Asymptotically Optimal Kinodynamic Motion Planning for Self-reconfigurable Robots

John H. Reif[1] and Sam Slee[1]

Department of Computer Science, Duke University, Durham, NC, USA
{reif,sgs}@cs.duke.edu

Abstract. Self-reconfigurable robots are composed of many individual modules that can autonomously move to transform the shape and structure of the robot. In this paper we present a kinodynamically optimal algorithm for the following "x-axis to y-axis" reconfiguration problem: given a horizontal row of n modules, reconfigure that collection into a vertical column of n modules. The goal is to determine the sequence of movements of the modules that minimizes the movement time needed to achieve the desired reconfiguration of the modules. Prior work on self-reconfigurable (SR) robots assumed a constant velocity bound on module movement and so required time linear in n to solve this problem.

In this paper we define an abstract model that assumes unit bounds on various physical properties of modules such as shape, aspect ratio, mass, and the maximum magnitude of force that an individual module can exert. We also define concrete instances of our abstract model similar to those found in the prior literature on reconfigurable robots, including various examples where the modules are cubes that are attached and can apply forces to neighboring cubes. In one of these concrete models, the cube's sides can contract and expand with controllable force, and in another the cubes can apply rotational torque to their neighbors. Our main result is a proof of tight $\Theta(\sqrt{n})$ upper and lower bounds on the movement time for the above reconguration problem for concrete instances of our abstract model.

This paper's analysis characterizes optimal reconfiguration movements in terms of basic laws of physics relating force, mass, acceleration, distance traveled, and movement time. A key property resulting from this is that through the simultaneous application of constant-bounded forces by a system of modules, certain modules in the system can achieve velocities exceeding any constant bounds. This delays modules with the least distance to travel when reconfiguring in order to accelerate modules that have the farthest to travel. We utilize this tradeoff in our algorithm for the x-axis to y-axis problem to achieve an $\mathrm{O}(\sqrt{n})$ movement time.

1 Introduction

This paper develops efficient algorithms by treating reconfiguration as a kinodynamic planning problem. Kinodynamic planning refers to motion planning problems subject to simultaneous kinematic and dynamics constraints [2]. To that end, in this paper we define an abstract model for modules in SR robots.

S. Akella et al. (Eds.): Algorithmic Foundation of Robotics VII, STAR 47, pp. 457–472, 2008.
springerlink.com

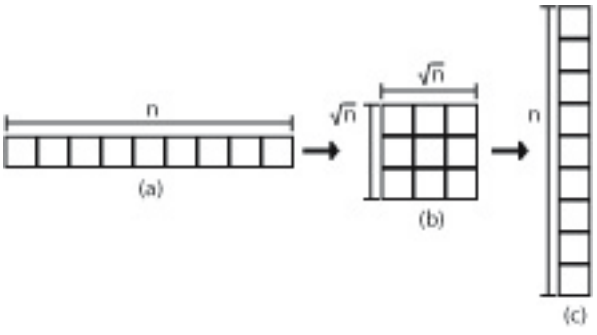

Fig. 1. The x-axis to y-axis problem: transforming a row of modules into a column. Our algorithm uses an intermediate step of forming a square.

This extends the definitions of previous models [3, 4] by setting fixed unit bounds on a module's shape, aspect ratio, mass, and the force it can apply, among other requirements.

To exhibit the significance of these bounds we consider the following "x-axis to y-axis" reconfiguration problem. Given a horizontal row of n modules, reconfigure that collection into a vertical column of n modules. This simple problem, illustrated in Figure 1, provides a worst-case example in that $\Omega(n)$ modules must move $\Omega(n)$ module lengths to reach any position in the goal configuration regardless of that goal column's horizontal placement. Any horizontal positioning of the vertical column along the initial row configuration is deemed acceptable in our treatment of the problem in this paper.

We define the *movement time* of a reconfiguration problem to be the time taken for a system of n modules to reconfigure from an initial configuration to a desired goal configuration. Modules are assumed to be interchangeable so the exact placement of a given module in the initial or goal configuration is irrelevant. An implicit assumption in various prior papers on reconfiguring robotic motion planning [3, 5, 6] is that the modules are permitted only a fixed unit velocity. This assumption is not essential to the physics of these systems and has constrained prior work in the area.

For the x-axis to y-axis problem, if only one module in the SR robot is permitted to move at a given time and with only a fixed unit velocity, a lower bound of $\Omega(n^2)$-time is clear [5]. Allowing concurrent movement of modules, a movement time of $O(n)$ is possible while still keeping all modules connected to the system and keeping unit velocities. However, we observe that faster reconfiguration is possible if we do not restrict modules to move at a fixed, uniform pace.

Another major principle that we use, and that has seen use in prior reconfiguration algorithms [1, 7], is what we refer to as the principle of *time reversal*. This principle is simply that executing reconfiguration movements in reverse is always possible, and they take precisely the same movement time as in the forward direction. This is, of course, ignoring concerns such as gravity. Otherwise

an example of rolling a ball down a hill would require less time or less force than moving that ball back up the hill. We ignore gravity and use this principle extensively in work of this paper.

