luchsinger
The University of Texas Rio Grande Valley

Robert Schweller
The University of Texas Rio Grande Valley

See next page for additional authors

Building Patterned Shapes in Robot Swarms with Uniform Control Signals*

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. We look at the problem of assembling patterns within this model. We first derive upper and lower bounds on the worst-case number of steps needed to reconfigure a general purpose board into a target pattern. We then show that the construction of k-colored patterns of size-n requires $\Omega(n \log k)$ steps in general, and $\Omega(n \log k + \sqrt{k})$ steps if the constructed shape must always be placed in a designated output location. We then design algorithms to approach these lower bounds: We show how to construct k-colored $1 \times n$ lines in $O(n \log k + k)$ steps with unique output locations. For general colored shapes within a $w \times h$ bounding box, we achieve $O(wh \log k + hk)$ steps.

1 Introduction

In this paper we investigate a model of robot motion planning first proposed by Becker et. al. [7] in which n robots exist on a 2D grid consisting of "open" and "blocked" spaces, and are controlled by way of uniform control signals which tell all robots to move one step in any one of the four cardinal directions. This model, which we call the single-step model, has important applications for the scalable development of microbot and nanobot swarms due to the simplified method of control [9,11]. While previous work in this model has investigated how to build general shapes [7], and the hardness of relocation related problems [1], here we focus on the problem of quickly rearranging the robots into a desired colored pattern (with an arbitrary shape).

In particular, our problem is as follows. Given a color palette of k distinct colors, as well as a bounding box of width w and height h, our goal is to design a *universal* board configuration (a board with open and blocked locations, as well as specified locations for a set of robots each assigned one of the k colors) with the property that any pattern fitting within a $w \times h$ bounding box can be assembled (the robots can be reconfigured into the provided pattern) in a near-optimal number of steps.

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

Our results. We first focus on a special case class of patterns consisting of $1 \times n$ lines over k colors. We provide a board that can assemble any $1 \times n$ patterned line over k colors within $O(n \log k + k)$ steps, along with showing a lower-bound of $\Omega(n \log k + \sqrt{k})$ under the assumption that the board must always place the output pattern in the same location. We extend this to general 2D shapes of size n and provide a construction achieving $O(wh \log k + hk)$ steps, which for dense shapes of size n is comparable to the lower bound of $\Omega(n \log k + \sqrt{k})$.

Previous Work. The single-step model of this paper was first studied in [7] where it was shown how to reconfigure n robots into any size n shape within $O(n^2)$ steps given a single blocked location. An additional line of related research considers global movement signals, but requires that all pieces move maximally in the input direction [4]. This line of research has explored building shapes [2,3,6,8,10], performing computation [5], as well complexities for reconfiguration and relocation of particles [2–5]. Additionally, [12] considers the reconfiguration of rectangular patterns of n colored robots within $O(n^2)$ steps. While closely related to our work, this work differs from the problem we are considering in that 1) they consider the maximal-movement of particles, and 2) we are attempting to build arbitrary patterns, while they are rearranging a given set of pieces (meaning the number of each color in the pattern is fixed). We also consider general shaped patterns and striving for near-optimal construction times, and are not attempting to reconfigure all pieces on the board.

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."

Tiles/Robots. A tile/robot is a labeled unit square centered on a non-blocked point on a given board. Formally, a tile is an ordered pair (c, a) where c is a coordinate on the board, and a is a label.

Configurations. A configuration is an arrangement of tiles on a board such that no tiles occupy the same

Result	Step Complexity		Theorem
	Lower	Upper	
Patterned Lines	$\Omega(n \log k + \sqrt{k})$	$\mathcal{O}(n \log k + k)$	Thms. 3, 4
General Patterns	$\Omega(n \log k + \sqrt{k})$	$\mathcal{O}(wh \log k + hk)$	Thms. 3, 5

Table 1: Construction Results. The patterned lines result is for $1 \times n$ lines using k colors. The general patterns result is for k-colored size-n $w \times h$-bounded shapes.

location, or occupy blocked board spaces. Formally, a configuration $C = (B, P)$ consists of a board B and a set of tiles P whose coordinates do not overlap each other, or with 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.

