# Universal Reconfiguration of Facet-Connected Modular Robots by Pivots: The $O(1)$ Musketeers

**Hugo A. Akitaya**
Tufts University, Medford, MA, USA
hugo.alves_akitaya@tufts.edu

**Esther M. Arkin**
State University of New York at Stony Brook, NY, USA
esther.arkin@stonybrook.edu

**Mirela Damian**
Villanova University, PA, USA
mirela.damian@villanova.edu

**Erik D. Demaine**
Massachusetts Institute of Technology, Cambridge, MA, USA
edemaine@mit.edu

**Vida Dujmović**
University of Ottawa, Canada
vida.dujmovic@uottawa.ca

**Robin Flatland**
Siena College, Loudonville, NY, USA
flatland@siena.edu

**Matias Korman**
Tufts University, Medford, MA, USA
Matias.Korman@tufts.edu

**Belen Palop**
Universidad de Valladolid, Spain
belen.palop@uva.es

**Irene Parada**
Graz University of Technology, Austria
iparada@ist.tugraz.at

**André van Renssen**
The University of Sydney, Australia
andre.vanrenssen@sydney.edu.au

**Vera Sacristán**
Universitat Politècnica de Catalunya, Barcelona, Spain
vera.sacristan@upc.edu

## Abstract

We present the first universal reconfiguration algorithm for transforming a modular robot between any two facet-connected square-grid configurations using pivot moves. More precisely, we show that five extra "helper" modules ("musketeers") suffice to reconfigure the remaining $n$ modules between any two given configurations. Our algorithm uses $O(n^2)$ pivot moves, which is worst-case optimal. Previous reconfiguration algorithms either require less restrictive "sliding" moves, do not preserve facet-connectivity, or for the setting we consider, could only handle a small subset of configurations defined by a local forbidden pattern. Configurations with the forbidden pattern do have disconnected reconfiguration graphs (discrete configuration spaces), and indeed we show that they can have an exponential number of connected components. But forbidding the local pattern throughout the configuration is far from necessary, as we show that just a constant number of added modules (placed to be freely reconfigurable) suffice for universal reconfigurability. We also classify three different models of natural pivot moves that preserve facet-connectivity, and show separations between these models.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Reconfiguration, geometric algorithm, pivoting squares, modular robots

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2019.3

**Related Version** A full version of the paper is available at `https://arxiv.org/abs/1908.07880`.

**Funding** *Hugo A. Akitaya*: Supported by NSF CCF-1422311 and CCF-1423615.
*Esther M. Arkin*: Partially funded by NSF (CCF-1526406).
*Erik D. Demaine*: Supported in part by NSF ODISSEI grant EFRI-1240383 and NSF Expedition grant CCF-1138967.
*Belen Palop*: Partially supported by MTM2015-63791-R (MINECO/FEDER).
*Irene Parada*: Supported by the Austrian Science Fund (FWF): W1230.

*André van Renssen*: Supported by JST ERATO Grant Number JPMJER1201, Japan.
*Vera Sacristán*: Partially supported by MTM2015-63791-R (MINECO/FEDER) and Gen. Cat. DGR 2017SGR1640.

**Acknowledgements** This research started at the 32nd Bellairs Winter Workshop on Computational Geometry in 2017. We want to thank all participants for the fruitful discussions on the topic. We would also like to thank the reviewers for their insightful and valuable comments.

## 1 Introduction

*Shape shifting* is a powerful idea in science fiction: T-1000 robots (from *Terminator 2: Judgement Day*), Changelings (from *Star Trek: Deep Space 9*), Symbiotes (from *Venom*), Mystique (from *X-Men*), and Metamorphagi (from *Harry Potter*) all have the ability to transform their shape nearly arbitrarily. How can we make shape shifting into science?

*Modular robots* [5, 19, 22] are perhaps the best answer to this question. The idea to build a single "robot" out of many small units called *modules*, each of which can attach and detach from each other, move relative to each other, communicate with each other, and compute. Modular robots offer extreme adaptability to changing environment or user needs, in particular by reconfiguring the modules into exponentially many effective shapes of the overall robot. Modularity also offers a practical future for manufacturing (identical modules can be mass-produced, making them relatively cheap) makes robots easy to repair by just replacing the broken modules, and makes it possible to re-use components from one robot/task to another.

For computational geometry, modular robots offer exciting challenges: what shapes can a modular robot self-reconfigure into, and what are good algorithms for reconfiguration? According to [19], the main difficulties in self-reconfiguration are the physical motion constraints of the modules themselves, connectivity requirements for the robot to hold together, collisions between moving and/or static modules, and "deadlocks" where no module can move or some module gets "trapped" within the configuration.

The wide diversity of mecatronic solutions to modular robots can be characterized from a geometric viewpoint by three key properties: (1) the lattice, (2) connectivity requirement, and (2) allowed moves.

**Lattice.** Most modular robots follow a space-filling lattice structure (e.g., squares or hexagons in 2D, or cubes in 3D), to simplify both reconfiguration and the characterization of possible shapes. Pure lattice modular robots [13, 6, 10, 17, 2, 20] have one robot per lattice element and always remain on the lattice, while hybrid modular robots [14, 18, 16, 23] also allow units move out of the lattice. We focus here on the well-studied square lattice, though we suspect our results can be generalized to cube lattices.