Before continuing further, it will be useful to define some notation that we will use throughout the remainder of this paper. As given above, let n denote the number of modules in the SR robot undergoing reconfiguration. In the initial row configuration of the x-axis to y-axis problem, let the modules be numbered $1, \ldots, n$ from left to right. Let $x_i(t)$ be the x-axis location of module i at time t. Similarly, let $v_i(t)$ and $a_i(t)$ be the velocity and acceleration, respectively, of module i at time t. For simplicity, the analysis in our examples will ignore aspects such as friction or gravity. The effect is similar to having the reconfiguration take place with ideal modules while lying flat on a frictionless planar surface.

In the following Section 2, we differentiate between different styles of self-reconfigurable robots and survey related work in the field. We begin introducing the work of this paper in Section 3 by giving our abstract model for SR robot modules. Given the bounds set by this model, Section 4 references physics equations that govern the movement of modules and define what reconfiguration performance is possible. To explain our algorithm, in Section 4 we also begin with a 1-dimensional case of n masses represented as a row of n points with a separation of 1 unit between adjacent points, for some unit of distance measure. The points will then contract so that they have only 1/2 unit separation. After showing that this contraction takes $O(\sqrt{n})$ movement time while still satisfying the bounds of our abstract model, we then immediately extend this result to a matching example using a known physical architecture for modules.

Section 5 will then extend the result to a contraction/expansion case in 2 dimensions while maintaining the same time bound. This will then lead to a $O(\sqrt{n})$ movement time algorithm for the x-axis to y-axis reconfiguration problem. This algorithm recursively uses the 1-dimensional contraction operation and uses the reversible process of transforming the initial n module row into an intermediate stage $\sqrt{n} \times \sqrt{n}$ cube. The process is then reversed to go from the cube to the goal column configuration. Section 6 then matches this with a $\Omega(\sqrt{n})$ lower bound for the 1-dimensional example and the x-axis to y-axis problem, showing both cases to be $\Theta(\sqrt{n})$. Finally, the conclusion in Section 7 summarizes the results of this paper. Some proofs are omitted due to length requirements. A version of this paper including all proofs can be found at: `www.cs.duke.edu/~sgs/publications/reifsleeWAFR06.pdf` .

2 Related Work

When developing models and algorithmic bounds for self-reconfigurable (SR) robots (also known as metamorphic robots [8, 4, 5]) it is important to note the style of SR robot we are dealing with. Two of the main types of SR robots are closed-chain style robots and lattice style robots. Closed-chain SR robots are composed of open or closed kinematic chains of modules. Their topology is described by one-dimensional combinatorial topology. [1] To reconfigure these

modules are required to swing chains of other modules to new locations. One such implementation of this design by Yim et al. has had several demonstrations of locomotion [9].

For the other major style, lattice or substrate SR robots attach together only at discrete locations to form lattice-like structures. Individual modules typically move by walking along the surfaces formed by other modules. The hardware requirements for this style of module are more relaxed than those for closed-chain style systems. Here individual modules need only be strong enough to move themselves or one or two neighbors. Closed-chain style modules typically must be strong enough to swing long chains of other modules. The models and algorithms presented in this paper are meant for lattice style modules.

Previous work by several research groups has developed abstract models for lattice style SR robots. One of the most recent is the sliding cube model proposed by Rus et al. [3]. As the name implies, this model represents modules as identical cubes that can slide along the flat surfaces created by lattices of other modules. In addition, modules have the ability to make convex or concave transitions to other orthogonal surfaces. In this abstraction, a single step action for a module would be to detach from its current location and then either transition to a neighboring location on the lattice surface, or make a convex or concave transition to another orthogonal surface to which it is next. Transitions are only made in the cardinal directions (no diagonal movements) and for a module to transition to a neighboring location that location must first be unoccupied. Most architectures for lattice style SR robots satisfy the requirements of this model.

One such physical implementation is the compressible unit or expanding cube design [6, 7]. Here an individual module can expand from its original size to double its length in any given dimension, or alternatively compress to half its original length. Neighbor modules are then pushed or pulled by this action to generate movement and allow reconfiguration. This particular module design is relevant because it provides a good visual aide for the algorithms we present in this paper. For this purpose we also add the ability for expanding cubes to slide relative to each other. Otherwise a single row of these modules can only exert an expanding or contracting force along the length of that row and is unable to reconfigure in 2 dimensions. Prior work has noted that while individual expanding cube modules do not have this ability, it can be approximated when groups of modules are treated as atomic units [6, 7].

3 Abstract Model

We now begin introducing our results for this paper by defining our abstract model for SR robot modules. It generalizes many of the requirements and properties of reconfigurable robots and in particular, the properties of the sliding cube model and the expanding cube hardware design described in the previous section. Our abstract model explicitly states bounds on physical properties such as the size, mass, and force exerted for each module. These properties will be

utilized by the algorithms and complexity bounds that later follow. The requirements of our model are described by the following set of axioms.