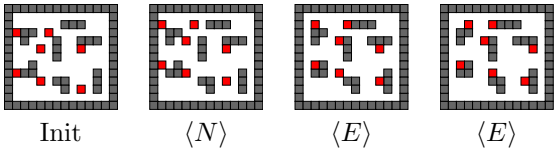
We say that a configuration C can be *directly reconfigured* into configuration C' (denoted $C \rightarrow_1 C'$) if applying one step in some direction $d \in \{N, E, S, W\}$ to C results in C'. We define the relation $\rightarrow_*$ to be the transitive closure of $\rightarrow_1$ and say that C can 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 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 Figure 1a.

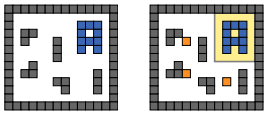
Universal Configuration. A configuration C' is universal to a set of configurations $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ if and only if $C' \rightarrow_* C_i, \ \forall \ C_i \in \mathcal{C}$.

Shape/Pattern. We define a shape to be a connected subset $S \subset \mathbb{Z}^2$. We define a pattern to be a tuple (S, L), where S is a shape and $L : S \rightarrow A$ is a total function that maps each point to a label in a set of labels A.

Configuration Representation. A configuration may be interpreted as having constructed a "shape" in a natural way. A configuration C *strongly* represents shape S if the collection of all tile coordinates in C is exactly the set of points of some translation $t(S)$.



(a) An example step sequence



(b) Representation Examples

Figure 1: (a) An example step sequence. The initial board configuration followed by the resulting configurations after an N step, E step, and then final E step. (b) Configuration Representation Examples. Both of these configurations are different representations of the shape "A." First, we show a strong representation where every tile in the configuration contributes to the shape. Then we show a regional representation. The yellow square represents the output region, and the orange tiles represent additional polyominoes on the board which do not count towards shape representation (as they are not in the output region).

An alternate form of representation allows for a rectangular region of the board to be deemed the *output* region. Here, we say a configuration *regionally* represents a shape S *w.r.t.* output region T if the collection of all tile coordinates in T is exactly the set of points of some translation $t(S)$. Figure 1b illustrates the different types of representations. In the regional representation, any tiles outside of the output region are ignored.

We extend this idea of shape representation to include patterns. A configuration C represents a pattern (S, L) if C represents S and there exists a translation t, such that for all tiles $(c, a), L(t(c)) = a$. The idea of regional representation of a pattern extends in the same way.

Universal Pattern Builder. Given the concept of pattern representation, a configuration C' is *universal* for a set of patterns $\mathcal{S}$ if and only if there exists a set of configurations $\mathcal{C}$ such that 1) each $S \in \mathcal{S}$ is represented

by a unique $C \in \mathcal{C}$ and 2) C' is universal for $\mathcal{C}$.

We say that C' is *regionally universal* for $\mathcal{S}$ *w.r.t.* output region T if $\forall S \in \mathcal{S}\ \exists\ C \in \mathcal{C}$ s.t. C regionally represents S *w.r.t.* T. Further, C' achieves *unique placement* if output region T is exactly the size of the minimum bounding box which can contain any of the patterns in $\mathcal{S}$. In this paper we focus on designing regionally universal configurations for general patterns fitting within a $w \times h$ bounding box.

Worst-Case Step Complexity for Universal Configurations. Given a universal configuration C, the worst-case step complexity is the maximum number of steps required to reconfigure C into some element from its universe set. Consider a universal configuration C over a set of configurations U. For each $u \in U$, let $d(C, u)$ denote the length of the smallest step-sequence from C to u. The worst-case step complexity of C over U is defined to be $\max\{d(C, u) | u \in U\}$.

