


Optimal In-Place Compaction of Sliding Cubes

Irina Kostitsyna ✉ 

TU Eindhoven, The Netherlands

Tim Ophelders ✉ 

Utrecht University, The Netherlands

TU Eindhoven, The Netherlands

Irene Parada ✉ 

Universitat Politècnica de Catalunya, Barcelona, Spain

Tom Peters ✉ 

TU Eindhoven, The Netherlands

Willem Sonke ✉ 

TU Eindhoven, The Netherlands

Bettina Speckmann ✉ 

TU Eindhoven, The Netherlands

Abstract

The sliding cubes model is a well-established theoretical framework that supports the analysis of reconfiguration algorithms for modular robots consisting of face-connected cubes. As is common in the literature, we focus on reconfiguration via an intermediate canonical shape. Specifically, we present an in-place algorithm that reconfigures any n -cube configuration into a compact canonical shape using a number of moves proportional to the sum of coordinates of the input cubes. This result is asymptotically optimal and strictly improves on all prior work. Furthermore, our algorithm directly extends to dimensions higher than three.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases Sliding cubes, Reconfiguration algorithm, Modular robots

Digital Object Identifier 10.4230/LIPIcs.SWAT.2024.31

Related Version *Full Version*: <https://arxiv.org/abs/2312.15096>

Supplementary Material *Audiovisual*: <https://www.computational-geometry.org/SoCG-videos/socg24video/optimal-in-place-compaction-of-sliding-cubes.mp4>

Funding *Tim Ophelders*: partially supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.

1 Introduction

Modular robots consist of a large number of comparatively simple robotic units. These units can attach and detach to and from each other, move relative to each other, and in this way form different shapes or configurations. This shape-shifting ability allows modular robots to robustly adapt to previously unknown environments and tasks. One of the major questions regarding modular robots is *universal reconfiguration*: is there a sequence of moves which transforms any two given configurations into each other, and if so, how many moves are necessary? There are a variety of real-world mechatronics or theoretical computational models for modular robots and the answer to the universal reconfiguration question differs substantially between systems [3].

In this paper, we study the *sliding cube model*, which is a well-established theoretical framework that supports the analysis of reconfiguration algorithms for modular robots consisting of face-connected cubes. In this model, a module (cube) can perform two types of moves: straight-line moves called *slides* and moves around a corner called *convex transitions*



© Irina Kostitsyna, Tim Ophelders, Irene Parada, Tom Peters, Willem Sonke, and Bettina Speckmann; licensed under Creative Commons License CC-BY 4.0

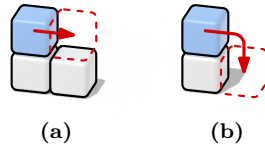
19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024).

Editor: Hans L. Bodlaender; Article No. 31; pp. 31:1–31:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** Moves in the sliding cube model: slide (a) and convex transition (b). Solid cubes are part of the configuration.

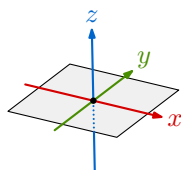
(see Figure 1). Maintaining connectivity during a sequence of moves is the main challenge when developing algorithms in the sliding cube model. During a move, the configuration (excluding the moving cube) must stay connected. Furthermore, there have to be sufficient empty cells to perform the move. This connectivity is crucial for most actual modular robotic systems since it allows them to retain their structure, communicate, and share other resources such as energy.

Almost 20 years ago, Dumitrescu and Pach [8] showed that the sliding cube model in 2D (or *sliding square model*) is universally reconfigurable. More precisely, they presented an algorithm that transforms any two given configurations with n squares into each other in $O(n^2)$ moves. This algorithm transforms any given configuration into a canonical shape (a horizontal line) and then reverts the procedure to reach the final configuration. It was afterwards adapted to be *in-place* using flooded bounding boxes as canonical intermediate configurations [15]. Recently, Akitaya et al. [4] presented Gather&Compact: an input-sensitive in-place algorithm which uses $O(Pn)$ moves, where P is the maximum among the perimeters of the bounding boxes of the initial and final configurations. The authors also show that minimizing the number of moves required to reconfigure is NP-hard.

These algorithms in 2D do not directly transfer to 3D: they fundamentally rely on the fact that a connected cycle of squares encloses a well-defined part of the configuration. One could ask whether the fact that enclosing space in 3D is more difficult has positive or negative impact on universal reconfiguration in 3D. Miltzow et al. [14] showed that there exist 3D configurations in which no module on the external boundary is able to move without disconnecting the configuration. Hence, simple reconfiguration strategies [11, 13] can generally not guarantee reconfiguration for all instances.

Until very recently, the most efficient algorithm for the reconfiguration problem in 3D was the algorithm by Abel and Kominers [1], which uses $O(n^3)$ moves to transform any n -cube configuration into any other n -cube configuration. As is common in the literature, this algorithm reconfigures the input into an intermediate canonical shape. Stock et al. [17] just announced a worst-case bound of $O(n^2)$ moves for the Abel and Kominers algorithm. Furthermore, their paper presents an in-place reconfiguration algorithm, which runs in time proportional to a measure of the size of the bounding box times the number of cubes. Specifically, their algorithm requires $O(n(wd + h))$ moves in the worst-case, where w , d , and h are the width, depth, and height of the bounding box, respectively.

Results. In this paper we present an in-place algorithm that reconfigures any n -cube configuration into a compact canonical shape using a number of moves proportional to the sum of coordinates of the input cubes. This result is asymptotically optimal and strictly stronger than the bounds obtained by Stock et al. [17]. Furthermore, our algorithm directly extends to hypercube reconfiguration in dimensions higher than three. Last but not least, the restriction of our algorithm to two dimensions improves upon the best bound for compacting sliding squares [4].



