Noname manuscript No.
(will be inserted by the editor)

Fast Reconfiguration of Robot Swarms with Uniform Control Signals

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

Received: date / Accepted: date

Abstract This paper investigates a simplified model of robot motion planning where particles on a board respond to global signals, causing them to move uniformly in a particular direction. We consider two types of movement commands: 1) *Steps*, which cause particles to move one unit distance in the given direction, and 2) *Tilts*, which cause particles to move maximally in the given direction. Under the overarching theme of reconfiguring robot swarms, we look at the problem of assembling general shapes both within systems that exclusively use step commands and systems that exclusively use tilt commands. We derive upper and lower bounds on the worst-case number of movements needed to reconfigure a general purpose board into a target shape. Under step transformations, we show a set of obstacles that can reconfigure n robots from any size-n shape to construct any other size-n shape with optimal $\Theta(n)$ steps, which improves on previous techniques taking $O(n^2)$ steps. We then provide a board configuration that, under tilt transformations, can construct any size-n shape (given "helper particles") in optimal $\Theta(n)$ tilts, which also improves upon the previous best known time of $O(n^2)$ tilts.

1 Introduction

Robot motion planning has been an area of interest for many years. When considering robots at the micro and nano-scale, power and bandwidth limitations often make individual robot control infeasible. Thus, abstract models of motion

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

D. Caballero · A. A. Cantu · T. Gomez · R. Schweller · T. Wylie
University of Texas Rio Grande Valley
E-mail: {david.caballero01,angel.cantu01,timothy.gomez01,robert.schweller,timothy.wylie}@utrgv.edu

A. Luchsinger
University of Texas Austin
E-mail: amluchsinger@utexas.edu

planning started considering global signals that control all robots uniformly. Perhaps the simplest (first proposed in [8]) consists of movable particles (an abstract representation of robots) that exist on a 2D grid environment with "open" and "blocked" spaces. These particles are controlled by global signals which uniformly move all particles in a particular direction when given a movement command (unless movement is prevented by a blocked space). In [8], the movement commands cause particles to move one unit distance in the given direction. Motivated by further limitations, another version of motion planning with global signals was considered in [4]. They analyze the complexity of steering particles through an environment when movement commands require them to move maximally in a direction. This spurred further investigation into computation and complexity of relocating particles [6] (recently [1–3, 5, 11, 10]).

With a better understanding of the power of this model, efforts went towards the problem of engineering particular environments to reconfigure particles into desired shapes [12]. While [8] discusses the reconfiguration of robots into particular forms, [12] has an emphasis on "building shapes" with these particles. Then in [7], they formally analyze and improve on the results from [12] by creating fixed shape "micro-factories" (or *shape builders*) that are capable of constructing a shape from a particular class of shapes (later named "drop shapes") by attaching particles to each other using maximal movement commands. In [13], they investigated the natural next step to improve the efficiency of engineered environments to build their particular shape.

Recent developments for "particle swarm shape builders" have focused on *universal constructors*, which are environments where movements can transform a particle swarm from a starting configuration into another from a given universe of configurations. In [3], they are formally introduced and two universal constructors using maximal movements are presented. One is capable of building any shape up to a given size, but allows for a relaxed notion of shape construction where "extra" particles are allowed to exist in the environment (*weak* construction). They also consider shape construction in which all particles in the environment must be considered in the final configuration (*strong* construction), and provide a universal constructor which can strongly build any "drop shape" (up to a certain size). This work is continued in [2] where they expand the set of shapes their universal constructors can build.

Previous Results. The micro-factories of [7] use "sticky" particles (which adhere to each other) for their shape construction. These micro-factories use maximal movement commands, and the number of movement commands grows linearly with the size of the shape. However, these are fixed-shape constructors (i.e., the shape created by the factory is hard-coded into the environment). More progress was made on these fixed shape micro-factories when [13] presented an "efficient" shape constructor that achieves sublinear construction times by successively combining subassemblies in "staged" assembly. Although these fixed shape results achieve outstanding runtimes for shape construction, each shape requires a unique environment designed for its construction.

Model	**Step/Tilt Complexity**		**Represent.**	**Reconfig.**	**Theorem**
	Lower	Upper			
Step	$\Omega(n)$	$O(n^2)$	Strong	Yes	**Cor. 1**,[8]
Step	$\Theta(n)$		Strong	Yes	**Cor. 1, Thm. 2**
Tilt	$\Omega(n)$	$O(n^2)$	Weak	No	**Cor. 1**, [3]
Tilt	$\Theta(n)$		Weak	No	**Cor. 1, Thm. 3**

Table 1: Size-n shape universal construction results. **Model** is which transformation the constructor uses. **Step/Tilt Complexity** is the number of transformations required for reconfiguration. **Represent** is the type of shape representation achieved by the universal constructor. **Reconfig** denotes whether or not a constructor can be reconfigured to represent any other shape in the set after a shape is built. **Theorem** is where the result can be found. The two citations provide upper bounds for their respective problems.

The idea for universal constructors is that, rather than using different environments to construct shapes, different sequences of movement commands are used to determine the shape that is constructed. In [8], the authors present a strongly universal constructor which is capable of building any size-n shape in $O(n^2)$ unit distance commands. The work of [3] presents a weakly universal constructor which can build any size-n shape in $O(n^2)$ maximal movement commands. It should be noted that this result contains the use of tile attachment, but a simple modification to the construction can eliminate that use.