**Connectivity requirement.** A modular robot generally needs to be connected at all times while reconfiguring, so that the modules do not fall apart. The most common and practical constraint is that the modules are always *facet-connected*, meaning a connected *facet-adjacency graph* where vertices represent modules and edges represent adjacencies by shared facets (edges in 2D). The exception is that the moving module is excluded from this graph during each move, meaning that other modules must be facet-connected while the moving module may briefly disconnect during the move. A weaker connectivity constraint, considered in some theoretical research [4, Ch. 4], is that the robot is connected via shared vertices. In such case, reconfiguration is always possible. We focus here on the more challenging facet-connectivity constraint.

**Allowed moves.** One of the most popular models is *sliding squares/cubes* [8, 7, 1], illustrated in Figure 1 left. In this case, modules live in a square or cube lattice, move by sliding relative to each other, and require facet-connectivity. For this model, universal reconfiguration is possible between any two facet-connected configurations, in any dimension [7, 1].

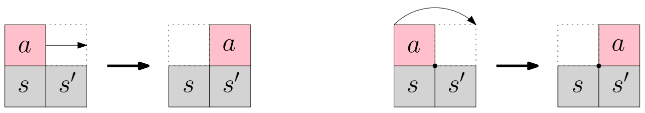

**Figure 1** Two ways a module $a$ starting above module $s$ can move to the adjacent lattice position, above module $s'$. Left: sliding. Right: pivoting. Pivoting requires more free space to execute.

We focus here on a more challenging model, *pivoting squares/cubes* [21, 20, 4], illustrated in Figure 1 right. In this case, modules live in a square or cube lattice, move by rotating relative to each other, and require facet-connectivity. The key difference is that a module needs two additional squares/cubes of empty space in order to pivot, whereas a slide just needs the destination square/cube to be empty. Unfortunately, some configurations are *rigid* in this model, meaning that no module can move without disconnecting the robot.

Rigid configurations appear also in the sliding square model when the sliding capability is restricted to turning corners [12]. However, in this model the existence of free space around the modules does not guarantee reconfigurability, while in the pivoting squares model it does, as we will discuss.

As a consequence, all known reconfiguration algorithms for pivoting squares/cubes are somehow partial. One algorithm follows some heuristics without a termination guarantee [3] (see also [11] for heuristics for hexagons). A recent algorithm guarantees reconfiguration by forbidding one or more local patterns in both the start and goal configurations [20], essentially preventing narrow holes in the shape. (A similar result was obtained for hexagons [15].) These assumptions severely restrict the possible shapes that can be reconfigured, to a $o(1)$ fraction. The absence of such local patterns though is far from being necessary for reconfigurability. In 3D, some further strong conditions are added, such as that every hole must be orthogonally convex [20].

**Our results.** Our main result is that *universal* reconfiguration is possible if we allow the addition of a constant number of (five) extra "helper" modules, which we call *musketeer modules.*[1] The key is that these musketeer modules are not considered part of the initial or target shape, and thus we are free to place them where we like (in particular, along the external boundary of the robot). Surprisingly, this small amount of additional freedom is enough to achieve universal reconfiguration. In fact, we prove in Section 4 that five musketeer modules are both sufficient and sometimes necessary to solve any reconfiguration under our strategy. Our algorithm is based on the old idea of following the right-hand rule to escape a maze [9]. The number of pivots it makes is $O(n^2)$, which is optimal in the worst case by an earth-moving lower bound: each robot may need to move a distance of $\Theta(n)$.

This result can be seen as proving connectivity of the *reconfiguration graph* $\mathcal{G}_{n,k}$, where vertices represent facet-connected configurations of $n$ modules and edges represent valid pivot moves, with the addition of $k \geq 5$ musketeer modules. With $k = 0$ musketeers, $\mathcal{G}_{n,k}$ is known to be disconnected. Surprisingly, there have been no (successful) attempts to understand

[1] *The Three Musketeers* is a story about four musketeers. This paper is a story about five musketeers.

the structure of this reconfiguration graph. In Section 3, we analyze the structure of this reconfiguration graph. Specifically, we prove that $\mathcal{G}_{n,0}$ can have an exponential number of connected components of exponential size, and in some models, can have an exponential number of singleton connected components (rigid configurations); while in other models, the reconfiguration graph cannot have any singleton connected components.

The other main contribution of this paper is to precisely define a variety of natural models for pivot moves. Pivoting is naturally defined as the rotation of one module about one of its vertices that is shared with a (static) module. But there are some subtleties in this definition depending on exactly which modules must be facet-connected at what times. (Obviously, for example, the moving module is not facet-connected to the others during the move.) In Section 2, we define three nested models, each at least as powerful as the previous, and in Section 3, we prove strict separations between these models. Our analysis of connected components in the reconfiguration space (in Section 3) also consider the effects of these different models. We conclude with open problems in Section 5.

## 2 Models and Definitions

### 2.1 Pivot Moves

In a square grid, the fact that two squares may share a vertex without actually sharing an edge opens a wider range of possibilities for the pivoting move. Refer to Figure 2. The most restrictive set of moves (Set 1 in Figure 2) requires module $a$ to be facet-adjacent to module $s$ and to rotate about one of the two vertices of the edge they share. Such move can be a 90° or a 180° rotation, depending on whether or not $s$ has a neighboring module $s'$ adjacent to it through the other edge of $s$ incident to the rotation center, and of course, requires the goal grid position to be empty and some intermediate positions to be (at least partially) clear. These cells are depicted in white in Figure 2.