Additional related work. For more restricted sliding models, for example, only allowing one of the two moves in the sliding cube model, reconfiguration is not always possible. Michail et al. [13] explore universal reconfiguration using *helpers* or *seeds* (dedicated cubes that help other cubes move). They show that the problem of deciding how many seeds are needed is in PSPACE.

Another popular model for modular robots is the pivoting cube model, in which the modules move by rotating around an edge shared with a neighboring module. In this model the extra free-space requirements for the moves that come from pivoting instead of sliding mean that there are configurations in which no move is possible. Akitaya et al. [3] show that the reconfiguration problem in this setting is PSPACE-complete. In contrast, adding five additional modules to the outer boundary guarantees universal reconfigurability in 2D using $O(n^2)$ moves [2]. Other algorithms for pivoting modules require the absence of narrow corridors in both the initial and final configurations [10, 18]. A more powerful move is to allow the modules to *tunnel* through the configuration. With it, $O(n)$ parallel steps suffice to reconfigure 2D and 3D cubes [5, 6, 12]. However, for most real-world prototype systems, tunnelling requires the use of *metamodules* [16] which are sets of modules which act as a single unit with enhanced capabilities, increasing the granularity of the configurations.

We require that the configuration stays connected at all times. In a slightly different model that relaxes the connectivity requirement (referred to as the *backbone property*), Fekete et al. [9] show that scaled configurations of labeled squares can be efficiently reconfigured using parallel coordinated moves with a schedule that is a constant factor away from optimal.

2 Preliminaries

In this paper, we study cubical modules moving in the 3-dimensional grid \mathbb{Z}^3 . The handedness of the coordinate system does not have any impact on the correctness of our algorithm.

A *configuration* \mathcal{C} is a subset of coordinates in the grid. The elements of \mathcal{C} are called *cubes*. We call two cubes *adjacent* if they lie at unit distance. For a configuration \mathcal{C} , denote by $G_{\mathcal{C}}$ the graph with vertex set \mathcal{C} , whose edges connect all adjacent cubes. We say a *cell* is a vertex of $G_{\mathbb{Z}^3}$ which is not occupied by a cube in \mathcal{C} . We always require a configuration to remain *connected*, that is, $G_{\mathcal{C}}$ must be connected. For ease of exposition we assume \mathcal{C} consists of at least two cubes. Let $B_{\mathcal{C}}$ be the bounding box of a configuration \mathcal{C} . W.l.o.g. we assume that the vertex in $B_{\mathcal{C}}$ with minimum x -, y -, and z -coordinate is the origin of $G_{\mathbb{Z}^3}$.

In the sliding cubes model, a configuration can rearrange itself by letting cubes perform moves. A move replaces a single cube $c \in \mathcal{C}$ by another cube $c' \notin \mathcal{C}$. Moves come in two types: *slides* and *convex transitions* (see Figure 1). In both cases, we consider a 4-cycle γ in $G_{\mathbb{Z}^3}$. For slides, exactly three cubes of γ are in \mathcal{C} ; c' is the cell of γ not in \mathcal{C} , and c is adjacent to c' . For convex transitions, γ has exactly two adjacent cubes in \mathcal{C} ; c is one of these two cubes, and c' is the vertex of γ not adjacent to c . The slide or convex transition is a *move* if and only if $\mathcal{C} \setminus \{c\}$ is connected.

Call a cube $c = (x, y, z)$ *finished* if the cuboid spanned by the origin and c is completely in \mathcal{C} , that is, if $\{0, \dots, x\} \times \{0, \dots, y\} \times \{0, \dots, z\} \subseteq \mathcal{C}$. We call \mathcal{C} *finished* if all cubes in \mathcal{C} are finished. The *compaction problem* starts with an arbitrary connected configuration \mathcal{C} with bounding box $B_{\mathcal{C}}$ and is solved when all cubes are finished. An algorithm for this problem is *in-place* if at most a single cube simultaneously moves through cells face-adjacent to $B_{\mathcal{C}}$.

Most of the algorithm works on vertical contiguous strips of cubes in \mathcal{C} called subpillars. More precisely, a *subpillar* of \mathcal{C} is a subset of \mathcal{C} of the form $\{x\} \times \{y\} \times \{z_b, \dots, z_t\}$. In the remainder of this paper, we denote this subpillar by $\langle x, y, z_b .. z_t \rangle$. The cube (x, y, z_t) is called the *head*, and the remainder $\langle x, y, z_b .. z_t - 1 \rangle$ is called the *support* of the subpillar. A *pillar* is a maximal subpillar, that is, a subpillar that is not contained in any other subpillar. Note that there can be multiple pillars above each other with the same x - and y -coordinates, as long as there is a gap between them. Two sets S and S' of cubes are *adjacent* if S contains a cube adjacent to a cube in S' . The *pillar graph* \mathcal{P}_S of a set S of cubes is the graph whose vertices are the pillars of S and whose edges connect adjacent pillars.