The construction of patterns was explored in [14] where the authors present a configuration capable of rearranging a n tile rectangular pattern in $O(n^2)$ maximal movement commands. This problem was also studied in [9] with unit distance commands where the authors show nearly optimal step complexity universal constructors for size-n k-color patterns.

Our Contributions. We focus on a natural problem in this model of motion planning: reconfiguring particle swarms into desired shapes. We improve on the results of the universal general shape constructors from the literature by reducing the number of movements required to transform one configuration into another. We consider both variations of movement commands: unit distance commands (which we call *steps*) and maximal movement commands (*tilts*). Our contributions (in bold) are outlined in Table 1.

We derive lower bounds on the worst-case number of movements needed to reconfigure a particle swarm into a desired shape. We present two universal constructors which improve upon previous work by meeting these lower bounds. The first constructor (under step transformations) strongly builds any size-n shape in optimal $\Theta(n)$ steps. This beats the previously best runtime of $O(n^2)$ steps [8]. The second constructor (under tilt movements) weakly builds any size-n shape in optimal $\Theta(n)$ tilts. This is an improvement on the previous best runtime of $O(n^2)$ tilts [3]. We note the universal constructor from [8] has the property of being infinitely reconfigurable (any configuration it can reach can also reach all other configurations). We formally define this property, and

provide extensions to our constructors making them *reconfigurable universal shape builders*. We show these extensions do not affect the optimal runtime.

2 Preliminaries

Board. A *board* (or *workspace*) is a rectangular region of the 2D square lattice which consists of *open* and *blocked* locations. 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." Based on a geometric hierarchy [2], all our constructions use a *connected* board, i.e., the set of open spaces O is a connected shape.

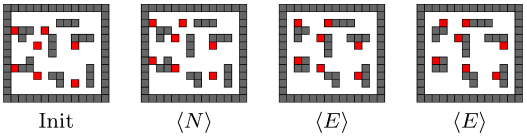
Tiles and Configurations. A *tile* (or robot/particle) is a labeled unit square that may exist on an open board location. Formally, a tile is an ordered pair (c, a) where c is a coordinate on the board, and a is a label. A *configuration* is an arrangement of tiles on a board. Formally, a configuration $C = (B, P = \{p_1 \ldots p_k\})$ consists of a board B and a set of tiles P with unique coordinates that do not overlap with the blocked locations of board B.

Step Transformation. 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. If a configuration does not change under a step transformation for direction d, the configuration is *d-terminal*. In the case a step causes a tile to leave the board, we remove it from the configuration.

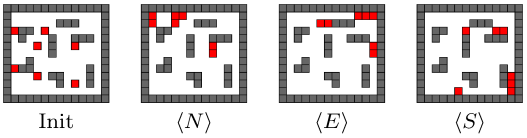
A configuration C can be *directly reconfigured* (under the step transformation) into configuration C' (denoted $C \rightarrow_S^1 C'$) if applying one step in direction $d \in \{N, E, S, W\}$ to C results in C'. Define the relation $\rightarrow_S^*$ to be the transitive closure of $\rightarrow_S^1$ and say that C can be *reconfigured* into C' if and only if $C \rightarrow_S^* C'$, i.e., C may be reconfigured into C' by way of a sequence of step transformations.

Tilt Transformation. A *tilt* is another way to turn one configuration into another. A *tilt* in direction $d \in \{N, E, S, W\}$ for a configuration is executed by repeatedly applying a step transformation in direction d until a d-terminal configuration is reached. To differentiate between reconfigurations under tilt transformations rather than step transformations, we slightly modify the notation. A configuration C can be *directly reconfigured* into configuration C' (denoted $C \rightarrow_T^1 C'$) if applying one tilt in some direction $d \in \{N, E, S, W\}$ to C results in C'. Define the relation $\rightarrow_T^*$ to be the transitive closure of $\rightarrow_T^1$ and say that C can be *reconfigured* into C' if and only if $C \rightarrow_T^* C'$, i.e., C may be reconfigured into C' by way of a sequence of tilt transformations.

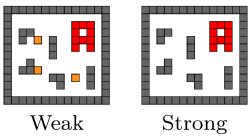
Step/Tilt Sequence. A *step sequence* is a series of steps that can be inferred from a series of directions $D_S = \langle d_1, d_2, \ldots, d_k \rangle$; each $d_i \in D_S$ implies a step in



(a) An example step sequence



(b) An example tilt sequence



(c) Representation Examples

Fig. 1: (a) An example step sequence. The initial board configuration followed by an N step, E step, and E step. (b) An example tilt sequence. Under tilt transformations, tiles move maximally in the given direction. (c) Configuration representation examples. Both of these configurations are different representations of the shape "A." The weak example contains tiles that do not contribute to the shape. In the strong example, all tiles are part of the shape.

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 transformations. An example step sequence $\langle N, E, E\rangle$ and the corresponding sequence of configurations can be seen in Figure 1a.

Similarly, a *tilt sequence* is a series of tilts which can be inferred from a series of directions $D_T = \langle d_1, d_2, \ldots, d_k\rangle$. Each $d_i \in D_T$ implies a tilt in that direction. We use the same shorthand of referring to the series of directions when discussing tilt sequences. An example tilt sequence $\langle N, E, S\rangle$ and the corresponding sequence of configurations can be seen in Figure 1b.