- Each module is assumed to be an object in 3D with either (i) fixed shape and size or (ii) a limited set of geometric shapes that it can acquire. In either case, the module also has a constant bound on its total volume and the aspect ratio of its shape.
- Each module can latch onto or grip adjacent modules and apply to these neighboring modules a force or torque.
- Generally, modules are all connected either directly or indirectly at any time.
- For each module there is a constant bound on its mass.
- There is a constant bound on the magnitude of the force and/or torque that a module may apply to those modules with which it is in contact.
- The motion of modules is such that they never collide with a velocity above a fixed constant magnitude.
- For modules in direct contact with each other, the magnitude of the difference in velocity between these contacting modules is always bounded by a constant.
- Each module, when attached (or latched) to other modules, can dynamically set the stiffness of the attachment. This in addition to the ability of the module to apply contraction/expansion or rotational forces at its specified attachments.

Note: The axiom on the stiffness of the attachments to neighboring modules is required to ensure that forces (external to the module) are transmitted through it to neighboring modules. We add this since contraction and/or rotational forces along a chain of modules can accumulate to amounts more than the unit maximum applied by any one module, and may need to be transfered from neighbor to neighbor. Note also that we assume idealized modules that act in synchronized movements under centralized control. This reduces analysis difficulties for cases when many sets of modules operate in parallel.

While the axioms given above state important module requirements, more concrete models are necessary for algorithm design and analysis. The sliding cube model and the expanding cube hardware design described in the previous section satisfy the axioms of our abstract model so long as bounds on the physical abilities of modules are implied. In particular, for these modules the "stiffness" requirement means that the transmitted forces are translational. The exact use of this will become apparent in our first 1-dimensional example with expanding cube modules.

Finally, while not used in this paper, we note two other useful abilities for physical modules: the ability to *push* and the ability to *tunnel*. The ability to push requires one module exerting enough force to push a second module in front of it while sliding along a surface in a straight line. The tunneling ability would allow modules to transition through the interior of a lattice structure in addition to moving along external surfaces.

4 1D Force Analysis

All analysis of movement planning in this paper is based on the physical limitations of individual modules stated in the last section. Given this abstract model, we can now make use of the elementary equations of Newtonian physics governing the modules' movement in space and time. For these equations we use the notation first mention in Section 1. Let $x_i(t)$ be the distance traveled by object i during movement time t. Similarly, $v_i(t)$ is its velocity after time t. Finally, given an object i with mass m_i, let F_i be the net force applied to that object and a_i be the resulting constant acceleration. Although the relevant equations are basic, we state them here so that they can be referred to again later in the paper:

$$F_i = m_i a_i \tag{1}$$

$$x_i(t) = x_i(0) + v_i(0)t + \frac{1}{2}a_i t^2 \tag{2}$$

$$v_i(t) = v_i(0) + a_i t \ . \tag{3}$$

In the first equation we get that a net force of F_i is required to move an object with mass m_i at a constant acceleration with magnitude a_i. In the second equation, an object's location $x_i(t)$ after traveling for a time t is given by it's initial position $x_i(0)$, it's initial velocity $v_i(0)$ multiplied by the time, and a function of a constant acceleration a_i and the time traveled squared. Similarly, the third equation gives the velocity after time t, $v_i(t)$, to be the initial velocity $v_i(0)$ plus the constant acceleration a_i multiplied by the time traveled.

All modules in the examples that follow are assumed to have unit mass $m = 1$ and sides with unit length 1. Our algorithms all require 2 stages of motion: one stage to begin motion and a second stage to slow it and ensure zero final velocity. So, we assume that each module is capable of producing a unit amount of force 1 once for each stage. This still keeps within the constant-bounded force requirement of our abstract model. For an expanding cube module this unit force is capable of contracting or expanding it in unit time while pulling or pushing one neighbor module. Also, since the module has unit mass, by equation (1) we get a bound on its acceleration $a \leq 1$ given the force applied in just a single motion stage. Again, all concerns for friction or gravity (or a detailed description of physical materials to ensure the proper stiffness of modules) are ignored to simplify calculations.

Before tackling the x-axis to y-axis reconfiguration problem, we first begin by analyzing a simpler 1-dimensional case which we will refer to as the *Point Mass Contraction* problem. Here a row of n point masses will contract from a total length of n units, to a length of $n/2$. Although these point masses are not connected and do not grip each other, they otherwise satisfy the axioms of our model. After showing this reconfiguration to take $O(\sqrt{n})$ movement time, we will show that same result holds for a case with expanding-cube style modules instead of point masses. This 1-dimensional contraction case, which we refer to as the *Squeeze* problem, will be a recursive step in our algorithm for the x-axis to y-axis reconfiguration problem.