3 Algorithm

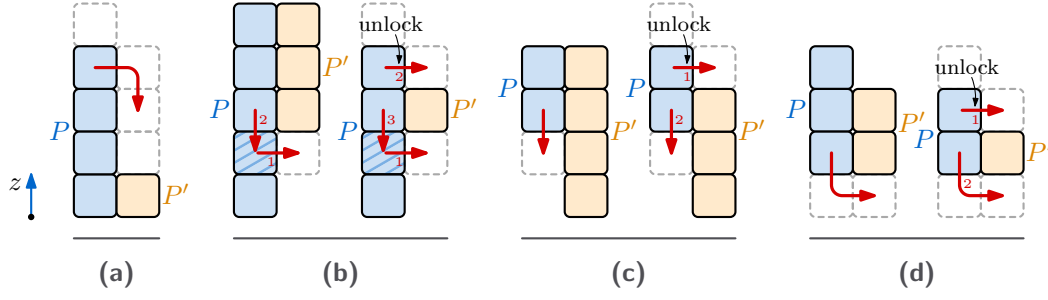
For a set of cubes $S \subseteq \mathcal{C}$, let its *coordinate vector sum* be $(X_S, Y_S, Z_S) = \sum_{(x,y,z) \in S} (x, y, z)$. Let $\mathcal{C}_{>0}$ be the subset of cubes $(x, y, z) \in \mathcal{C}$ for which $z > 0$, and \mathcal{C}_0 be the subset of cubes for which $z = 0$. Let the *potential* of a cube $c = (c_x, c_y, c_z)$ be $\Pi_c = w_c(c_x + 2c_y + 4c_z)$, where the *weight* w_c depends on the coordinates of c in the following way. If $c_z > 1$, then $w_c = 5$; if $c_z = 1$, then $w_c = 4$. If $c_z = 0$, then w_c depends on c_y . If $c_y > 1$, then $w_c = 3$; if $c_y = 1$, then $w_c = 2$ and lastly, if $c_z = c_y = 0$, then $w_c = 1$. We aim to minimize the *potential function* $\Pi_{\mathcal{C}} = \sum_{c \in \mathcal{C}} \Pi_c$. From now on, let \mathcal{C} be an unfinished configuration. We call a sequence of m moves applied to \mathcal{C} *safe* if the result is a configuration \mathcal{C}' , such that $\Pi_{\mathcal{C}'} < \Pi_{\mathcal{C}}$ and $m = O(\Pi_{\mathcal{C}} - \Pi_{\mathcal{C}'})$. This means that the sequence of moves reduces the potential by at least some constant fraction of m by going from \mathcal{C} to \mathcal{C}' . We show that if \mathcal{C} is unfinished, it always admits a safe move sequence.

The main idea is as follows. For a configuration \mathcal{C} , whenever possible, we try to reduce $Z_{\mathcal{C}}$ by some sequence of moves. If that is not possible, then the configuration must admit another sequence of moves, where a complete pillar is moved to a different x - or y -coordinate. In this way, by reducing either the z -coordinate of cubes, or the x - or y -coordinate, we guarantee that eventually every cube becomes finished.

In this paper, we describe the algorithm in three dimensions. However, it naturally extends to higher dimensions, and also works for squares in two dimensions. In fact, we will use the algorithm in two dimensions as a subroutine for the three-dimensional case.

Local z reduction. Let $P = \langle x, y, z_b .. z_t \rangle$ be a subpillar of \mathcal{C} . We refer to the four coordinates $\{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$ as the *sides* of P . On each side, P has zero or more adjacent pillars. We order these by their z -coordinates; as such, we may refer to the top- or bottommost adjacent pillar on a side of P . We say that a set of cubes $S \subseteq \mathcal{C}$ is *non-cut* if $G_{\mathcal{C} \setminus S}$ is connected or empty. A pillar of \mathcal{C} is non-cut if and only if it is a non-cut vertex of the pillar graph $\mathcal{P}_{\mathcal{C}}$.

Let $P = \langle x, y, z_b .. z_t \rangle$ be a non-cut subpillar, and let $P' = \langle x', y', z'_b .. z'_t \rangle$ be a pillar adjacent to P . We define a set of operations of at most three moves within P which locally reduce $Z_{\mathcal{C}}$ (see Figure 2). Because P is non-cut, $\mathcal{C} \setminus P$ is connected. Therefore, if cubes of P move in such a way that each component (of cubes originally in P) remains adjacent to a cube of $\mathcal{C} \setminus P$, then the result of that operation is a valid configuration.



■ **Figure 2** Examples of operations (a–d); hatched cubes are non-cut and dashed outlines indicate cells that must be empty. Each case admits a move sequence that reduces Z_C .

- (a) If P' is a topmost adjacent pillar of P and $z'_t \leq z_t - 2$, then the topmost cube of P admits a convex transition that decreases Z_C .
- (b) If $z'_b > z_b$ and $(x, y, z'_b - 1)$ is a non-cut cube, then there is a move sequence that decreases Z_C : first slide $(x, y, z'_b - 1)$ to $(x', y', z'_b - 1)$ and then slide (x, y, z'_b) to $(x, y, z'_b - 1)$. There is one special case. We say that P is *locked* if the head of P has no adjacent cubes except for P' 's support. If P is locked and $z'_b = z_t - 1$, then the second slide would disconnect P' 's head from the rest of the configuration. To avoid this, before performing the second slide, we *unlock* P by sliding the head of P from (x, y, z_t) to (x', y', z_t) , as shown in the right part of Figure 2b.
- (c) If $(x, y, z_b - 1) \notin \mathcal{C}$ and $z'_b < z_b$, then (after unlocking P , if necessary) (x, y, z_b) admits a slide to $(x, y, z_b - 1)$ that decreases Z_C .
- (d) If $(x, y, z_b - 1) \notin \mathcal{C}$, P' is a bottommost adjacent pillar of P , and $z_b = z'_b > 0$, then (after unlocking P , if necessary) (x, y, z_b) admits a convex transition to $(x', y', z'_b - 1)$ that decreases Z_C .