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 $C' \rightarrow_* C_i \ \forall \ C_i \in \mathcal{C}$.

Reconfigurable Set. A set of configurations $\mathcal{C}$ is a reconfigurable set if and only if $\forall\ C_i, C_j \in \mathcal{C}\ C_i \rightarrow_* C_j$.

Configuration Representation. A configuration may be interpreted as having constructed a "shape" in a natural way. Define a shape to be a connected subset $s \subset \mathbb{Z}^2$. A configuration C may represent a shape s if C contains a collection of "output labeled" tiles $L \subseteq P$ whose coordinates are exactly the set of points of some translation $t(s)$. We say C *strongly* represents s if $|P| = |L| = |s|$ (i.e., all tiles in the configuration are used for the shape representation). A weaker version allows for some "helper" tiles to exist in the configuration and not count towards the represented shape. In this case, we say a configuration *weakly* represents s if $|P| > |L|$, and no tile $p \in P \backslash L$ has the output label. Figure 1c illustrates the different types of representations.

Universal Shape Builder. Given the concept of shape representation, we say a configuration C' is *universal* for a set of shapes $\mathcal{S}$ 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 *strongly universal* for $\mathcal{S}$ if each $s \in \mathcal{S}$ is strongly represented by some $C \in \mathcal{C}$. If C' is universal for S, but not strongly universal, then we say that C' is *weakly universal*.

Reconfigurable Universal Shape Builder. A set of configurations $\mathcal{C}$ are a *Reconfigurable Universal Shape Builder* for a set of shapes $\mathcal{S}$ if and only if 1) each $s \in \mathcal{S}$ is represented by a unique $C \in \mathcal{C}$, and 2) $\mathcal{C}$ is a reconfigurable set.

Worst-Case 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 $\max(\{d(C, u)|u \in U\})$. We extend this notion to reconfigurations performed under the tilt transformation, and refer to the maximum number of tilts required for reconfiguration as the worst-case tilt complexity for the universal configuration.

Worst-Case Step/Tilt Complexity for Reconfigurable Sets. Consider a reconfigurable set $\mathcal{C}$. The worst-case step complexity for $\mathcal{C}$ is the maximum number of steps between two configurations in $\mathcal{C}$. For each $C_i, C_j \in C$, let $d(C_i, C_j)$ denote the length of the smallest step sequence from C_i to C_j. The worst-case step complexity of $\mathcal{C}$ is defined to be $\max(\{d(C_i, C_j)|C_i, C_j \in \mathcal{C}\})$. Again, we can extend this notion to reconfigurations performed under the tilt transformation to define the worst-case tilt complexity for reconfigurable sets.

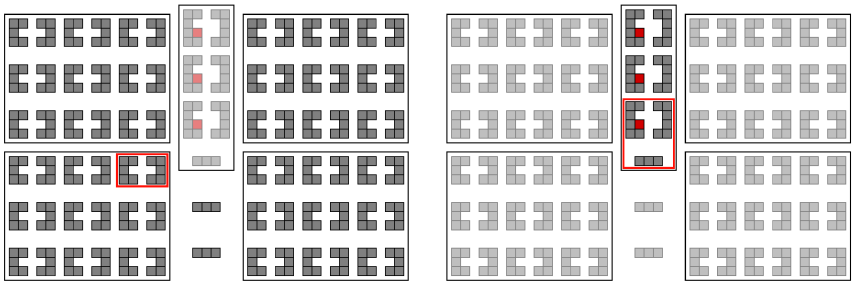
3 Lower Bounds for Shape Construction

Theorem 1 *Any universal configuration for a set of configurations U under either the step transformation or the tilt transformation has worst-case $\Omega(\log |U|)$ step/tilt complexity.*

Proof Let r denote the worst-case step or tilt complexity for a universal configuration C that is universal for U. We can upper bound the number of distinct configurations C may reach by counting the number of distinct length-r step/tilt sequences. As each step/tilt in a sequence consists of one of four possible directions, there are exactly 4^i distinct length-i step sequences, and $4 \cdot 3^{i-1}$ distinct tilt sequences (since repeating a tilt within a sequence does not change a configuration). Thus, the total number of distinct step/tilt sequences of length at most r is upper bounded by

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

As the number of distinct length-r step/tilt sequences is an upper bound on the number of distinct configurations reachable within r steps/tilts, this number must be at least $|U|$, which implies that $r = \Omega(\log |U|)$. □



(a) Pixel Gadgets (b) Fuel Gadgets

Fig. 2: Illustration of the board. Outlined in red are the (a) head pixel gadget and the (b) fuel depositor gadget.

Corollary 1 *Any universal configuration for connected shapes of size-n has worst-case $\Omega(n)$ step/tilt complexity.*

Proof First note that there are at least $2^{\Theta(n)}$ distinct connected shapes of size-n, and thus any universal configuration for this set of shapes must be universal for a set of configurations of at least this cardinality. Therefore, by Theorem 1, we get a worst-case step/tilt complexity of $\Omega(\log(2^{\Theta(n)})) = \Omega(n)$. □

Corollary 2 *Any reconfigurable set for connected shapes of size-n has worst-case $\Omega(n)$ step/tilt complexity.*

Proof Observe that any configuration in a universal reconfigurable set is also universal configuration for that set, so any lower bounds on universal configurations also hold for reconfigurable sets. □

4 Fast Size-n Shapes Under Step Transformations

In this section we give a reconfigurable universal shape builder for size-n shapes under the step transformation that achieves optimal $O(n)$ step complexity. We first present a universal shape builder, and then expand on the construction to make it reconfigurable.