3 Fast Universal Constructors: Patterns

We now focus on building shapes with a desired color pattern. To model this, we specify each robot in the system to have a designated color from a given set of k colors. Our goal is then to design configurations that allow quick reconfiguration of the robots into a specified shape with a specified color pattern. We start with an analysis of some lower bounds for any k-color pattern constructors in Section 3.1. We then derive upper bounds for linear patterns in Section 3.2, general patterns in Section 3.3. Accompanying videos for these constructions can be found at `https://asarg.hackresearch.com/main/CCCG2020-Patterns`.

3.1 Lower Bounds on Patterns

Lemma 1 *For a given set of n distinct points from the 2D integer lattice, consider the corresponding set of all size-n colored patterns over those points using at most k distinct colors. Any universal configuration for such a set of patterns has worst-case step complexity $\Omega(n \log k)$.*

Proof. There are k^n distinct k-color patterns over n points. Therefore, any universal configuration for this set of patterns must be universal to a set of k^n configurations. The maximum number of distinct configurations reachable from an initial configuration C' within r steps is upper bounded by

$$\sum_{i=0}^{r}(4^i) = \frac{4^{r+1}-1}{3}.$$

Thus C' must satisfy that $\frac{4^{r+1}-1}{3} \geq k^n$, implying that $r = \Omega(n \log k)$. □

Lemma 2 *Any universal configuration for all k-colored patterns over a size-n shape with unique placement has worst-case step complexity $\Omega(\sqrt{k})$.*

Proof. Consider a unique placement universal constructor for all k-colored patterns over some size-n shape. As this is a unique placement constructor, the output zone is a fixed region of size exactly the bounding box of the size-n shape. Select an arbitrary point $p = (x, y)$ within the output region that is covered by the size-n shape when inscribed within the output region. Let $d = \lfloor \frac{\sqrt{k}}{4} - 1 \rfloor$ and note that the number of points within (Manhattan) distance d of (x, y) is strictly less than k. Therefore, there must be one color c for which all tiles of color c are at least distance d from point (x, y). Further, as this system is universal for all k-colored patterns over the target shape, and the unique placement restriction enforces the output shape into a fixed position for each represented pattern, there exists a pattern in the universe for which the color c must be placed at position (x, y). The step-sequence to place a color c tile at location (x, y) requires at least $d = \Omega(\sqrt{k})$ steps, and therefore requires at least $\Omega(\sqrt{k})$ steps to finish this pattern. □

Theorem 3 *Any universal configuration for all k-colored patterns over a shape of size-n has worst-case run-time at least $\Omega(n \log k)$. If the configuration satisfies the unique placement requirement, the worst-case run-time is at least $\Omega(n \log k + \sqrt{k})$.*

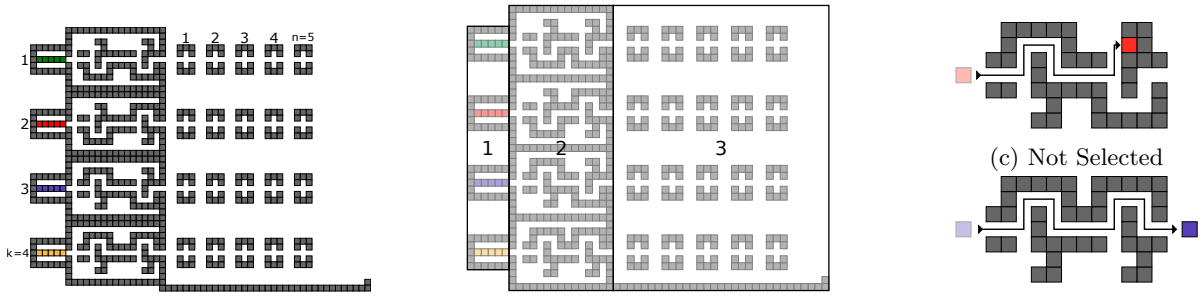
Proof. This follows from Lemma 1 and Lemma 2. □

3.2 Fast Linear Patterns

For our first positive result on universal pattern building we focus on the case of linear $1 \times n$ shapes over k colors. We construct a universal configuration with worst-case run time of $\mathcal{O}(n \log k + k)$ (Theorem 4), which is reasonably close to the lower bound of $\Omega(n \log k + \sqrt{k})$ shown in Theorem 3, and optimal in the case where $n \geq k$. The linear pattern constructor is made up of three sections: *fuel chambers*, *bit selectors*, and *holding chambers*.