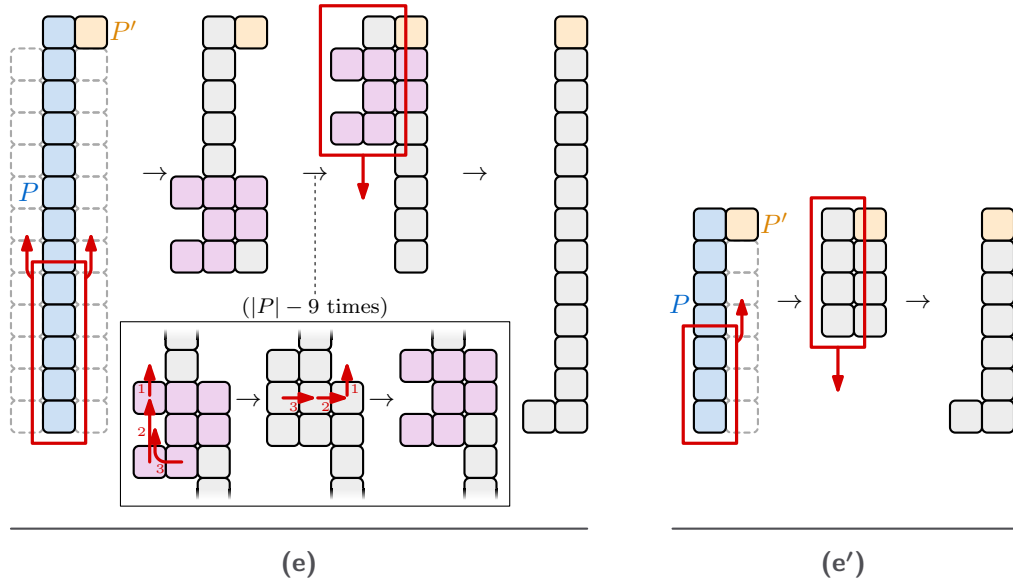
► **Lemma 1.** Let $P = \langle x, y, z_b \dots z_t \rangle$ be a non-cut pillar. Assume P does not admit any operation of type (a–d). Then, on each side, P has at most one adjacent pillar $P' = \langle x', y', z'_b \dots z'_t \rangle$. For these pillars P' , we have $z_t \leq z'_t + 1$, and either $z_b < z'_b$ or $z_b = z'_b = 0$.

Proof. Consider one side s of P . Because (a) does not apply to P , for any adjacent pillar $P' = \langle x', y', z'_b \dots z'_t \rangle$, we know that $z_t \leq z'_t + 1$. Consider the case that $(x, y, z'_b - 1)$ is a non-cut cube. Because (b) does not apply, $z_b \geq z'_b$, and because (c) does not apply either, $z_b = z'_b$, and finally because (d) does not apply, we have $z_b = z'_b = 0$. Now consider the case that $(x, y, z'_b - 1)$ is a cut cube. Then because (c) does not apply, we have $z_b \leq z'_b$, and finally because (d) does not apply, we have $z_b < z'_b$ or $z_b = z'_b = 0$. If $z_b = z'_b = 0$ for each adjacent pillar P' , then each side of P can have at most one pillar. If $z_b < z'_b$, then, because (b) does not apply, $(x, y, z'_b - 1)$ is a cut cube. Therefore, also in this case, there can be no cube adjacent to the subpillar $\langle x, y, z_b \dots z'_b - 2 \rangle$, and therefore also in this case there can be at most one adjacent pillar on each side. ◀

Pillar shoves. Next, we consider longer move sequences that still involve a single subpillar. A central operation of our algorithm is a *pillar shove*, which takes as parameters a subpillar $P = \langle x, y, z_b \dots z_t \rangle$ and a side (x', y') of P . The result of the pillar shove is the set of cubes

$$\text{shove}(\mathcal{C}, P, (x', y')) := (\mathcal{C} \setminus P) \cup \langle x', y', z_b \dots z_t - 1 \rangle \cup \{(x, y, z_b)\},$$

in which the support is effectively shifted to the side (x', y') , and the head is effectively moved from (x, y, z_t) to (x, y, z_b) . Although $\text{shove}(\mathcal{C}, \langle x, y, z_b \dots z_t \rangle, (x', y'))$ is well-defined, it is not necessarily a connected configuration, let alone safely reachable from \mathcal{C} .



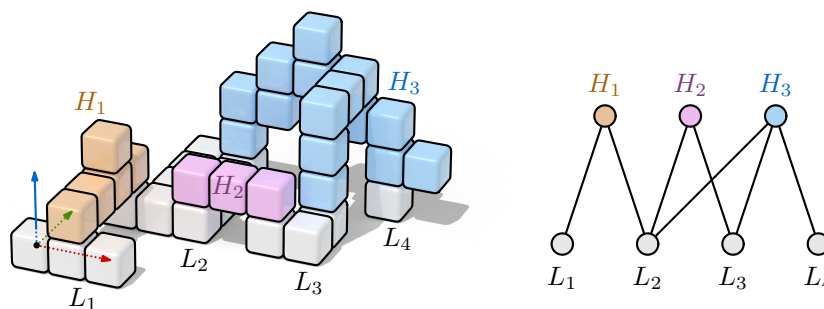
■ **Figure 3** Examples of pillar shoves for a long pillar (e) and a short pillar (e'). The zipper operation on the left is executed $|P| - 9$ times.

Let $P = \langle x, y, z_b .. z_t \rangle$ be a non-cut subpillar, and assume that on at least two sides (x', y') and (x'', y'') of P , no cube except possibly the head (x, y, z_t) has an adjacent cube. Moreover, assume that $(x', y', z_t) \in \mathcal{C}$. We define two types of pillar shoves, each of which transforms \mathcal{C} into $\text{shove}(\mathcal{C}, \langle x, y, z_b .. z_t \rangle, (x', y'))$: a *long pillar shove* (for $|P| \geq 9$; see Figure 3(e)) and a *short pillar shove* (for $|P| < 9$; see Figure 3(e')). Note that the short pillar shove could be applied to the $|P| \geq 9$ case as well. However, a short pillar shove takes a number of moves quadratic in $|P|$ and hence would not be safe. A long pillar shove, on the other hand, takes a number of moves linear in $|P|$ as for each cube, we take a constant number of moves to move it to its new location (the “zipper” operation shown in the framed part of Figure 3(e)). As such, both pillar shoves reduce $Z_{\mathcal{C}}$ by $z_t - z_b$ and take $O(z_t - z_b)$ moves, so they are safe.