4.1 Universal Shape Builder

The mechanics of the constructor introduced here are designed to mimic an additive manufacturing method of construction where structures are gradually built by material depositors that move around a surface ejecting material and building the object piece by piece.

The construction area consists of four $n \times n$ groups of *pixel* gadgets organized in a grid− each occupying a 5×3 region on the board used to hold a single tile in place. The construction area is divided into a left and right $2n \times n$ region due to the allocation of the *fuel* gadgets, each a 5×4 region initialized

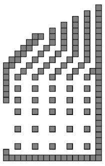
Name	Sequence
Insert	$\langle E, S^4, W^7, S^2, W, E, N, W, S\rangle$
Move Up	$\langle N^5, W, S, E, N, W, S\rangle$
Move Down	$\langle S^5, W, E, N, W, S\rangle$
Move Left	$\langle N^3, W^6, S^3, W^8, S^2, W, S, W, E, N, W, S\rangle$
Move Right	$\langle N^3, E^6, S^3, E^8, S^2, E, S, E, W^2, E, N, W, S\rangle$

Table 2: Step sequences used for the construction of size-n shapes. As a standard convention for languages, exponents represent repeating the string (or sequence of moves) the number of times in the exponent.

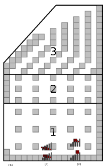
with an output tile for a total of n gadgets, spaced out in the manner illustrated in Figure 2a. In this way, shapes are built when the output tiles from the fuel gadgets are held in pixel gadgets that are positioned relative to the tile's position in the shape. All tiles are placed in the construction area by the fuel *depositor* gadget, the bottommost fuel gadget that utilizes the 1×3 line of concrete tiles beneath it to disalign the fuel with the one being deposited. Moreover, all tiles enter the construction area through the *head* pixel gadget, which is the pixel gadget diagonal to the fuel depositor. Given the dimensions of the construction area, a size-n shape can be translated such that any position in the shape can be put into the head pixel gadget without the shape extending beyond the construction area. This is true so long as the step sequences, as shown in Table 2, are used in the shape construction process.

Tiles are inserted into the construction area by performing the *insert* sequences, causing the withdrawn tile to disalign itself from the rest of the tiles in the fuel gadgets and simultaneously move them to the fuel gadget below them. This sequence also places the withdrawn tile inside of the head pixel gadget while maintaining the tiles already in the construction area inside of their pixel gadgets. Tiles can be moved around the construction area using the other sequences. The *move up* and *move down* sequences are straightforward, but the *move right* and *move left* sequences are more complex. These sequences move all tiles in pixel gadgets to the corresponding neighboring pixel gadgets in the given direction, while taking into consideration tiles that travel across to the left or right region of the construction area. The left and right move sequences begin by first moving the tiles that are in the left or right regions to their respective pixel gadgets, and then move the tiles that move from region to region into to the pixel gadgets of that region. Examples of the tile movements are shown in Figure 6. Any construction of a size-n shape, using the board configuration presented in this section, is therefore describable as some combination of these step sequences.

After all the tiles have been placed in the construction area, moving them on to the left or right region of the board allows them to be withdrawn from the pixel gadgets. They are vertically spaced by four spaces and horizontally by five. The group of tiles can then be sent through the *funneling* gadget (Figure 3) that is used to remove the spaces and group the tiles so that they take the form of the shape. This gadget was originally presented in [9], but we give an overview of the gadget for clarity.

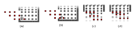


(a)



(b) (c) (d)

Fig. 3: (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.



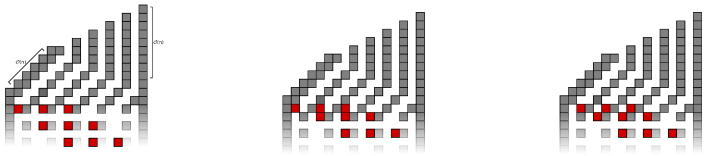
(a) (b) (c) (d)

Fig. 4: (a-b) Reducing the horizontal distance between the rightmost column of tiles. (c-d) Reducing the vertical distance between the topmost row of tiles.

4.2 Funneling Gadget

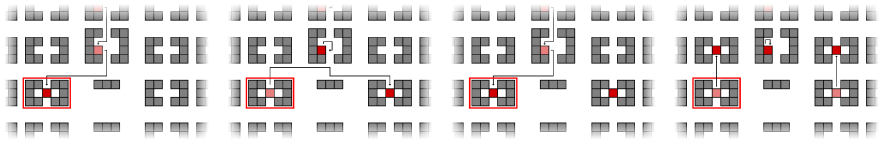
The *funneling* gadget is designed to take a group of tiles separated by a constant amount of spaces and coalesce them into a desired shape. The architecture and sections of the funneling gadget are illustrated in Figure 3. Let α and β be constants equaling the largest number of vertical and horizontal spaces, respectively, that separates a tile from its neighbor in the shape. The first section of the funneling gadget reduces β so that the largest horizontal separation between two tiles is two. Section two takes the group of tiles 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 tiles as the desired shape outside the funneling gadget. However, this process skews the shape in one direction. Thus, our constructor builds a shape skewed in the *opposite* direction, thus negating the skewing effect of the funneling gadget.