We assume the initial configuration consists of an even number n of point masses arranged in a row on the x-axis, each initially having 0 velocity. For $i = 1, \ldots, n$ the ith point initially at time $t = 0$ has x-coordinate $x_i(0) = i - (n+1)/2$. We assume each point has unit mass and can move in the x-direction with acceleration magnitude upper bounded by 1. For simplicity, we assume no friction nor any gravitational forces. Our goal configuration at the final time T is the point masses arranged in a row on the x-axis with 0 final velocity, each distance $1/2$ from the next in the x-axis direction, so that for $i = 1, \ldots, n$ the ith point at the final time T has x-coordinate

$$x_i(T) = \frac{x_i(0)}{2} = \frac{i}{2} - \frac{n+1}{4} .$$

To differentiate notation, we'll use parentheses to denote a function of time, such as $x_i(t)$, and square brackets to denote order of operations, such as $2 * [3 - 1] = 4$. Furthermore, we require that the velocity difference between consecutive points is at most 1, and that consecutive points never get closer than a distance of $1/2$ from each other. Given this, our goal is to find the minimal possible movement time duration from the initial configuration at time 0 to final configuration at time T.

Lemma 1. *The Point Mass Contraction problem requires at most total movement time* $T = \sqrt{n-1}$.

Proof: Fix $T = \sqrt{n-1}$. For each point mass $i = 1, \ldots, n$ at time t, for $0 \leq t \leq T$, let $x_i(t)$ be the x-coordinate of the ith point mass at time t and let $v_i(t), a_i(t)$ be its velocity and acceleration, respectively, in the x-direction (our algorithm for this Point Mass Contraction Problem will provide velocity and acceleration only in the x-direction).

For $i = 1, \ldots, n$ the ith point mass needs to move from initial x-coordinate $x_i(0) = i - (n+1)/2$ starting with initial velocity $v_i(0) = 0$ to final x-coordinate $x_i(T) = x_i(0)/2 = i/2 - (n+1)/4$ ending with final velocity $v_i(T) = 0$. To ensure the velocity at the final time is 0, during the first half of the movement time the ith point mass will be accelerated by an amount α_i in the intended direction, and then during the second half of the movement time the ith point mass will accelerate by $-\alpha_i$ (in the reverse of the intended direction). For $i = 1, \ldots, n$, set that acceleration as:

$$\alpha_i = \frac{n + 1 - 2i}{n - 1} .$$

Note that the maximum acceleration bound is unit, since $|\alpha_i| \leq 1$, satisfying the requirement of our abstract model. During the first half of the movement time t, for $0 \leq t \leq T/2$, by equations (3) and (2) we get that the velocity at time t is $v_i(t) = v_i(0) + \alpha_i t$ which implies $v_i(t) = \alpha_i t$ and the x-coordinate is given by:

$$x_i(t) = x_i(0) + v_i(0)t + \frac{\alpha_i}{2} t^2$$
$$x_i(t) = i - \frac{n+1}{2} + \frac{\alpha_i}{2} t^2 .$$

At the midway point $T/2$, this gives the ith point mass velocity $v_i(T/2) = \alpha_i T/2$ and x-coordinate

$$x_i(T/2) = i - \frac{n+1}{2} + \frac{\alpha_i}{2}\left[\frac{T}{2}\right]^2$$
$$x_i(T/2) = i - \frac{n+1}{2} + \frac{a_i T^2}{8} .$$

During the second half of the movement time t, for $T/2 < t \leq T$, we will set the ith point mass acceleration at time t to be $a_i(t) = -\alpha_i$. By equation (3) we get that the velocity at time t during the second half of the movement time is

$$v_i(t) = v_i\,(T/2) - \alpha_i\left[t - \frac{T}{2}\right]$$
$$v_i(t) = \alpha_i\left[\frac{T}{2}\right] - \alpha_i\left[t - \frac{T}{2}\right]$$
$$v_i(t) = \alpha_i[T - t] .$$

and the x-coordinate given by equation (2) is

$$x_i(t) = x_i(T/2) + v_i(T/2)\left[t - \frac{T}{2}\right] - \frac{\alpha_i}{2}\left[t - \frac{T}{2}\right]^2$$
$$x_i(t) = \left[i - \frac{n+1}{2} + \frac{\alpha_i T^2}{8}\right] + \frac{\alpha_i T}{2}\left[t - \frac{T}{2}\right] - \frac{\alpha_i}{2}\left[t - \frac{T}{2}\right]^2 .$$

This implies that at the final time T, the ith point mass acceleration has velocity $v_i(T) = \alpha_i(T - T) = 0$ and x-coordinate

$$x_i(T) = \left[i - \frac{n+1}{2} + \frac{\alpha_i T^2}{8}\right] + \frac{\alpha_i T}{2}\left[T - \frac{T}{2}\right] - \frac{\alpha_i}{2}\left[T - \frac{T}{2}\right]^2$$
$$x_i(T) = \left[i - \frac{n+1}{2} + \frac{\alpha_i T^2}{8}\right] + \frac{\alpha_i T^2}{4} - \frac{\alpha_i T^2}{8}$$
$$x_i(T) = i - \frac{n+1}{2} + \frac{\alpha_i T^2}{4} .$$