(e) Assume that no operations of type (a–d) are possible. Then, by Lemma 1, $P = \langle x, y, z_b .. z_t \rangle$ has at most one adjacent pillar on each side, and there exists an adjacent pillar $P' = \langle x', y', z'_b .. z'_t \rangle$ with $z'_b > z_b$ (assume that P' is such a pillar with the lowest z'_b), and there is a side $(x'', y'') \neq (x', y')$ of P such that $(x'', y'', z_b) \notin \mathcal{C}$. Together, this implies that both sides (x'', y'') and (x', y') are empty up to at least $z'_b - 1$ (otherwise P' would not be the pillar with lowest $z'_b > z_b$). Then the subpillar $\langle x, y, z_b .. z'_b \rangle$ (after unlocking P , if necessary) admits a pillar shove.

▶ **Lemma 2.** *Let $P = \langle x, y, z_b .. z_t \rangle$ be a non-cut subpillar. Assume P does not admit any operation of type (a–e). Then P has no adjacent pillar $\langle x', y', z'_b .. z'_t \rangle$ with $z'_b > z_b$.*

Proof. Assume that P has at least two adjacent pillars, say $P' = \langle x', y', z'_b .. z'_t \rangle$ and $P'' = \langle x'', y'', z''_b .. z''_t \rangle$, such that $z_b < z'_b$ and $z_b < z''_b$; let P' denote the lowest one, such that $z_b < z'_b \leq z''_b$. Then $(x'', y'', z_b) \notin \mathcal{C}$, which contradicts that (e) does not apply. Therefore, there can be at most one such pillar. However, this, together with Lemma 1, implies that on all other sides $(x'', y'') \neq (x', y')$, $(x'', y'', z_b) \in \mathcal{C}$. This means that $(x, y, z'_b - 1)$ is a non-cut cube, contradicting that (b) does not apply. Therefore, there can be no such adjacent pillars. ◀



■ **Figure 4** An example configuration \mathcal{C} and its low-high graph $\mathcal{LH}_{\mathcal{C}}$. This configuration does still admit operations of type (a–g).

In summary, if (a–e) do not apply to any non-cut subpillar, then for any non-cut pillar $P = \langle x, y, z_b \dots z_t \rangle$ and any adjacent pillar $\langle x', y', z'_b \dots z'_t \rangle$, we have $z'_b = z_b = 0$.

Local potential reduction. Let $\mathcal{C}_{>0}$ be the subconfiguration consisting of cubes with $z > 0$. We may greedily reduce the potential by moving individual cubes in $\mathcal{C}_{>0}$.

(f) Perform any move of \mathcal{C} that moves a cube c of $\mathcal{C}_{>0}$, reduces the potential, and keeps c inside the bounding box $B_{\mathcal{C}}$ of \mathcal{C} .

► **Lemma 3.** *If an unfinished configuration \mathcal{C} does not admit any operation of type (a–f), and some maximal connected component of $\mathcal{C}_{>0}$ consists of a single pillar $P = \langle x, y, z_b \dots z_t \rangle$, then $P = \{(0, 0, 1)\}$ and $(0, 0, 0) \in \mathcal{C}$.*

Proof. Because $\mathcal{C}_{>0}$ does not contain cubes with $z = 0$, we have $z_b = 1$ or $z_b > 1$. If $z_b > 1$, then $\langle x, y, z_b \dots z_t \rangle$ would be disconnected from the rest of \mathcal{C} , so this cannot be the case. Likewise, if $z_b = 1$ then $(x, y, 0) \notin \mathcal{C}$ would mean that $\langle x, y, z_b \dots z_t \rangle$ is disconnected from \mathcal{C} , so this cannot be the case either. Therefore $z_b = 1$ and $(x, y, 0) \in \mathcal{C}$. If $z_t > 1$, then the topmost cube of P can do a convex transition to $(x + 1, y, z_t - 1)$, reducing the potential. Therefore $z_t = 1$ and $P = \{(x, y, 1)\}$. If $x > 0$ or $y > 0$, then we can move the single cube of P : using the cube at $(x, y, 0)$, the cube of P can slide or convex transition closer to the origin $(0, 0, 0)$. Therefore, $z_b = z_t = 1$ and $x = y = 0$. ◀

► **Corollary 4.** *If a configuration \mathcal{C} does not admit any operation of type (a–f) then of the connected components of $\mathcal{C}_{>0}$, at most one consists of a single pillar.*

Low and high components. Let $\mathcal{LH}_{\mathcal{C}}$ be the bipartite graph obtained from $G_{\mathcal{C}}$ by contracting the components of $G_{\mathcal{C}_0}$ and $G_{\mathcal{C}_{>0}}$ to a single vertex (see Figure 4). We call $\mathcal{LH}_{\mathcal{C}}$ the *low-high graph* of \mathcal{C} , and we call the vertices of $\mathcal{LH}_{\mathcal{C}}$ that correspond to components of $G_{\mathcal{C}_0}$ and $G_{\mathcal{C}_{>0}}$ *low* and *high* components, respectively. For brevity, we may refer to a low or high component by its corresponding vertex in $\mathcal{LH}_{\mathcal{C}}$ and vice versa.

We will use the following lemma several times.