The authors of [20] propose an expanded set of moves (Set 2 in Figure 2) that allows module $a$ to rotate 90° about module $s$ even when $s'$ is not present, as long as module $a$ is again facet-adjacent to another module $t$ at the end of the move. Since their reconfiguration algorithm relies on reversible moves, this implies allowing also the reverse move: module $a$ can rotate 90° about a vertex of another module $s$ incident to $a$, without requiring $s$ to be facet-adjacent to $a$, as long as $a$ is facet-adjacent to some module before performing the move and after performing the move. We call this enlarged set the *leapfrog* set of moves.

If the previous move is allowed (i.e., if it is feasible for a given modular robot prototype), it seems natural to allow concatenating more than one of such moves, i.e., to allow concatenating consecutive rotations about vertices incident to the pivoting module. It is easy to prove that such concatenation cannot involve more than two pivots before the moving module becomes facet-adjacent to another module. Indeed, if a module $a$ is facet-adjacent to a module $s_1$, after at most two such moves it necessarily becomes adjacent to a module $s_2$ (Set 3 in Figure 2). We call this complete set the *monkey* set of moves.

### 2.2 Reconfiguration Problem

Consider a configuration $C$ of $n$ robot modules in a given grid. The *facet-adjacency graph* of $C$ has a node for each module, and an edge between a pair of nodes if the corresponding modules are facet-adjacent. Throughout this paper we will often refer to the facet-adjacency graph simply as the *adjacency graph*. We will say that a configuration $C$ is *facet-connected* if the facet-adjacency graph of $C$ is connected.

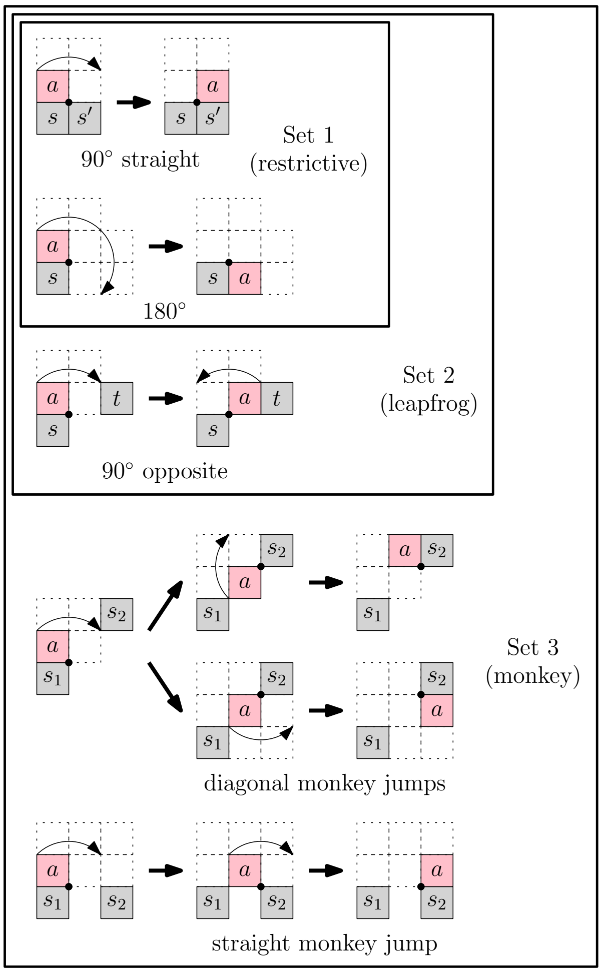

**Figure 2** The possible sets of moves for a pivoting module $a$ about a module $s$, in a square grid.

Applying a pivot move from one of the three sets of moves described in the previous section to a facet-connected configuration $C$, means applying one of the moves to a module in $C$, in such a way that the configuration (without the pivoting module) stays facet-connected before, after, and during the move, and the pivoting module does not collide with any other module. Note that this implies that even deleting the moving module the configuration is still facet-connected. Reconfiguring $C$ consists of applying a concatenation of such moves.

The (universal) reconfiguration problem asks whether it is possible to reconfigure any facet-connected configuration of $n$ modules in a given grid into any other configuration with the same number of modules.

For any positive integer $n$, the *reconfiguration graph* $\mathcal{G}_n$ has a node for each facet-connected configuration with $n$ modules, and an edge between two nodes if the corresponding configurations can be reconfigured into each other through a single pivoting move. We call *rigid* any configuration in which no module can move, i.e., any configuration that is an isolated node of $\mathcal{G}_n$, forming a connected component that is a singleton. We call *locked* any configuration that cannot be reconfigured into a straight strip of modules, i.e., any configuration belonging to a connected component of $\mathcal{G}_n$ that does not contain a strip.