Fuel Chambers. This section of the constructor consists of the fuel chambers, where each are $3 \times n$ open spaces surrounded by concrete with an opening on the center right. Morever, each chamber contains a $1 \times n$ line of robots of one color. Using the opening on the right side of each chamber we can "chop" off one robot at a time. By chopping off a robot from each chamber in parallel, we transmit a column of k differently colored robots into section 2.

Bit Selectors. The *bit selectors* are gadgets used to assign a unique *bit-string* to each colored robot entering the section. These bit-strings are created by the unique combination of two smaller gadgets called the



(a) Constructor (b) Sections (c) Not Selected (d) Selected

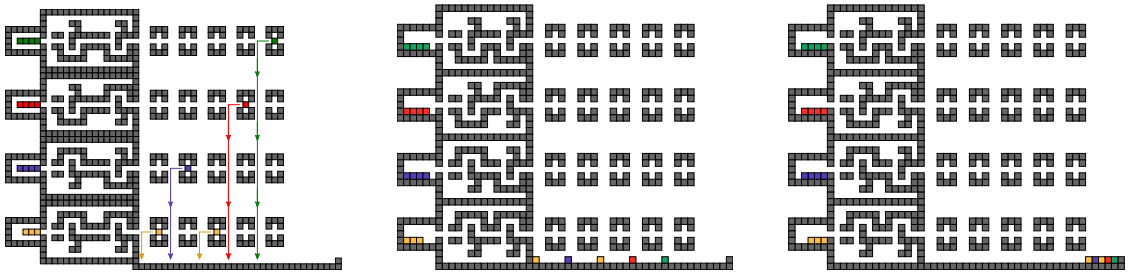
Figure 2: (a) The linear patterns constructor and (b) the different sections. Section 1 consists of the fuel chambers. Section 2 consists of bit-selector gadgets. Section 3 consists of tile holding chambers, as well as a concrete floor where the line will be assembled. (c) Example bit selection where one robot is extracted via execution of its unique bit-string.

up-select and *down-select*. Each of these smaller gadgets has an open space path from one side to the other such that each of their paths are the opposite of the other smaller gadget. The incorrect path causes a robot to get stuck. This idea is demonstrated in Figure 2d, where one robot can sucessfully traverse the bit selectors at the cost of the other robot stopping in the up-select gadget. Therefore, for all k robots entering this section from the fuel chamber, we can design a unique combination of up-select and down-select gadgets such that traversal of any robot through their respected gadgets will yield that robot on the other side, while all others stay within their gadgets. Therefore, bit selectors consisting of $\log k$ bits each are needed to yield a unique bit-string per robot, each of which creates an open space path from one side to the other of length $\mathcal{O}(\log k)$.

Holding Chambers. Each individual robot enters section 3 through the left side at possibly different heights since each colored robot comes from a different bit selector. There are nk holding chambers in the output area of the bit selectors, where each holding chamber is a 1×3 open space surrounded by concrete tiles, with an open space path from the center left to right. These chambers hold the robots in place while another robot is being extracted from the fuel chambers. After outputting a robot from a bit selector gadget, we place the robot in the closest holding chamber to the right. After placing the new robot in a holding chamber, we address the unselected robots that were blocked in the bit selectors. The unused robots are placed back into the fuel chambers by the sequence $\langle W^4, N^2, W^{\mathcal{O}(\log k)}, S^8, W^{\mathcal{O}(\log k)}, N^4, W^2, N, E^{\mathcal{O}(\log k)}\rangle$. After returning the unselected robots to their fuel chambers, we continue the building process. The sequence to extract robots and traverse the robot through the bit selector gadgets also moves all robots in a holding chamber to the next holding chamber on their right. After the n^{th} robot has been placed in section 3, we combine them by extracting them from the holding chambers and placing them all on the concrete floor. Then, we push them together using the single concrete tile on the right of this floor. Figure 3 shows an example of this sequence.