► **Lemma 5.** *Let H be a high component and P be a pillar of H . For every component H' of $H \setminus P$, there exists a non-cut pillar of H' that is also a non-cut pillar of H .*

Proof. Any component with at least two pillars contains at least two non-cut pillars and every component contains at least one non-cut pillar, so let P' be an arbitrary non-cut pillar of H' . $H \setminus P'$ has at most two components, namely $H' \setminus P'$ and $H \setminus H'$. If P' is a non-cut pillar

31:8 Optimal In-Place Compaction of Sliding Cubes

of H , the lemma holds. Else, if P' is a cut pillar of H , then it has exactly these components, so $H' \setminus P'$ is nonempty and P is adjacent to P' . Therefore, H' consists of multiple pillars and hence contains at least two non-cut pillars. Let $P'' \neq P'$ be a second non-cut pillar of H' . We claim that P'' is also a non-cut pillar of H . Indeed, because P and P' are adjacent, the sets $H' \setminus P'' \supseteq P'$ and $H \setminus H' \supseteq P$ are adjacent. Hence, $H \setminus P'' = (H' \setminus P'') \cup (H \setminus H')$ has a single component, so P'' is a non-cut pillar of H . ◀

► **Lemma 6.** *Assume \mathcal{C} does not admit any operation of type (a-f). Suppose that H is a high component such that $\mathcal{C} \setminus H$ is connected. Then any pillar of H is a non-cut subpillar of \mathcal{C} .*

Proof. Suppose for a contradiction that a pillar P of H is a cut subpillar of \mathcal{C} . Then $\mathcal{C} \setminus P$ contains at least one component H' that does not intersect $\mathcal{C} \setminus H$. Therefore, H' is also a component of $H \setminus P$, so by Lemma 5, there exists a non-cut pillar $P' = \langle x', y', z'_b \dots z'_t \rangle$ of H' that is also a non-cut pillar of H . If $z'_b > 1$ or $(x', y', 0) \notin \mathcal{C}$, then P' would be a non-cut pillar of \mathcal{C} . If \mathcal{C} does not admit any operation of type (a-f), all non-cut pillars of \mathcal{C} start at $z = 0$, which contradicts $z'_b > 1$ or $(x', y', 0) \notin \mathcal{C}$. Therefore, $z'_b = 1$ and $(x', y', 0) \in \mathcal{C}$, but then H' would not be a component of $\mathcal{C} \setminus P$, as H' is adjacent to $(x', y', 0) \in \mathcal{C} \setminus P$. Hence, H' cannot exist, completing the proof. ◀

► **Corollary 7.** *If H is a high component such that $\mathcal{C} \setminus H$ is connected, then every pillar of H is part of a pillar of \mathcal{C} starting at $z = 0$.*

► **Lemma 8.** *Assume \mathcal{C} does not admit any operation of type (a-f). If H is a high component such that $\mathcal{C} \setminus H$ is connected, then H consists entirely of finished cubes.*

Proof. Assume for contradiction that H contains an unfinished cube. Because of Corollary 7, every pillar of H is part of a pillar of \mathcal{C} starting at $z = 0$. Therefore, H contains an unfinished cube (x, y, z) with $x > 0$ or $y > 0$. Let c be such a cube that lexicographically maximizes $(z, -y, -x)$. If $x > 0$ and $(x - 1, y, z) \notin H$ (and thus $(x - 1, y, z) \notin \mathcal{C}$), then we can move c to either $(x - 1, y, z)$ or $(x - 1, y, z - 1)$, reducing the potential while keeping all cubes within the bounding box $B_{\mathcal{C}}$, so (f) would apply. If $y > 0$ and $(x, y - 1, z) \notin H$, then we can similarly move c to either $(x, y - 1, z)$ or $(x, y - 1, z - 1)$. On the other hand, if both (1) $x = 0$ or $(x - 1, y, z) \in H$ and (2) $y = 0$ or $(x - 1, y, z) \in H$, then because c is the unfinished cube of H that maximizes $(z, -y, -x)$, the cubes $(x - 1, y, z)$ (if $x > 0$) and $(x, y - 1, z)$ (if $y > 0$) are finished, but then (x, y, z) would also be finished. Contradiction. ◀

► **Corollary 9.** *There is at most one high component that contains a finished cube, as any high component that contains a finished cube also contains $(0, 0, 1)$.*

Handling low components. We pick a vertex R of $\mathcal{LH}_{\mathcal{C}}$ that we call the *root* of $\mathcal{LH}_{\mathcal{C}}$. If $(0, 0, 0) \in \mathcal{C}$, pick R to be the low component that contains $(0, 0, 0)$. Otherwise, pick R to be an arbitrary low component. Let d be the maximum distance in the graph $\mathcal{LH}_{\mathcal{C}}$ from R to any vertex. Let \mathcal{U} be the set of vertices of $\mathcal{LH}_{\mathcal{C}}$ that are locally furthest away (in $\mathcal{LH}_{\mathcal{C}}$) from R . That is, all neighbors v of a vertex $u \in \mathcal{U}$ lie closer to R . All vertices of \mathcal{U} are non-cut subsets of \mathcal{C} . Therefore, if \mathcal{U} contains a high component H , then H consists entirely of finished cubes (and H is adjacent to R), so \mathcal{U} contains at most one high component.

If $d = 0$, then \mathcal{C} consists of a single low component. If $d = 1$, then \mathcal{C} consists of one high and one low component. Set \mathcal{U} contains exactly one high component, and it consists entirely of finished cubes. If $d \geq 2$, then \mathcal{C} consists of at least two low components and \mathcal{U} consists of at least one low component, and at most one high component. We now give operations that can be executed when $d \geq 2$, such that we end up with a configuration where $d = 0$ or $d = 1$. We will show how to handle the case where $d = 0$ or $d = 1$ afterwards.

We call a low component L *clear* if $\mathcal{C} \setminus L$ is connected, $L \neq R$, and L is connected to a non-cut pillar P in $\mathcal{C} \setminus L$. We call such a pillar P a *clearing pillar*. We show in Lemma 10 that if $d \geq 2$, there is at least one clear low component. For this, consider a low component that is furthest from R (that is, at distance d), and let H be an adjacent high component. Let \mathcal{L}_H be the set of low components in \mathcal{U} that are adjacent to H (and hence also lie at distance d from R).