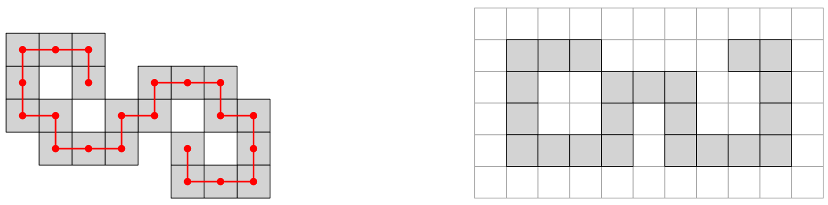

**Figure 3** Left: a rigid configuration of edge-connected pivoting squares. Right: A configuration that can be reconfigured into a strip, in spite of containing instances of the three forbidden patterns.

## 3 Reconfiguration Graph

Figure 3 (left) shows an example of a configuration that is rigid under the largest possible set of pivoting moves (set 3 in Figure 2). In [20] it is proved that reconfiguration for set 2 of pivoting moves (leapfrog moves) is possible between two facet-connected configurations of the same number of squares, provided that they are both *admissible* shapes. Admissibility is defined in terms of forbidden patterns: a facet-connected configuration of squares is admissible if it does not contain any of the patterns depicted in Figure 4. However, this local separation condition is certainly not necessary, as proves the example in Figure 3 (right).

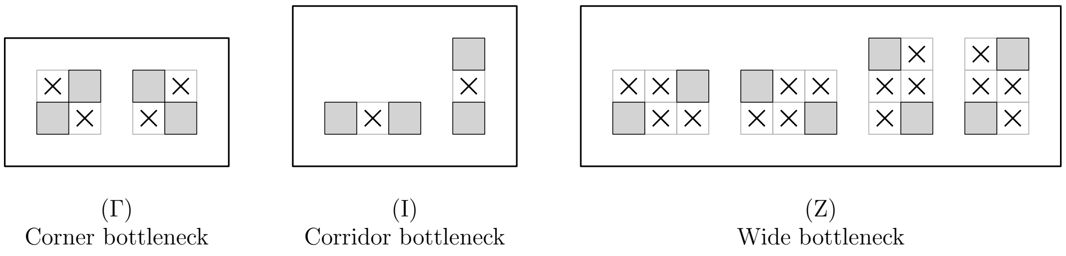

**Figure 4** The three forbidden patterns for facet-connected pivoting squares; solid squares represent modules, and ×-ed squares represent empty spaces.

These results raise several natural questions for facet-connected pivoting squares: Are the three sets of moves equivalent? In particular, is reconfigurability between admissible shapes also guaranteed when using the most restrictive set of pivoting moves? This latter question has been answered positively by the results from [20]. Although not explicitly stated, the reconfiguration algorithm from [20] uses only restrictive moves.

Several other interesting questions are open. Can the admissible condition be relaxed when using the largest set of pivoting moves? Do there exist rigid configurations that contain only one type of pattern? If so, are they rigid with respect to all three sets of pivoting moves? What can we say about the reconfiguration graph $\mathcal{G}_n$ for the different sets of pivoting moves? We try to answer these questions in the remaining of this section. Due to space constraints, the proofs of the propositions in this section are omitted.

We start by showing that the three sets of moves for pivoting squares are not equivalent, as they produce three different reconfiguration graphs.

▶ **Proposition 1.** *The monkey set of moves for pivoting squares (set 3) is stronger than the leapfrog set (set 2), and the leapfrog set is stronger than the restrictive set (set 1). That is, the resulting reconfiguration graph $\mathcal{G}_n$ has strictly fewer connected components for set 3 than for set 2, and fewer connected components for set 2 than for set 1.*

Let us now discuss the differences between the three forbidden patterns. From a purely geometric viewpoint, pattern Γ produces a (corner) bottleneck along the boundary of a configuration that is narrower than the one produced by pattern I (corridor bottleneck). This one is in turn narrower than the one produced by pattern Z (wide bottleneck). The next propositions show how the presence or the absence of each of such patterns influences reconfiguration under each of the 3 sets of pivoting moves.

### Pattern Γ : Corner Bottleneck

We start by showing that pattern Γ alone suffices to make a configuration rigid, regardless of the set of pivoting moves used (restrictive, leapfrog, or monkey).

▶ **Proposition 2.** *Let $\mathcal{G}_n$ be the reconfiguration graph of facet-connected pivoting squares. If only pattern Γ is allowed, while patterns I and Z are forbidden, the number of connected components of $\mathcal{G}_n$ that are singletons and the number of connected components of $\mathcal{G}_n$ of exponential size are both exponential, regardless of the set of pivoting moves used.*

### Pattern I : Corridor Bottleneck

The forbidden pattern I is weaker than pattern Γ in the sense that it suffices to make a configuration rigid for the sets of moves 1 and 2 (restrictive and leapfrog) but, if the entire set 3 of moves is allowed, pattern I alone cannot make a configuration rigid, as we will see.

▶ **Proposition 3.** *Let $\mathcal{G}_n$ be the reconfiguration graph of facet-connected pivoting squares under sets 1 and 2 of pivoting moves (restrictive and leapfrog). If only pattern I is allowed, and patterns Γ and Z are forbidden, the number of connected components of $\mathcal{G}_n$ that are singletons and the number of connected components of $\mathcal{G}_n$ of exponential size are both exponential.*

In contrast, if the entire set of monkey-pivoting moves is allowed, then no configuration can be rigid if it only contains instances of pattern I (and no instance of patterns Γ and Z).