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 4. To reduce β to two, we place the rightmost column of the group of tiles between the two leftmost columns of concrete tiles (Figure 4b). By stepping in the $\langle E \rangle$ direction enough times, the second to rightmost column of the group of tiles will meet the leftmost column of section one, reducing the spaces between the two columns of tiles. After repeating this process for every column of tiles, section one will contain the group of tiles vertically separated by α and horizontally separated by a distance of two.



(a) (b) (c)

Fig. 5: Making the rows of tiles adjacent by using section three.



(a) Insert Tile (b) Move Right (c) Insert Tile (d) Move Up

Fig. 6: A simple example of how the given step sequences can be used to insert and move tiles inside the pixel gadgets.

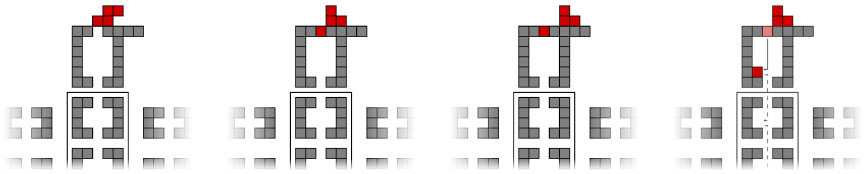
Section two is a grid-like configuration of concrete tiles that are vertically separated by one space and horizontally by two spaces. The same basic process is applied here, but we instead place the rows of tiles in-between the rows of concrete tiles and perform sufficient steps in the $\langle N \rangle$ direction (Figure 5).

By positioning the group of tiles in **Section three** as depicted in Figure 5a, 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 tiles from the funneling gadget, bringing together each column of tiles and outputting the desired shape at the top of the gadget, (Figures 3c, 3d).

4.3 Reconfigurable Universal Shape Builder

The universal shape constructor can be extended to a reconfigurable universal shape builder by taking the output of the funneling gadget and breaking its components apart and placing each tile back into the fuel gadgets.

A shape can be decomposed with an additional gadget placed above the topmost fuel gadget as shown in Figure 7. This gadget consists of a similar structure as the fuel gadgets, although with more space inside the gadget and an additional length $n-2$ horizontal line of concrete tiles extending the top area of the gadget. Shapes moved to the top of this gadget can have their tiles removed one by one by placing a tile directly above the top opening and executing the sequence $\langle S, E \rangle$. By performing the *re-fuel* sequence $\langle S^4, W \rangle$, the tile just removed will be moved towards the bottom left corner of this gadget. When this is done for the next tile, the tile inside the gadget will be moved simultaneously towards the topmost fuel gadget and be placed on the



(a) Placement (b) $\langle S \rangle$ (c) $\langle E \rangle$ (d) Re-fuel Sequence

Fig. 7: (a-c) Removing a single tile from a size 4 shape using the topmost opening of the gadget. The additional row of tiles are to the right of the fuel gadget to keep the assembly from moving downward. (d) Performing the *re-fuel* sequence will move the tile to where the arrow points. The dotted arrows indicate alternate paths that tiles would take if they resided in alternate locations.

bottom left corner of that fuel gadget similar to the second tile just placed in the gadget above it. When this is repeated for every tile in the shape, every tile that was once in the shape will have been placed into the fuel gadget, and thus another size-n shape can be constructed. Since the funneling gadget must consider any extreme case of a size-n shape (e.g., a $1 \times n$ or $n \times 1$ shape), the dimensions of the funneling gadget for the reconfigurable constructor is designed to accommodate shapes that fit into an $n \times n$ bounding box.

Theorem 2 *For any positive integer n there exists a universal reconfigurable shape builder for all connected size-n shapes with worst-case step complexity $\Theta(n)$ and board size $O(n^2)$.*

Proof We use the above board configuration $C = (B, P)$ that is composed of an arrangement of pixel and fuel gadgets. Pixel gadgets are arranged on the board in a grid-like fashion, inducing a grid graph of dimension $2n \times 2n$ which defines the construction area. Moreover, the board consists of n fuel gadgets placed in the middle of the construction area, each containing one output tile. We define various step sequences that dictate how to maneuver the tiles around the board, including the step sequence to insert tiles into the construction area. The *head* pixel gadget is the recipient of all the tiles deposited into the construction area by the fuel *depositor* gadget. Any position in a size-n shape can be translated to reside in the head pixel gadget without over extending beyond the construction area. Therefore, if we consider a spanning tree of a size-n shape, we can traverse through the tree mapping the traversal to the step sequences defined above and simultaneously place tiles in the construction area for every new vertex we visit. Thus, every tile can be inserted in its appropriate position in a size-n shape in runtime $O(n)$. After all the tiles have been placed in the construction area, they can be withdrawn from the construction area (assuming all the tiles are on either the left or right region) and sent to the funneling gadget to coalesce the tiles into the shape, which also has a runtime of $O(n)$ steps. In addition to shape construction, we demonstrate a way to decompose a size-n shape and place back the composite tiles into the fuel