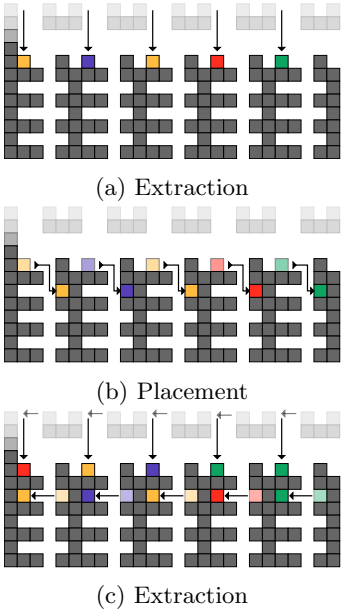
Theorem 4 *For any positive integers n and k, there exists a regionally universal configuration for all $1 \times n$ k-colored lines with worst-case step complexity $\mathcal{O}(n \log k + k)$. Moreover, this configuration obtains unique placement and has board-size $\mathcal{O}(n + \log k) \times \mathcal{O}(k)$.*

Proof. Above we describe a configuration $C = (B, P)$ such that it consists of three sections. The first section pertains to the fuel chambers, which is used to hold k $1 \times n$ lines of robots, one line for each color, in separate chambers. It follows that a single robot can be extracted from each of these chambers, resulting in a column of k robots entering the third section. The third section consists of the bit-string gadgets, where each receives one of the k robots. We have shown that the bit-string gadgets each have a specific unique sequence that takes the robot from the left side to the right side such that performing the sequence of one bit-string gadget will make all robots, save for the one that is within that bit-string gadget, stuck in one of the compartments in the bit selectors of the other bit-string gadgets. Therefore, it is possible to send one robot to the third section in $\mathcal{O}(\log k)$ steps. The holding chambers in the third section are used to hold the robots in place while the next robots are being extracted from the former two sections. Together, these sections can place n robots in the holding chambers in the third section, after which we can remove these robots from the holding chambers and combine them to form a line at the bottom side of the third section. Placing the robots at the bottom of the third section takes $\mathcal{O}(k)$ steps, while combining them takes $\mathcal{O}(n)$ steps. Therefore, the configuration C = (B, P) is a universal configuration for all $1 \times n$ k-colored lines



(a) Holding Chambers (b) Extraction (c) Line Building

Figure 3: Line building depicted.



(a) Extraction

(b) Placement

(c) Extraction

Figure 4: Line holders depicted. After each line is built and extracted from the holding chambers, we place them in the first row of the line holder. This, in parallel, will move each line already within the line holder down into the next row.

with worst-case runtime $\mathcal{O}(n \log k + k)$. Moreover, this configuration has board size $\mathcal{O}(n + \log k) \times \mathcal{O}(k)$ and achieves unique placement w.r.t. a $1 \times n$ rectangular output region located at the bottom-right of the board, along the concrete floor. □

3.3 General Patterns

We now generalize our line pattern construction to general shapes over k colors. For given positive integers n, h and w we focus on size-n shapes fitting in a $h \times w$ bounding box.