▶ **Proposition 4.** *Let $\mathcal{G}_n$ be the reconfiguration graph of facet-connected pivoting squares under the entire set 3 of monkey-pivoting moves. If only pattern I is allowed, and patterns Γ and Z are forbidden, then $\mathcal{G}_n$ contains no singleton components.*

### Pattern Z : Wide Bottleneck

The forbidden pattern Z is weaker than the forbidden patterns Γ and I in the sense that no configuration can be rigid if it contains only instances of pattern Z.

▶ **Proposition 5.** *Let $\mathcal{G}_n$ be the reconfiguration graph of facet-adjacent pivoting squares. If only pattern Z is allowed, and patterns Γ and I are forbidden, then $\mathcal{G}_n$ contains no singleton components, regardless of the set of pivoting moves allowed.*

However, there can be locked configurations containing only instances of pattern Z.

▶ **Proposition 6.** *Let $\mathcal{G}_n$ be the reconfiguration graph of facet-connected pivoting squares under pivoting set of moves 1. If only pattern Z is allowed, and patterns Γ and I are forbidden, the number of connected components of $\mathcal{G}_n$ of exponential size is exponential.*

## 4 Universal Reconfiguration Algorithm with $O(1)$ Musketeers

In this section, we aim for the important practical goal of universal reconfiguration, that is, connectivity of the reconfiguration graph. We have seen that the local separation condition (while sufficient) is too strong: Robot configurations can contain many instances of the forbidden patterns and still be reconfigurable. On the other hand, we proved that as soon as the local separation condition is relaxed, the reconfiguration graph breaks into at least an exponential number of connected components of exponential size.

In what follows, we propose and analyze a new approach for reconfiguring arbitrary facet-connected configurations (which may contain an arbitrary number of instances of the forbidden patterns). Our strategy is based on the addition of $O(1)$ *musketeer modules*, i.e., modules that can freely move around the boundary of our robot configuration and will be used as helpers in certain situations. These modules are not necessarily part of the specified initial or target configuration.

### 4.1 Preliminaries: Outer Shell

Let $C$ be an arbitrary facet-connected configuration of pivoting squares. We start introducing a few definitions.

Let $G$ be the facet-adjacency graph of $C$, and $\overline{G}$ the facet-adjacency graph of the lattice cells that are not occupied by a module of $C$. Each bounded connected component of $\overline{G}$ is a *hole* of the robot configuration $C$. The only unbounded connected component of $\overline{G}$ is the *exterior* of $C$. The *boundary* of $C$ is the set of lattice cells that are empty and are facet-adjacent to (at least) one module of $C$. If the configuration has holes, we define its *external boundary* as the subset of the boundary contained in the unbounded connected component of $\overline{G}$.

▶ **Lemma 7.** *Let $C$ be an arbitrary and static facet-connected configuration of pivoting squares. Let $m$ be an active module attached to $C$, North of the topmost rightmost module of $C$. Using the monkey set of moves (set 3) $m$ can pivot along the external boundary of $C$ following the right-hand rule and return to its initial position. If only the leapfrog set of moves (set 2) is allowed, this is not always possible.*

The proof of this lemma is omitted due to space constraints.

It is worth noticing that the proof of Lemma 7 does not require the use of diagonal monkey jumps, but only of straight monkey jumps. This is relevant form a practical viewpoint, since it allows our results to be applied to a larger class of modular robots. For example, the hardware systems modeled in [3, 20] can performs straight monkey jumps, but not diagonal ones.

We can now define the *outer shell* of a facet-connected configuration $C$ of pivoting squares to be the subset of the external boundary of $C$ formed by the lattice cells eventually occupied by any active robot module $m$ initially positioned North of the topmost-rightmost module in $C$, in its right-hand rule traversal of the boundary of $C$, described in Lemma 7. Figure 5 illustrates this concept.

### 4.2 Algorithm Overview

Our reconfiguration algorithm transforms any initial facet-connected configuration $C$ of pivoting squares into any goal configuration with the same number of modules.

In order to simplify the algorithm's description, we use an intermediate canonical configuration, say a strip, and describe the transformation from the initial shape to the strip. The reconfiguration from the strip to the final shape is obtained by reversing the steps of the

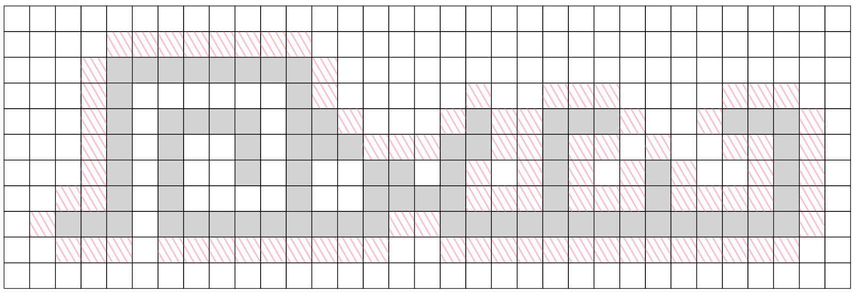

**Figure 5** A robot configuration (in gray) and its associated outer shell (striped in pink).

algorithm. The strip can be built from any lexicographically best positioned module of the configuration. For example, we will grow a horizontal strip to the left of the bottommost of the leftmost modules of the configuration.