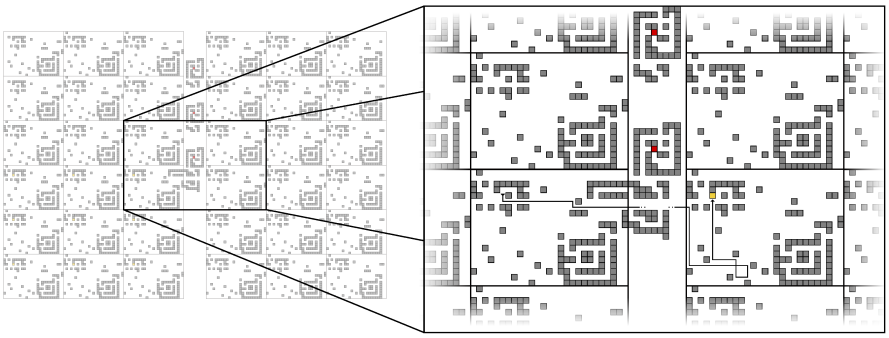
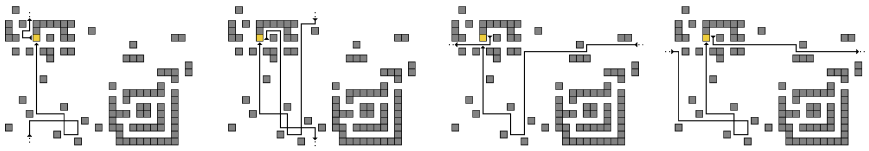


Fig. 8: The construction area of the board. The pixel gadget boundaries are outlined to show how they are placed on the board.



(a) North sequence (b) South sequence (c) West sequence (d) East sequence

Fig. 9: Tilt sequences to send and receive robots.

gadgets. In doing so, we allow ourselves to build any other size-n shape. Thus, configuration C is a reconfigurable shape builder for all size-n shapes with construction runtime $\Theta(n)$ and board size $O(n^2)$. □

5 Fast Size-n Shapes under Tilt Transformations

In this section, we present a *weakly* reconfigurable universal shape builder for size-n shapes under full-tilt transformations. Previous work showed the existence of a weakly universal constructor for general shapes [3], but the runtime scales linearly in the size of the bounding box (it's height times it's width) rather than size of the shape. The universal constructor presented in this section achieves optimal $\Theta(n)$ construction time to weakly build size-n shapes. The construction uses the same basic idea as the single step construction, but a lot of additional geometric complexity is required to deal with the less precise tilt operation.

5.1 Constructor

Similar to the single-step constructor, the construction area consists of four groups of $n \times n$ pixel gadgets organized in a grid-like fashion separated by a column of n fuel gadgets located in-between the left and right regions of

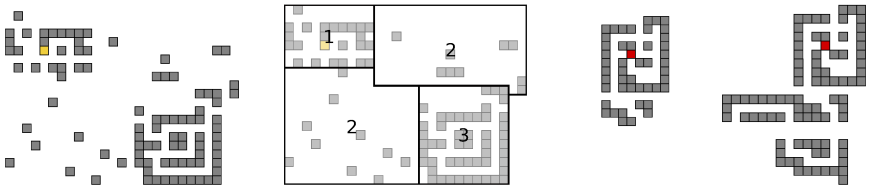


Fig. 10: (a) Pixel Gadget. (b) Pixel gadget sections. (c) Fuel Gadget. (d) Fuel Depositor Gadget.

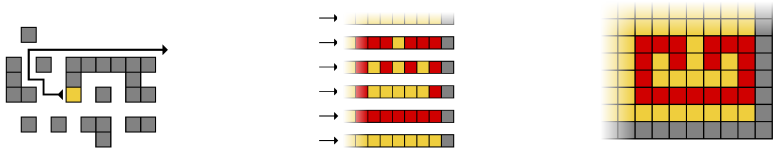
Name	Sequence
Insert Pixel	$\langle E, N, W, N, (W, S)^2, (E, N, E, S)^2 \rangle$
Reposition	$\langle (W, N)^2, E, S, (E, N)^2, W, N, E, N, (W, S)^2, (W, N)^2 \rangle$
Prepare Board	$\langle W, S, (E, S)^2, W, N \rangle$
Move South	$\langle E, N, (E, S)^2, W, S, (W, N)^2 \rangle$
Move East	$\langle (E, S)^4, (W, N)^2 \rangle$
Move West	$\langle E, S, (W, S)^2, (W, N)^2 \rangle$
Move North	$\langle W, N, E, N, E, S, (W, N)^2 \rangle$

Table 3: Tilt sequences used during the construction process.

the construction area. The full tilt construction is initialized with an $n \times n$ bounding box of helper tiles inside the pixel gadgets in the position shown by the tile in Figure 10a (each pixel gadget has one helper tile). Shapes in this model are built by replacing the helper tiles of the bounding box with the shape output tiles from the fuel gadgets, gradually placing them in their appropriate positions in the bounding box.

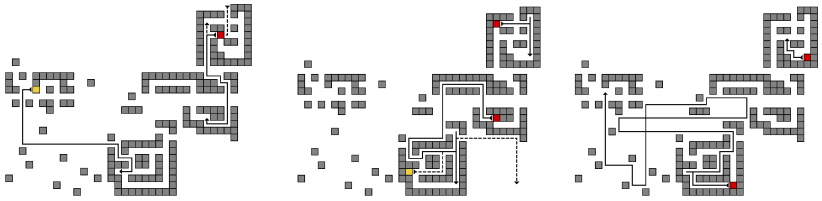
Pixel gadgets in this construction are significantly more complex than the single-step version, consisting of three sections that are each used at different points in the construction process. Sections one and two of the pixel gadgets are made of concrete tiles that aid to send and receive tiles from adjacent pixel gadgets. The tiles on the board are moved around the construction area using the tilt sequences defined in Table 3 and shown in Figure 9, where the arrows depict where the tile will exit a pixel gadget and enter the neighboring pixel gadget. When output tiles are to be placed inside the construction area, tiles in the bounding box are placed first in the third section of their pixel gadget to prevent any decomposition of the bounding box.