Line Holders. For general shapes, we replace the south concrete floor of section 3 of the line pattern builder with the *line holder* depicted in Figure 4. As shown, each new line built can be moved into the line holder. If some lines are inside the line holder already, those lines will move in parallel to the next chamber below whenever a new line is added to the line holder. After all lines have been built, we extract them, yielding essentially a general $w \times h$ pattern, but with a constant vertical and horizontal gap between tiles. Further, by adding in an "empty" color chamber, we can include empty spaces within this pattern, yielding a general patterned shape. Finally, to remove the gaps in the shape, we apply a *funneling* operation, described in Section 4.

Theorem 5 *For positive integers w and h, each greater than some constant, and positive integer k, there exists a regionally universal configuration for any k-colored size-n shape fitting within a $h \times w$ bounding box with worst-case step complexity $\mathcal{O}(wh \log k + hk)$ and board size $\mathcal{O}(wh + \log k) \times \mathcal{O}(\max(h, k))$.*

Proof. The k-colored shape constructor is a simple extension from the $1 \times n$ constructor. The main addition of the line constructor is the line holders at the bottom of the third section. Each different line we construct can be held inside one of these different line holders in order to build another line. After each line is made, we can move that line into the line holders and at the same time move any line already in the line holders down one row. After each line is built, we can extract them and send them through the funneling gadget in order to remove the *constant* amount of space between each tile. With the inclusion of "empty" tiles, we obtain general patterned shapes. The details of the funneling gadget are presented in 4. Therefore, the configuration $C = (B, P)$ is a universal configuration for all k-colored size-n shapes fitting within a $h \times w$ bounding box with worst-case step complexity $\mathcal{O}(wh \log k + hk)$. Moreover, this configuration has board size $\mathcal{O}(n + \log k) \times \mathcal{O}(k)$ and achieves unique placement w.r.t. a $w \times h$ output region located just above the funneling gadget. □

4 Funneling Gadget

The *funneling* gadget is designed to take a group of robots separated by a constant amount of spaces and co-

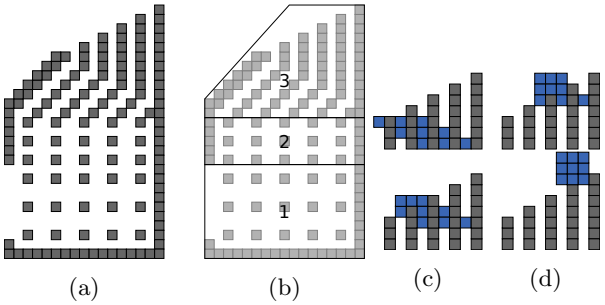


Figure 5: (a) An example funneling gadget for a 3×3 shape. (b) Basic functional sections of the funneling gadget. (c-d) Repeating the sequence $\langle N, E \rangle$ will yield the shape on the outside of the funneling gadget.

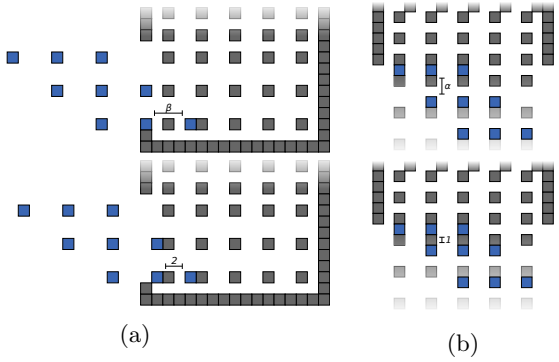


Figure 6: (a-b) Reducing the horizontal distance between the rightmost column of robots. (c-d) Reducing the vertical distance between the topmost row of robots.

alesce them into a desired shape. The architecture and sections of the funneling gadget are illustrated in Figure 5. Let α and β be constants equaling the largest number of vertical and horizontal spaces, respectively, that separates a robot from its neighbor in the shape. The first section of the funneling gadget reduces β so that the largest horizontal separation between two robots is two. Section two takes the group of robots from the former section and reduces α until it is one. The third section finally reduces α and β to zero, and outputs the group of robots as the desired shape outside the funneling gadget. However, this process skews the shape in one direction. This effect can be countered if the input group of robots are instead skewed in the *opposite* direction before passing them through the funneling gadget.