The strategy behind the algorithm is simple. It consists of sequentially choosing a module from the configuration that is not a cut vertex of its facet-adjacency graph, and make it pivot, following the right-hand rule, along the outer shell, until it reaches the tip of the strip and stops. The problem of this strategy, as we saw in Section 3, is that the reconfiguration graph is not connected, even under the extended set of monkey moves. In order to overcome this problem, the algorithm uses *musketeer modules*. Any module from the canonical strip can serve as a musketeer module. We will prove that five musketeer modules are sufficient and sometimes necessary to solve any reconfiguration based on our strategy. Because the canonical strip is initially empty, it may be necessary to add musketeer modules to the strip if fewer than needed are available (this may happen at most once).

## 4.3 Algorithm Details

The description of the algorithm and the proof of its correctness make use of a *potential* function. If $m$ is a module located in the lattice position with coordinates $(x, y)$, the potential function at $m$ is defined as $\Phi(m) = (x+y, x)$. The potential being a two-dimensional function, we sort its values lexicographically. The maximum potential $\Phi_{max}$ (minimum potential $\Phi_{min}$) of a configuration is the lexicographically largest (smallest) potential of all its modules. Note that, whenever we use the term *configuration*, we refer to the facet-connected component that includes all modules other than the ones in the canonical strip.

Given any configuration, we define NE and SW as being the modules with highest and lowest potential, respectively. Notice that in any configuration $C$, both NE and SW are facet-adjacent to the outer shell of $C$.

### 4.3.1 Musketeer Modules

▶ **Definition 8.** *We say that a module $m$ is* outer-free *in a configuration $C$ if it is facet-adjacent to the outer shell and it can pivot clockwise, without disconnecting the robot.*

The first step of the algorithm is to look for an outer-free module, and pivot it to the tip of the strip. This step is repeated until no further outer-free modules exist in the initial configuration. If at that point the configuration is a strip, the algorithm ends.

Otherwise, all the modules in the strip may be used as musketeer modules, one at at time, starting from the tip of the strip, pivoting them to the positions where they are needed, as described in next Section 4.3.2. Since our algorithm may require five such modules, it

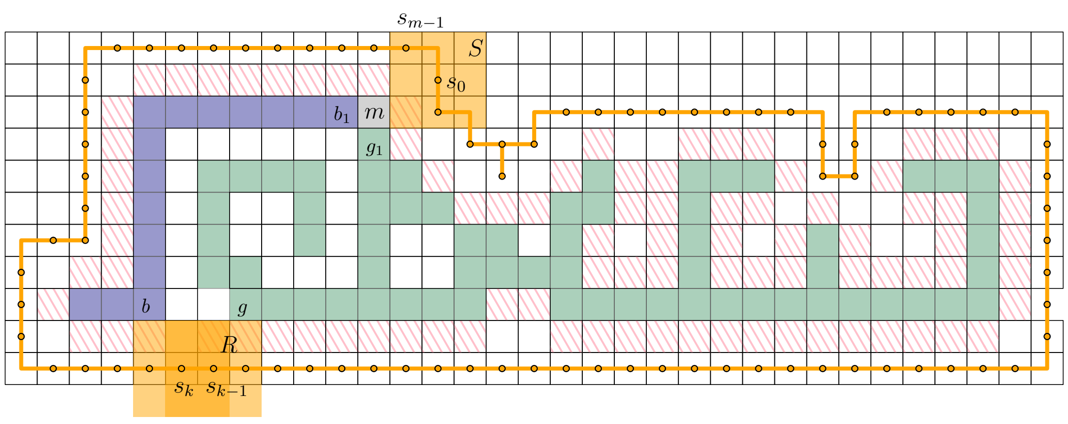

**Figure 6** Top: $3 \times 3$ square $S$ in its initial position $s_0$. The outer thick line indicates the path traversed by the center of $S$. Dots correspond to the center positions where $S$ is adjacent to a boundary edge. Bottom: the rectangular union $R$ of $S$ centered at $s_k$ and at $s_{k-1}$.

may be necessary to add extra modules to the strip (or anywhere in the configuration where they are outer-free) in order to complete the necessary set of musketeer modules. This can be done at this stage or on the fly, as needed. This second option may be preferable in some cases, as not all configurations require as many as five musketeer modules.

### 4.3.2 Bridging Procedure

In this section we describe an operation necessary in some situations when there are no outer-free modules in the configuration. Let $m$ be the NE module, i.e., the maximum potential module of a given configuration $C$. Trivially, there can be no modules of $C$ located North, North-East, or East of $m$, i.e., in positions $(0, 1)$, $(1, 1)$, or $(1, 0)$ relative to $m$. Therefore, the degree of $m$ in the facet-adjacency graph can only be 1 or 2. Since $m$ is not outer-free, it must be a cut vertex and have degree 2. Let $b_1$ and $g_1$ respectively be its counterclockwise and clockwise facet-adjacent modules (see Figure 6). We color the two connected components of $C$ connected by $m$ blue and green, so that $b_1$ is blue and $g_1$ is green. One important procedure of our algorithm, which we call *bridging*, is the act of using musketeer modules to connect the green and blue components so that $m$ becomes outer-free.