► **Lemma 10.** *At least one low component $L \in \mathcal{L}_H$ is connected to H via a non-cut pillar of $\mathcal{C} \setminus L$.*

Proof. Let $\mathcal{C}' = \mathcal{C} \setminus \bigcup_{L \in \mathcal{L}_H} L$. Let $H_{\mathcal{L}_H}$ be the set of cubes of H that are adjacent to a low component in \mathcal{L}_H . Fix an arbitrary cube c_s of R , and in the graph $G_{\mathcal{C}'}$, consider a cube c of $H_{\mathcal{L}_H}$ that is farthest from c_s . Let P be the pillar of H that contains c . We will show that P is a non-cut pillar of \mathcal{C}' . Suppose for a contradiction that P is a cut pillar, then $\mathcal{C}' \setminus P$ contains at least two components, at most one of which contains R . Let H' be a component of $\mathcal{C}' \setminus P$ not containing R . If H' contains a cube not in H , then that cube lies in a low or high component of \mathcal{C}' that lies closer to R , which therefore remains connected to R after removing P . Therefore, H' is a subset of H .

All cubes $(x, y, z) \in H'$ with $z = 1$ lie farther from c_s than c , and therefore H' does not contain any cubes of $H_{\mathcal{L}_H}$, so H' is not adjacent to any cubes of \mathcal{L}_H . Therefore, H' is not adjacent to any cubes of $(\mathcal{C} \setminus P) \setminus H'$. By Lemma 5, H' contains at least one non-cut pillar P' that is also a non-cut pillar of H . P' is also a non-cut pillar of \mathcal{C} , but all non-cut pillars of \mathcal{C} start at $z = 0$ (Corollary 7), which is a contradiction. ◀

We will now present an algorithm that repeatedly selects a (non-root) clear low component L , and performs the following operation on it:

(g) Perform any move of \mathcal{C} that moves a cube c of L , reduces the potential, and keeps c inside the bounding box $B_{\mathcal{C}}$ of \mathcal{C} .

Note that (g) is essentially the same as (f), but now executed on a low rather than a high component. When operations of type (g) are executed, one of three special events can occur:

- (1) L connects to a different low component, merging them.
- (2) L connects to the root, and becomes part of the root.
- (3) L reaches the origin (at which point it becomes the root).

When none of the operations (a–g) are available for a clear low component L with clearing pillar $\langle p_x, p_y, z_b \dots z_t \rangle$, there are two cases. Either L contains enough cubes to reach the origin ($|L| \geq x + y$), or L does not contain enough cubes to reach the origin ($|L| < x + y$). We call these low components *big* and *small* respectively and we handle them differently. For small low components, we would like to shove the clearing pillar. However, this might not be valid and might disconnect the configuration in the process. Therefore, we will devise a special operation that is only safe on small low components.

Small low components. If a clear low component is too small to reach the origin, we want to move the clearing pillar and do a pillar shove. However, it could be that moving the clearing pillar would disconnect the low component, or that there are cubes around the clearing pillar obstructing the shove. For both of these situations, we devise a new operation. Let N_P be the set of cells c with $z = 1$ in $B_{\mathcal{C}}$ neighboring P .

31:10 Optimal In-Place Compaction of Sliding Cubes

► **Lemma 11.** *Let \mathcal{C} be a configuration that does not admit operations of type (a–g). Let L be a clear component and $P = \langle x, y, z_b \dots z_t \rangle$ be its clearing pillar. Assume $z_t \geq 2$. The cubes in N_P need to be either all present, or all absent from \mathcal{C} .*

Proof. Assume for a contradiction that at least one, but at most three of the cubes neighboring P with $z = 1$ are present. Denote these cubes by c_1, c_2 , and c_3 ; not all need to be present. Since P is a clearing pillar, $(\mathcal{C} \setminus L) \setminus P$ is connected. If the cube $(x, y, 2)$ is completely surrounded by cubes, then it can do a potential reducing operation of type (b). Otherwise, the cube $(x, y, 2)$ can do a potential reducing operation of type (f). Both lead to a contradiction. ◀

Depending on if the cubes in N_P are present, we perform the following operations on L .

(h) Let L be a clear component and $P = \langle x, y, z_b \dots z_t \rangle$ be its clearing pillar. Assume at least one of the cubes in N_P is present. By definition, $L \neq R$. Therefore, there exists an empty cell e , with coordinates $(e_x, e_y, 0)$, with $e_x < x$ or $e_y < y$. Let e be such an empty cell with highest y , and from those, the one with highest x . We now want to take the cube $c = (x, y, 0)$ and move it on a shortest path via $z = -1$ towards e , reducing its potential. This however, could disconnect parts of L if c is a cut cube of L and if $z_t = 1$. (If $z_t \geq 2$, then by Lemma 11, all cubes in N_P are present and c is not a cut cube of the configuration.) In this case, we first gather cubes from L to fill the 3×3 horizontal square centered around c , making c a non-cut cube, before moving c towards e . Now the configuration stays connected when moving c to e . If we gathered cubes because $z_t = 1$, the head of P at $(x, y, 1)$ is not a cut cube, and can subsequently move down.

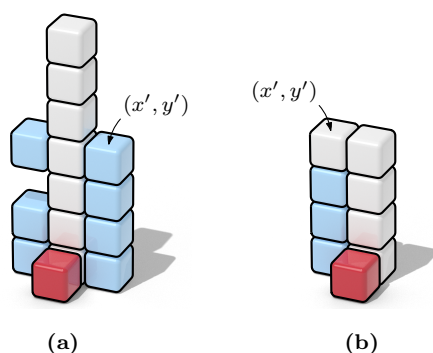
If (a–h) do not apply, then all cubes from N_P are not in \mathcal{C} .