Recalling that we initially set $\alpha_i = \frac{n+1-2i}{n-1}$ and $T = \sqrt{n-1}$ we get:

$$x_i(T) = i - \frac{n+1}{2} + \frac{n+1-2i}{4}$$
$$x_i(T) = \frac{i}{2} - \frac{n+1}{4}$$
$$x_i(T) = \frac{x_i(0)}{2} .$$

So we get that $x_i(T) = x_i(0)/2$ as required. Moreover, the time any consecutive points are closest is the final time T, and at that time they are distance $1/2$ from each other, as required in the specification of the problem.

It is easy to verify that the velocity difference between consecutive points masses i and $i+1$ is maximized at time $T/2$, and at that time the the magnitude of the velocity difference is

$$\begin{aligned} |v_i(T/2) - v_{i+1}(T/2)| &= |\alpha_i - \alpha_{i+1}|T/2 \\ &\leq \frac{2}{n-1}\frac{\sqrt{n-1}}{2} \\ &\leq 1 \end{aligned}$$

as required in the specification of the problem. Thus, we have shown it possible to complete this problem in time $T = \sqrt{n-1}$ while still satisfying the axioms of our abstract model (excluding connectivity and gripping). □

The *Squeeze* Problem. With this result proved for the Point Mass Contraction problem, the same reconfiguration using expanding cube modules can be solved by directly applying Lemma 1. We refer to this reconfiguration task as the *Squeeze* problem and define it as follows. Assume an even number of n modules, numbered $i = 1, \ldots, n$ from left to right as before. Keeping the modules connected as a single row, our goal is to contract the modules from each having length 1 to each having length $1/2$ for some unit length in the x-axis direction. This reconfiguration task is shown in Figure 2. Given the bounds on the module's physical properties required by our abstract model, the goal is to perform this reconfiguration in the minimum possible movement time T.

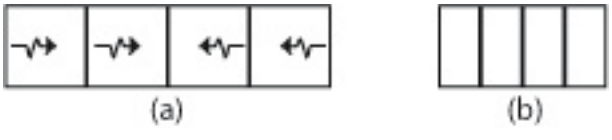

Fig. 2. The *Squeeze* problem: Expanding cube modules in (a), each of length 1, contract to each have length $1/2$ and form the configuration in (b)

Similarly, we define the *Reverse Squeeze* problem as the operation that undoes the first reconfiguration. Given a connected row of n contracted modules, each with length $1/2$ in the x-axis direction, expand those modules to each have length 1 while keeping the system connected as a single row. Since this problem is exactly the reverse of the original *Squeeze* problem, steps for an algorithm solving the *Squeeze* problem can be run in reverse order to solve the Reverse *Squeeze* problem in the same movement time with the same amount of force.

We first perform the force analysis for the case of solving the *Squeeze* problem. At the initial time $t = 0$ cube i's center x-coordinate has location

$x_i(0) = i - (n+1)/2$. Recall that each cube is assumed to have unit mass 1, can grip its adjoining cubes, and can exert expanding or contracting forces against those neighbors. Furthermore, a unit upper bound is assumed on the magnitude of this force, which as stated earlier also causes an acceleration bound $a_i \leq 1$ for all modules. We wish to contract the center x-coordinates of the modules from location $x_i(0)$ to $x_i(0)/2 = i/2 - [n+1]/4$ as before in the contraction example with point masses. Again, we wish to find the minimum possible movement time for contraction from the initial configuration at time 0 to the goal configuration at time T.

Lemma 2. *The Squeeze problem requires at most total movement time* $T = \sqrt{n-1}$.

The proof is omitted here to meet space requirements.

Given that the above problem requires at most $T = \sqrt{n-1}$ time to be solved, we get the same result for the *Reverse Squeeze* problem.

Corollary 1. *The Reverse Squeeze problem requires at most movement time* $T = \sqrt{n-1}$.

Note that all of the operations performed in solving the *Squeeze* problem can be done in reverse. The forces and resulting accelerations used to contract modules can be performed in reverse to expand modules that were previously contracted. Thus, we can reverse the above *Squeeze* algorithm in order to solve the *Reverse Squeeze* problem in exactly the same movement time as the original *Squeeze* problem. (Note: Although the forces are reversed in the Reverse Squeeze problem, the stiffness settings remain the same. It is important to observe that without these stiff attachments between neighboring modules, the accumulated expansion forces would make the entire assembly fly apart.)

5 2D Reconfiguration

The above analysis of the 1-dimensional *Squeeze* problem has laid the groundwork for reconfiguration in 2 dimensions. This includes the x-axis to y-axis reconfiguration problem which we will provide an algorithm for at the end of this section. We build to that result by first looking at a simpler example of 2-dimensional reconfiguration that will serve as an intermediate step in our final algorithm. We continue to use the same expanding cube model here as was used in the previous section.