▶ **Observation 9.** *The outer shell has two green-blue changes of color, one happening at $m$.*

Consider a grid-aligned $3 \times 3$ square $S$ centered at the lattice cell of coordinates $s_0 = (2, 1)$, relative to $m$ (see top of Figure 6). Translate $S$ orthogonally clockwise one unit at a time along the boundary of the configuration until it reaches $s_0$ again. Ignoring the positions where $S$ is not adjacent to a boundary edge (i.e., the positions of $S$ where one of its corners coincides with a convex corner of $C$), let $s_i$ be the $i$-th position of the center of $S$ along its boundary traversal, and let $s_m = s_0$. Refer to Figure 6. Since $m$ is the maximum potential module of the configuration, $S$ is empty of modules at position $s_0$ and all subsequent positions, and it does not share edges with the blue component when centered at $s_0$, while at $s_{m-2}$ it is facet-adjacent to the blue module $b_1$. Let $s_k$ be the first position of $S$ along its boundary traversal where $S$ becomes facet-adjacent to a blue module. Since $S$ travels along the boundary of the configuration, the rectangular union $R$ of $S$ centered at $s_k$ and at $s_{k-1}$ should also be facet-adjacent to a green module (see bottom of Figure 6).

The algorithm pivots the musketeer modules clockwise, following the right-hand rule along the outer shell of the configuration, and brings them to the vicinity of $s_k$ to connect the blue and green components, thus forming a cycle containing $m$.

Let $g$ and $b$ be the closest pair of respectively green and blue modules facet-adjacent to rectangle $R$, and let $d$ be the $L_1$ distance between them. The bridging procedure depends on the value of $d$. It is easy to see that $d$ can only be 2, 3, 4, 5, or 6. In the full version of this paper we prove the following lemma:

▶ **Lemma 10.** *Let $m$ be the NE module, i.e., the maximum potential module of a given configuration $C$. The bridging procedure for $m$ uses $O(n)$ pivoting operations and at most five musketeer modules, and does not change the maximum and minimum potential of the configuration. After the procedure ends, $m$ is still the NE module of the modified configuration, but no longer a cut vertex of its facet-adjacency graph.*

Notice also that the bound of five musketeer modules for bridging is tight: Figure 7 shows an example requiring five musketeer modules for bridging.

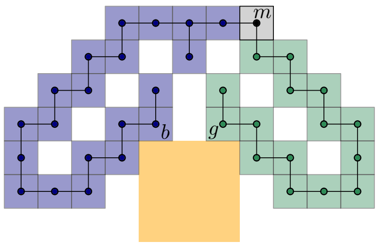

**Figure 7** A rigid configuration that requires the addition of five musketeer modules for bridging.

### 4.3.3 Reconfiguration Step

We now need to guarantee that module $m$ is able to move and thus it can pivot along the outer shell of $C$ and join the canonical strip. This is clear when $m$ is disjoint from the neighborhood of $m$. We also want to show that we can liberate and send to the canonical strip either the musketeers used or at least as many modules as musketeers used. In the full version of this paper we extend the analysis of the neighborhood of $m$, and for each of the possible cases we show that either invoking the bridging procedure or explicitly placing musketeer modules we can guarantee that.

Progress of the reconfiguration is measured in terms of the potential gap $\Delta\Phi = \Phi_{max} - \Phi_{min}$ of the configuration and the size of $C$ (recall that $C$ includes all modules that are not part of the canonical strip). In all the different cases we show that a reconfiguration step decreases the potential gap and/or the size of $C$.

## 4.4 Algorithm Pseudocode

Algorithm 1 solves the reconfiguration problem by combining the operations described in the previous sections:

▶ **Theorem 11.** *The reconfiguration algorithm (Algorithm 1) transforms a facet-connected configuration $C$ with $n$ modules into a canonical strip of the same size, using $O(n^2)$ monkey-move pivoting steps, which is worst-case optimal, and adding at most five extra modules.*

**Proof.** The input to the algorithm is a configuration $C$ of size $n$ and potential gap $\Delta\Phi = O(n)$. Each step of the innermost loop uses $O(n)$ pivoting operations to take an outer-free module to the end of the strip, thus decreasing the size of $C$ by one. Each reconfiguration step uses $O(n)$ pivoting steps to decrease either the potential gap or the size of $C$, leaving it facet-connected

**Algorithm 1** Reconfiguring an arbitrary facet-connected configuration into a canonical strip.

```
Data: An arbitrary facet-connected configuration $C$ with $n$ modules
Result: A canonical strip of modules of length $n$
while there are still modules in $C$ do
    while there exist outer-free modules do
        pick one outer-free module and pivot it all the way to the tip of the strip;
    end
    if the strip has fewer than five modules then
        make the strip five modules long by adding musketeer modules;
    end
    invoke the reconfiguration step;
end
```

(and never increasing the potential gap). Because the size of $C$ never increases, the length of the canonical strip never decreases. This means that the strip can have fewer than five modules only once and the conditional does not affect the complexity of the algorithm. We conclude that the algorithm terminates after $O(n)$ iterations in total. Because each iteration takes $O(n)$ pivoting steps, the total number of pivoting steps is $O(n^2)$. Optimality comes from the $\Omega(n^2)$ pivoting steps required to reconfigure a vertical strip into a horizontal one. ◀