This constructor includes some pre and post-processing procedures, shown in Figure 12, that are used during the construction process. The *prepare pixel* sequence is used before performing the *insert pixel* sequence in order to move all the tiles of the bounding box inside the third section of their respective pixel gadget. The *insert pixel* sequence ejects output tiles from the fuel depositor gadgets into the concrete structure below (Figure 10d), which will put the tile inside the head pixel gadget, allowing it to interfere with, and thus replace, the helper tile. This will discard the helper tile from the board. The *reposition* sequence is used after the *insert pixel* sequence in order to withdraw the tiles



(a) Extract Shape (b) Combine Columns (c) Combine Rows

Fig. 11: (a) Removing the elements from pixel gadgets. (b) The columns of the bounding box will combine via the collision with the concrete tiles after performing the sequence Extract Shape. (c) Performing a $\langle S \rangle$ tilt will combine all rows of the bounding box.



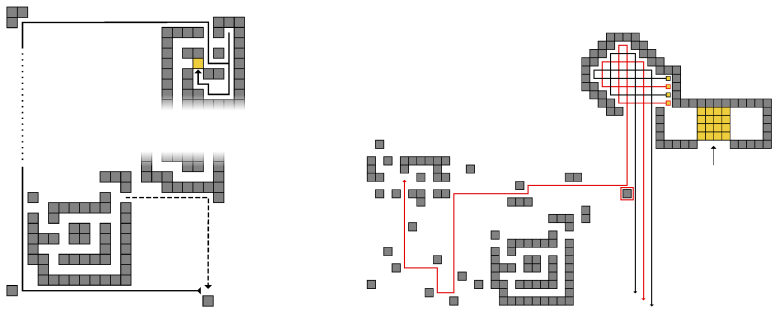
(a) Prepare Board sequence (b) Insert Pixel sequence (c) Reposition sequence

Fig. 12: The dashed line in (a) shows the path the tile from the fuel gadget above will take. The dashed line in (b) depicts the path the helper tile will take when substituting the robot for it.

of the bounding box from section three and allow them to continue moving around the construction area. After all of the output tiles have been placed in the construction area, all tiles (input and helper) are withdrawn from the pixel gadgets with the sequence $\langle W, N, W, N, E \rangle$ (Figure 11a), causing all of the tiles to be tilted to the right side of the board. Placing concrete tiles on the right side of the construction area that are aligned with the launching path of the pixel gadgets will cause all of the columns to combine in the manner shown in Figure 11b, which can be followed by a $\langle S \rangle$ tilt to combine all of the rows using the concrete tiles shown in Figure 11c, thus yielding the shape.

5.2 Reconfigurable Universal Shape Builder

Achieving reconfigurable universality is similar to the single-step method, but this version requires more concrete tiles and we must store or reuse the helper tiles. Rather than discarding the helper tile from the board as previously mentioned, we make it take the path shown in Figure 13a by including the concrete tile beneath the dotted arrow. As the *reposition* sequence is being performed, this helper tile is sent to collide with the concrete tiles shown in the figure at the topmost fuel gadget, allowing the helper tile to enter the gadget. After all the output tiles have been placed on the board, the fuel gadgets will



(a) Re-fuel (b) Dispenser Gadget

Fig. 13: (a) Performing the *reposition* sequence will cause the discarded helper tile to enter the topmost fuel gadget. (b) The paths that the dispenser gadget sends the tiles of the bounding box.

contain n helper tiles. When the rows and columns of the bounding box are put together, we move the bounding box into the *dispenser* gadget in order to begin removing one column at a time from the bounding box and placing them back into the pixel gadgets.

We place this dispenser gadget directly above the output region of the board (Figure 13b), and thus placing the bounding box inside with a simple $\langle N \rangle$ tilt. Simply repeating the *move west* sequence will cause each column, starting with the leftmost one, to be sliced off the bounding box and sent through the dispenser gadget. The dispenser gadget will separate the tiles and send them through a path that will collide with single concrete tiles that are aligned with the pixel gadgets on the east side of the board such that finishing the *move west* sequence will cause the tiles to enter the pixel gadgets. We repeat the *move west* sequence until all of the tiles are back into the construction area. To return the output tiles back into the fuel gadgets, we simply rebuild the same shape (now in a different translation, but connected nonetheless), and repeat this process.