Section One. Section one consists of a grid-like organization of concrete tiles that are themselves spaced out vertically by α but horizontally by two, as shown in Figure 6. To reduce β to two, we place the rightmost column of the group of robots between the two leftmost columns of concrete tiles (Figure 6a). By stepping in the $\langle E \rangle$ direction enough times, the second-to-rightmost column of the group of robots will meet the leftmost column of section one, reducing the spaces between the two columns of robots. After repeating this process for every column of robots, section one will contain within itself the group of robots vertically separated by α and horizontally separated by a distance of two.

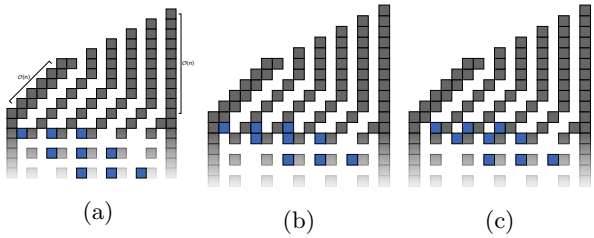


Figure 7: (a-b) Making the rows of robots adjacent by using section three.

Section Two. Section two is a grid-like configuration of concrete tiles that are vertically separated by one space and horizontally separated by two spaces. The same basic process is applied here, but we instead place the rows of robots in between the rows of concrete tiles and perform sufficient steps in the $\langle N \rangle$ direction, as shown in Figure 7.

Section Three. By positioning the group of robots in the third section as depicted in Figure 7a, stepping twice in the $\langle N \rangle$ direction will cause the topmost rows of the group of tiles to meet. This is repeated for every row by first stepping in the $\langle E \rangle$ direction, followed by two steps in the $\langle N \rangle$ direction. After every row has been made adjacent, repeating the step sequence $\langle N, E \rangle$ will output the robots from the funneling gadget, bringing together each column of robots and outputting the desired shape at the top of the gadget, (Figures 5c, 5d).

5 Future Work

Our work leads into a number of areas for future work. The first direction is to attempt to close the gaps between our upper bounds and our lower bounds for linear and general patterns. For lines, the goal is to close the $\Theta(\sqrt{k})$ gap between our upper and lower bounds, and with general shapes, we are interested in closing the gap for sparse shapes existing in large bounding boxes. Another direction is to consider how the unique placement requirement affects the required run-time. Without it, the $\Omega(\sqrt{k})$ lower bound no longer holds. Is it possible to achieve $O(n \log k)$ step complexity by placing different patterns at different locations? And if so, can this be done with a polynomial sized board? Finally, another interesting direction is to focus on pattern reconfiguration, similar to what [12] have looked at within the full tilt model. How fast can reconfiguration be done in the single-step model? Can reconfiguration be done quickly for general patterns and general shapes?

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, Erik D. Demaine, Sándor P. Fekete, Jarrett Lonsford, and Rose Morris-Wright, *Particle computation: complexity, algorithms, and logic*, Natural Computing **18** (2019), 6751–6756.

[6] Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, Christian Rieck, Christian Scheffer, and Arne Schmidt, *Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces*, 2017.

[7] 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.

[8] Sheryl Manzoor, Samuel Sheckman, Jarrett Lonsford, Hoyeon Kim, Min Jun Kim, and Aaron T. Becker, *Parallel self-assembly of polyominoes under uniform control inputs*, IEEE Robotics and Automation Letters **2** (2017), no. 4, 2040–2047.

[9] 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.

[10] Arne Schmidt, Sheryl Manzoor, Li Huang, Aaron T. Becker, and Sándor Fekete, *Efficient parallel self-assembly under uniform control inputs*, IEEE Robotics and Automation Letters (2018), 1–1.

[11] 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.

[12] Y. Zhang, X. Chen, H. Qi, and D. Balkcom, *Rearranging agents in a small space using global controls*, 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 3576–3582.