## 5 Conclusion and Open Problems

This paper addresses the problem of reconfiguring a facet-connected grid configuration of $n$ modules into any other configuration of $n$ modules under three increasingly more flexible sets of pivoting moves, namely restrictive, leapfrog and monkey. Previous results solve this problem under the leapfrog set of moves, as long as the initial and final configurations satisfy a strong local separating condition imposed by three forbidden patterns. We show that there exist robot configurations with many instances of the three forbidden patterns that are still reconfigurable, so the local separation condition is not necessary. On the other hand, we show that as soon as the local separation condition is relaxed, the reconfiguration graph breaks into an exponential number of connected components of exponential size. To overcome this obstacle we introduce a new pivoting move, called monkey, and a natural reconfiguration approach that does not depend on local features, but uses up to five extra modules that can freely move around the boundary of the robot configuration. These extra modules are used to unlock intermediate locked configurations so that progress can be made towards the target configuration. We show that our approach uses $O(n^2)$ monkey-pivoting moves to reconfigure any source configuration with $n$ pivoting modules into any given target configuration.

We leave open the question of whether universal reconfiguration can be accomplished under the more restrictive set of leapfrog pivoting moves using a constant number of extra modules.

Another question is whether our approach generalizes to three or higher dimensions. For example, when the slice graphs (where the vertices are the *slices* of the configuration cut along an axis and the edges connect slices with facet-adjacent modules) of the source and target configurations are both paths, we should be able to reconfigure each to a strip of modules, one slice at a time, similar to our 2-dimensional approach does. We conjecture that a similar approach will also work for general 3-dimensional configurations, potentially after increasing the number of musketeer modules to bridge larger gaps introduced by the higher dimensionality.

## References

1 Z. Abel and S. D. Kominers. Pushing hypercubes around. *CoRR*, abs/0802.3414, 2008. `arXiv:0802.3414`.

2 B. K. An. EM-Cube: cube-shaped, self-reconfigurable robots sliding on structure surfaces. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3149–3155, 2008.

3 N. Ayanian, P. J. White, Á. Hálász, M. Yim, and V. Kumar. Stochastic control for self-assembly of XBots. In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2008.

4 N. M. Benbernou. *Geometric algorithms for reconfigurable structures*. PhD thesis, Massachusetts Institute of Technology, 2011.

5 S. Chennareddy, A. Agrawal, and A. Karuppiah. Modular self-reconfigurable robotic systems: a survey on hardware architectures. *Journal of Robotics*, 2017(5013532), 2017.

6 G. S. Chirikjian. Kinematics of a metamorphic robotic system. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 449–455, 1994.

7 A. Dumitrescu and J. Pach. Pushing squares around. *Graphs and Combinatorics*, 22(1):37–50, 2006.

8 R. Fitch, Z. Butler, and D. Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2460–2467, 2003.

9 A. Hemmerling. *Labyrinth problems – labyrinth-searching abilities of automata*, volume 14 of *Teubner-Texte zur Mathematik (TTZM)*. Springer-Verlag, 1989.

10 H. Kurokawa, S. Murata, E. Yoshida, K. Tomita, and S. Kokaji. A 3-D self-reconfigurable structure and experiments. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 860–865, 1998.

11 T. Larkworthy and S. Ramamoorthy. A characterization of the reconfiguration space of self-reconfiguring robotic systems. *Robotica*, 29(1):73–85, 2011.

12 O. Michail, G. Skretas, and P. G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. *J. Comput. Syst. Sci.*, 102:18–39, 2019.

13 S. Murata, H. Kurokawa, and S. Kokaji. Self-assembling machine. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 441–448, 1994.

14 S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-TRAN: self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4):431–441, 2002.

15 A. Nguyen, L. J. Guibas, and M. Yim. Controlled module density helps reconfiguration planning. In *Algorithmic and Computational Robotics: New Dimensions (WAFR)*, pages 23–25. A. K. Peters, 2001.

16 E. H. Østergaard, K. Kassow, R. Beck, and H. H. Lund. Design of the ATRON lattice-based self-reconfigurable robot. *Autonomous Robots*, 21(2):165–183, 2006.

17 D. Rus and M. Vona. A physical implementation of the self-reconfiguring crystalline robot. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1726–1733, 2000.

18 B. Salemi, M. Moll, and W.-M. Shen. SUPERBOT: a deployable, multi-functional, and modular self-reconfigurable robotic system. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3636–3641, 2006.

19 K. Stoy, D. Brandt, and D. J. Christensen. *Self-reconfigurable robots: an introduction*. MIT Press, 2010.

20 C. Sung, J. Bern, J. Romanishin, and D. Rus. Reconfiguration planning for pivoting cube modular robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1933–1940, 2015.

21 C. Unsal, H. Kiliccote, and P. Khosla. I(CES)-Cubes: a modular self-reconfigurable bipartite robotic system. In *Proc. SPIE Conference on Mobile Robots and Autonomous Systems*, volume 3839, pages 258–269. SPIE, 1999.

22 M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

23 V. Zykov, A. Chan, and H. Lipson. Molecubes: an open-source modular robotic kit. In *IROS-2007 Self-Reconfigurable Robotics Workshop*, 2007.