Theorem 3 *For any positive integer n there exists a configuration C that is a reconfigurable weakly universal shape builder for all size-n shapes. The configuration has board size $O(n^2)$ and construction runtime $\Theta(n)$.*

Proof The construction above uses a bounding box of $n \times n$ helper tiles inside the pixel gadgets that are replaced by output tiles from the fuel gadgets. When the shape is built inside the bounding box, all tiles (output and helper) can be withdrawn from the construction area (with the sequences given) and combined on the right side of the construction area. To make this into a reconfigurable universal shape builder, the dispenser gadget can be added above the output region of the constructor, which can break apart the elements of the bounding box and place them back into the pixel gadgets. Moreover, instead of discarding the helper tiles, concrete tiles can be arranged on the board to maneuver the helper tiles back into the topmost fuel gadget. Once the reconfiguration process is done, the output tiles inside the pixel gadgets are replaced

with the helper tiles, and this process is repeated once more to end up with the output tiles back in the fuel gadgets and the helper tiles all in the bounding box. Thus, the configuration $C = (B, P)$ is a universal reconfigurable constructor that *weakly* builds size-n shapes in optimal construction runtime $\Theta(n)$ and board size $O(n^2)$. □

6 Future Work

In this paper we show the existence of a universal reconfigurable shape constructor in both the single step and full tilt models where both are optimal in the number of tilts needed to assemble the shape. The full-tilt constructor only weakly assembles the shape. A major open question is whether or not a strongly universal constructor exists for the full-tilt model. Only a few classes of shapes are known with strong universal constructors (drop shapes [3, 7], 2-cuttable shapes [13], and the drop shape hierarchy [2]). Finally, different types of board geometries have been defined in previous work with the least restrictive being connected (all the open spaces are connected) and the most being a rectangular frame. All of our constructors use connected board geometry but previous work has shown constructors that use a simple board. Is it harder to construct shapes with more constrained boards? Can we still achieve fast construction with simpler boards or does the lower bound increase?

References

1. Balanza-Martinez, J., Caballero, D., Cantu, A.A., Gomez, T., Luchsinger, A., Schweller, R., Wylie, T.: Relocation with uniform external control in limited directions. In: Proc. of the Japan Conf. on Discrete and Computational Geometry, Graphs, and Games, JCDCGGG'19, pp. 39–40 (2019)
2. Balanza-Martinez, J., Gomez, T., Caballero, D., Luchsinger, A., Cantu, A.A., Reyes, R., Flores, M., Schweller, R.T., Wylie, T.: Hierarchical shape construction and complexity for slidable polyominoes under uniform external forces. In: Proc. of the ACM-SIAM Symposium on Discrete Algorithms, SODA'20, pp. 2625–2641 (2020)
3. Balanza-Martinez, J., Luchsinger, A., Caballero, D., Reyes, R., Cantu, A.A., Schweller, R., Garcia, L.A., Wylie, T.: Full tilt: Universal constructors for general shapes with uniform external forces. In: Proc. of the ACM-SIAM Symposium on Discrete Algorithms, SODA'19, pp. 2689–2708 (2019)
4. Becker, A.T., Demaine, E.D., Fekete, S.P., Habibi, G., McLurkin, J.: Reconfiguring massive particle swarms with limited, global control. In: Algorithms for Sensor Systems, pp. 51–66 (2014)
5. Becker, A.T., Demaine, E.D., Fekete, S.P., Lonsford, J., Morris-Wright, R.: Particle computation: complexity, algorithms, and logic. Natural Computing **18**, 6751–6756 (2019). DOI 10.1007/s11047-017-9666-6
6. Becker, A.T., Demaine, E.D., Fekete, S.P., McLurkin, J.: Particle computation: Designing worlds to control robot swarms with only global signals. In: Proc. of the IEEE Inter. Conf. on Robotics and Automation, ICRA'14, pp. 6751–6756 (2014)
7. Becker, A.T., Fekete, S.P., Keldenich, P., Krupke, D., Rieck, C., Scheffer, C., Schmidt, A.: Tilt assembly: Algorithms for micro-factories that build objects with uniform external forces. Algorithmica (2018)
8. Becker, A.T., Habibi, G., Werfel, J., Rubenstein, M., McLurkin, J.: Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In: The Inter. Conf. on Intelligent Robots and Systems, pp. 520–527 (2013)

9. Caballero, D., Cantu, A.A., Gomez, T., Luchsinger, A., Schweller, R., Wylie, T.: Building patterned shapes in robot swarms with uniform control signals. In: Proc. of the Canadian Conference on Computational Geometry, CCCG'20, pp. 59–62 (2020)
10. Caballero, D., Cantu, A.A., Gomez, T., Luchsinger, A., Schweller, R., Wylie, T.: Hardness of reconfiguring robot swarms with uniform external control in limited directions. Journal of Information Processing (to appear) (2020). Arxiv:2003.13097
11. Caballero, D., Cantu, A.A., Gomez, T., Luchsinger, A., Schweller, R., Wylie, T.: Relocating units in robot swarms with uniform control signals is PSPACE-complete. In: Proc. of the Canadian Conference on Computational Geometry, CCCG'20, pp. 49–55 (2020)
12. Manzoor, S., Sheckman, S., Lonsford, J., Kim, H., Kim, M.J., Becker, A.T.: Parallel self-assembly of polyominoes under uniform control inputs. IEEE Robotics and Automation Letters **2**(4), 2040–2047 (2017). DOI 10.1109/LRA.2017.2715402
13. Schmidt, A., Manzoor, S., Huang, L., Becker, A.T., Fekete, S.: Efficient parallel self-assembly under uniform control inputs. IEEE Robotics and Automation Letters (2018). DOI 10.1109/LRA.2018.2853758
14. Zhang, Y., Chen, X., Qi, H., Balkcom, D.: Rearranging agents in a small space using global controls. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3576–3582 (2017)