In that previous section a connected row of an even number of n expanding cube modules was contracted from each module having length 1 in the x-axis direction to each having length $1/2$. We now begin with that contracted row and reconfigure it into two stacked rows of $n/2$ contracted modules. Modules begin with $\frac{1}{2} \times 1$ width by height dimensions and then finish with $1 \times \frac{1}{2}$ dimensions. This allows pairs of adjacent modules to rotate positions within a bounded 1×1 unit dimension square.We denote this reconfiguration task as the *confined cubes swapping* problem.

The process that we use to achieve reconfiguration is shown in Figure 3. We begin with a row of modules, each with dimension $\frac{1}{2} \times 1$. Here motion occurs in two parts. First, fix the bottom edge of odd numbered modules so that edge does not move. Do the same for the top edge of even numbered modules. Then contract all modules in the y direction from length 1 to length 1/2. Note that to achieve this two counterbalancing forces are required: (1) a force within each module to contract it, and (2) sliding forces between adjacent modules in the row to keep the required top/bottom edges in fixed locations as described earlier. This process creates the "checkered" configuration in part (c) of Figure 3.

We can then reverse the process, but execute it in the x direction instead, to expand the modules in the x direction and create 2 stacked rows of modules, each with dimension $1 \times \frac{1}{2}$. Note that requiring certain edges to stay in fixed locations had an important byproduct. This results in pairs of adjacent modules moving within the same 1×1 square at all times during reconfiguration. This also means that the bounding box of the entire row does not change as it reconfigures into 2 rows. This trait will be of great significance when this reconfiguration is executed in parallel on an initial configuration of several stacked rows.

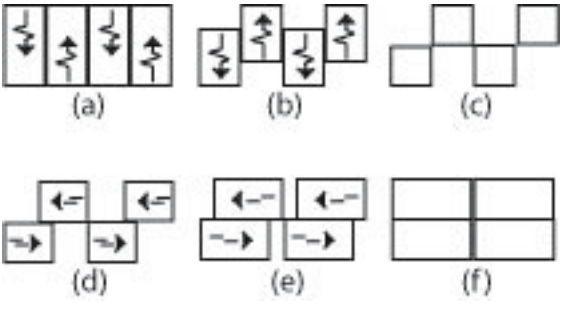

Fig. 3. Expanding cube modules contracting vertically (a - c), then expanding horizontally (d - f). Here the scrunched arrow represents contraction and the 3 piece arrow denotes expansion.

Again, number modules $i = 1, \ldots, n$ from left to right and assume modules have unit mass 1, can grip each other, and can apply contraction, expansion and sliding forces. In this problem we will only use a unit force 1 total per module for both stages of motion. This means a force of 1/2 applied in each motion stage and, by the physics equations in Section 4, an acceleration upper bounded by 1/2 in each stage as well. Finally, concerns for gravity or friction are again ignored for the sake of simplicity. Given these bounds, our goal is to find the minimum reconfiguration time for this confined cubes swapping problem.

Lemma 3. *The confined cubes swapping problem described above requires $T = O(1)$ movement time for reconfiguration.*

The proof is omitted here to meet space requirements.

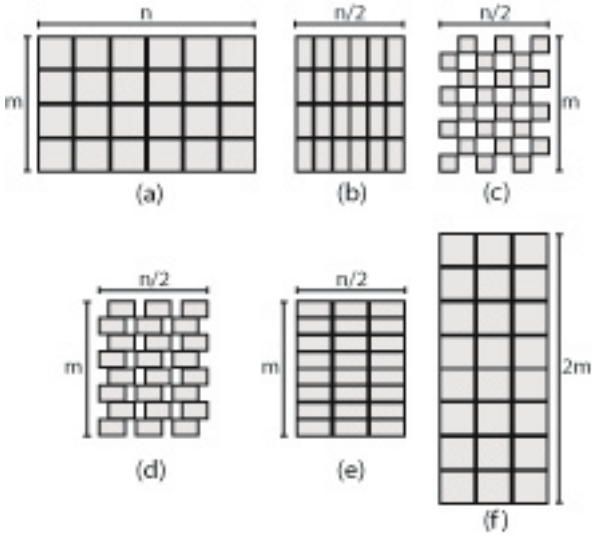

Fig. 4. Horizontal contraction from stage a to b. Vertical expansion from state e to f. Note that the dimensions of the array remain unchanged through stages b - e.

Extending this analysis, we now consider the case of an $m \times n$ array of normal, unit-dimension modules. That is, we have m rows with n modules each. We wish to transform this into a $2m \times \frac{n}{2}$ array configuration. All of the same bounds on the physical properties of modules hold and gravity and friction are still ignored. Once more we wish to find the minimum movement time T for this reconfiguration.

Lemma 4. *Reconfiguring from an $m \times n$ array of unit-dimension modules to an array of $2m \times \frac{n}{2}$ unit-dimension modules takes $O(\sqrt{m} + \sqrt{n})$ movement time.*

The proof is omitted here to meet space requirements.

The problem just analyzed may now be used iteratively to solve the x-axis to y-axis reconfiguration problem. By repeatedly applying the above array reconfiguration step, we double the height and halve the width of the array each time until we progress from a $1 \times n$ to an $n \times 1$ array of modules.

This process will take $O(\lg_2 n)$ such steps, and so there would seem to be a danger of the reconfiguration problem requiring an extra $\lg_2 n$ factor in its movement time. This is avoided because far less time is required to reconfigure arrays in intermediate steps. Note that the $O(\sqrt{m} + \sqrt{n})$ time bound will be dominated by the larger of the two values m and n. For the first $\lfloor (\lg_2 n)/2 \rfloor$ steps the larger value will be the number of columns n, until an array of $\lfloor \sqrt{n} \rfloor \times \lceil \sqrt{n} \rceil$ dimensions is reached. In the next step a $\lceil \sqrt{n} \rceil \times \lfloor \sqrt{n} \rfloor$ array of modules is created, and from that point on we have more rows than columns and the time bound is dominated by m.

The key aspect is that the movement time for each reconfiguration step is decreased by half from the time we begin until we reach an intermediate array of dimensions about $\sqrt{n} \times \sqrt{n}$. By the principle of time reversal it should take us the same amount of movement time to go from a single row of n modules to an $\sqrt{n} \times \sqrt{n}$ cube as it does to go from that cube to a single column of n modules. This tactic is now used in our analysis to find the minimum reconfiguration time for the x-axis to y-axis problem.

Lemma 5. *The x-axis to y-axis reconfiguration problem only requires movement time $O(\sqrt{n})$.*

Proof: For simplicity, let n be even and let $n = p^2$ for some integer $p > 0$. Let $r(i)$ and $c(i)$ be the number of rows and columns, respectively, in the module system after i reconfiguration steps. From the previous Lemma 4 in this section we have that a single step of reconfiguring an $m \times n$ array of modules into a $2m \times \frac{n}{2}$ array requires time $\mathrm{O}(\sqrt{m} + \sqrt{n})$. Initially, assuming a large initial row length, then $c(0) = n$, $n \gg 1$, and the reconfiguration step takes $\mathrm{O}(\sqrt{n})$ time. For subsequent steps we still have $c(i) \gg r(i)$, but $c(1) = n/2, c(2) = n/4$, etc. while $r(1) = 2, r(2) = 4$, etc. In general $c(i) = n/2^i$ and $r(i) = 2^i$. Eventually, we get $c(i') = r(i') = \sqrt{n}$ at $i' = (\lg_2 n)/2$. Up until that point the time for each reconfiguration stage $i+1$ is $\mathrm{O}(\sqrt{c(i)})$. So, the total reconfiguration time to that point is given by the following summation.

$$\begin{aligned}
\sum_{i=0}^{(\lg_2 n)/2} \sqrt{c(i)} &= \sum_{i=0}^{(\lg_2 n)/2} \sqrt{\frac{n}{2^i}} \\
&\leq \sqrt{n} \sum_{i=0}^{\infty} \left(\frac{1}{\sqrt{2}}\right)^i \\
&= \frac{\sqrt{n}}{1 - (1/\sqrt{2})} \\
&= O(\sqrt{n}) \ .
\end{aligned}$$

Thus we have that reconfiguration from the initial row of n modules to the intermediate $\sqrt{n} \times \sqrt{n}$ square configuration takes $\mathrm{O}(\sqrt{n})$ movement time. Reconfiguring from this cube to the goal configuration is just the reverse operation: we are simply creating a "vertical row" now instead of a horizontal one. This reverse operation will then take the exact same movement time using the same amounts of force as the original operation, so it too takes $\mathrm{O}(\sqrt{n})$ movement time. Thus, we have that while satisfying the requirements of our abstract model the x-axis to y-axis problem takes $\mathrm{O}(\sqrt{n})$ movement time. □

6 Lower Bounds

In Section 4 we showed that the 1-dimensional Point Mass Contraction Problem, reconfiguring a row of n point masses with unit separation to have $1/2$ distance

separation, could be solved in time $T = \sqrt{n-1}$. We now show a matching lower bound for this problem. Again, the same assumptions about the physical properties of the point masses are held and concerns for friction or gravity are ignored.

Lemma 6. *The Point Mass Contraction Problem requires at least total movement time* $T = \sqrt{n-1}$.

Proof: Consider the movement of the 1st point mass which needs to move from initial x-coordinate $x_1(0) = 1-[n+1]/2 = 1/2-n/2$ starting with initial velocity $v_1(0) = 0$ to final x-coordinate $x_1(T) = x_1(0)/2 = 1/4 - n/4$ and ending with final velocity $v_1(T) = 0$. The total distance this point mass needs to travel is $x_1(T) - x_1(0) = [n-1]/4$.

Taking into consideration the constraint that the final velocity is to be 0, it is easy to verify that the time-optimal trajectory for point mass $i = 1$ is an acceleration of 1 in the positive x-direction from time 0 to time $T/2$, followed by a reverse acceleration of the same magnitude in the negative x-direction. The total distance traversed in each of the two stages is at most $[a_1/2]t^2 = 1 * [T/2]^2/2$. So, the total distance traversed by the 1st point mass is at most $[T/2]^2 = T^2/4$ which needs to be $[n-1]/4$. Hence, $T^2/4 \geq [n-1]/4$ and so $T \geq \sqrt{n-1}$. □

Hence, we have shown:

Theorem 1. *The lower and upper bound for the total movement time for the Point Mass Contraction Problem is exactly* $T = \sqrt{n-1}$.

As in Section 4, we can use an extension of this lower bound argument to prove the following lemma.

Lemma 7. *The Squeeze problem requires total movement time* $\Omega(\sqrt{n})$.

The proof is omitted here to meet space requirements.

Hence, we have also shown:

Theorem 2. *The total movement time for the Squeeze problem is both upper and lower bounded by* $\Theta(\sqrt{n})$.

By the same argument, we can also get a bound on the x-axis to y-axis problem.

Corollary 2. *The x-axis to y-axis reconfiguration problem requires total movement time* $\Omega(\sqrt{n})$.

Proof: Given the initial row configuration and the goal column configuration, pick the end of the row farthest away from the goal column configuration's horizontal placement. Select the n/c cubes at this end of the row, for some real number greater than 2. Among these n/c cubes we can again find some ith cube that must have an acceleration at most $a_i \approx c/2$ and that it must travel a

distance $\geq n[1/4 - 1/[2c]] + 1/4$. This again leads to the movement time being bounded as $T = \Omega(\sqrt{n})$. □

7 Conclusion

In this paper we have presented a novel abstract model for self-recongurable (SR) robots that provides a basis for kinodynamic motion planning for these robots. Our model explicitly requires that SR robot modules have unit bounds on their size, mass, magnitude of force or torque they can apply, and the relative velocity between directly connected modules. The model allows for feasible physical implementations and permits the use of basic laws of physics to derive improved reconguration algorithms and lower bounds.

In this paper we have focused on a simple and basic reconguration problem. Our main results were tight upper and lower bounds for the movement time for this problem. Our recursive *Squeeze* algorithm recongures a horizontal row of n modules into a vertical column in $O(\sqrt{n})$-time. This result significantly improves on the running time of previous reconguration algorithms. Our algorithm satisfies the restrictions imposed by our abstract model and we also show that it is kinodynamically optimal given the assumptions of that model.

While carefully using the forces produced by modules, our analysis ignored forces caused by gravity and friction. Addressing these concerns is a topic for future work as the algorithm is brought closer to physical implementation. Also, the algorithm given was a centralized planner and only solved a simple example to demonstrate how faster reconfiguration algorithms were possible. Yet the multipart nature of SR robots makes distributed algorithms a necessity. Extending our lower-bound analysis to more complex analysis, and developing distributed algorithms to match those bounds, is a topic of future work.

References

1. Casal, A., Yim, M.: Self-reconfiguration planning for a class of modular robots. In: Proceedings of SPIE, Sensor Fusion and Decentralized Control in Robotic Systems II, vol. 3839, pp. 246–255 (1999)
2. Donald, B.R., Xavier, P.G., Canny, J.F., Reif, J.H.: Kinodynamic motion planning. Journal of the ACM 40(5), 1048–1066 (1993)
3. Kotay, K., Rus, D.: Generic distributed assembly and repair algorithms for self-reconfiguring robots. In: Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems (2004)
4. Pamecha, A., Chiang, C., Stein, D., Chirikjian, G.: Design and implementation of metamorphic robots. In: Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference (1996)
5. Pamecha, A., Ebert-Uphoff, I., Chirikjian, G.: Useful metrics for modular robot motion planning. In: IEEE Trans. Robot. Automat., pp. 531–545 (1997)
6. Vassilvitskii, S., Kubica, J., Rieffel, E., Suh, J., Yim, M.: On the general reconfiguration problem for expanding cube style modular robots. In: Proceedings of the 2002 IEEE Int. Conference on Robotics and Automation, May 11-15, pp. 801–808 (2002)

7. Vona, M., Rus, D.: Self-reconfiguration planning with compressible unit modules. In: 1999 IEEE International Conference on Robotics and Automation (1999)
8. Walter, J.E., Welch, J.L., Amato, N.M.: Distributed reconfiguration of metamorphic robot chains. In: PODC 2000, pp. 171–180 (2000)
9. Yim, M., Duff, D., Roufas, K.: Polybot: A modular reconfigurable robot. In: ICRA, pp. 514–520 (2000)