(i) Let L be a clear component and $P = \langle x, y, z_b \dots z_t \rangle$ be its clearing pillar. All cubes in N_P are absent from \mathcal{C} . Gather cubes from L towards P according to Figure 5 and do a pillar shove on P . Then, move the extra cubes back to their original location.

The only reason that (e) is not possible, is that P is a cut pillar, since it is the only pillar connecting L to the other components. Therefore, gathering cubes from L to P makes the operations (h) and (i) viable. This is done in the following way. Let the clearing pillar of L be $P = \langle x, y, z_b \dots z_t \rangle$. Let z_t be the highest z such that only (x, y, z_t) has a horizontal neighboring cube (x', y', z_t) . Let $c = (x, y, 0) \in L$ be the cube below P . Assume that P has at least size 5. Then, repeatedly select the non-cut cube $c \in L$ that lexicographically maximizes (z, y, x) and move it towards P to fill the 3×3 square (for (h)), or create the configuration shown in Figure 5a (for (i)). The cubes that were gathered keep the configuration connected during the operation. For (i), if P has fewer than 5 cubes, we gather cubes towards P in a different way. Because P is too small to gather enough cubes for a pillar shove, we simply fill the cells that P would want to go towards, see Figure 5b. Then, the original P can be deconstructed. Again, using a constant number of moves, we can decrease the potential vector, while maintaining connectivity.

► **Lemma 12.** *Operations of type (h) and (i) are safe.*

Proof. For an operation of type (h) or (i), let $c = (c_x, c_y, c_z)$ be the head of P . We will show that any operation of type (i) strictly decreases P_C by $O(c_z + c_y + c_x)$ and uses $O(c_z + c_y + c_x)$ moves to do so. First we analyze the operations of type (h) or (i) with a pillar of size larger than one: the head $c = (c_x, c_y, c_z)$ of the pillar involved moves down from c_z to $z = 1$, so P_C reduces by $4(c_z - 1)$. The cubes beneath c from $z = 1$ up to c_z might increase their x - or y -coordinate by 1. Therefore, the potential also increases by $2(c_z - 1)$ at most. The cubes that are gathered and then returned do not move positions and therefore do not affect the



■ **Figure 5** The start configuration for a pillar shove for a clearing pillar. The white pillar is the clearing pillar. The red cube is part of L . The blue cubes are required and need to be gathered. (a) Clearing pillar of height at least 5. (b) The configuration for a pillar shove of height at most 4.

potential. Moreover, w_c becomes one lower, because c_z decreases from $c_z > 1$ to $c_z = 1$. In total, the potential P_C decreases by $2(c_z - 1) + c_x + c_y$. For operations of this type with a pillar of size one, the potential decreases by $c_z + c_y + c_x$, since the head moves from $z = 1$ to $z = 0$.

Now we will show that executing one of these operations, which reduces Π_C by $O(c_z + c_y + c_x)$, takes $O(c_z + c_y + c_x)$ moves and is therefore safe. Moving via a shortest path over a component with x cubes takes $O(x)$ moves. Gathering the seven cubes from L to the clearing pillar and moving them back takes at most $O(c_x + c_y)$ moves, since L is a small low component and has therefore size at most $O(c_x + c_y)$. Then, the normal pillar shove takes $O(c_z)$ moves. Hence, the total operation takes $O(c_z + c_y + c_x)$ moves and is safe. ◀

Big low components. If a low component L is big, that is, if it contains enough cubes to reach the origin from its clearing pillar, we want L to actually contain the origin. Performing operations of types (g) is not sufficient to achieve this, and operations (h) and (i) are only safe on small low components. To make L contain the origin, note that all of our operations not only work in 3D, but also in 2D when instead of prioritizing reducing the z -coordinate, we prioritize reducing the y -coordinate. We run the algorithm on L in 2D, with an additional constraint. We fix an arbitrary clearing pillar of L , and call the cube p of L below that clearing pillar its *pinned cube*. When executing the algorithm in 2D, we never move p . With minor changes, all of the lemmas above still hold in the presence of at most one pinned cube.

We abstract from L being a clear low component, and instead consider a component \mathcal{C} in which we disallow a single cube $p \in \mathcal{C}$ from moving. We again call this cube p the *pinned cube*. We adapt our algorithm for configurations without pinned cubes to one for configurations with pinned cubes as follows. Whenever we are looking for the next operation to perform, simply disregard any operation that would move the pinned cube. We cannot guarantee that this adapted algorithm results in a finished configuration, but for our purposes, it is sufficient to prove that it reaches the origin if it is big enough.

Recall that R is a vertex of \mathcal{LH}_C chosen as follows. If \mathcal{C} contains a low component that contains the origin, then let R be that low component. Otherwise, we choose R to be an arbitrary low component. If \mathcal{C} already contains the origin, the lemma trivially holds, so we would choose R to be an arbitrary low component. However, since there exists a pinned cube p , we need to be more careful with our choice and instead pick R as follows. If there exists a low component that either contains p or that neighbors a pillar containing p , let R be that low component. We are now ready to prove the following lemma.

31:12 Optimal In-Place Compaction of Sliding Cubes

► **Lemma 13.** *Let \mathcal{C} be a configuration with a single pinned cube $p = (p_x, p_y, p_z)$ and assume \mathcal{C} has at least $p_x + p_y + p_z$ cubes. If \mathcal{C} does not admit operations of type **(a-i)** that do not move p , then \mathcal{C} contains the origin.*

Proof. Assume there are at least two low components. Let L_1 and L_2 be the low components such that the distance between them in $\mathcal{LH}_\mathcal{C}$ is maximized. Hence, if there would be no pinned cube p , we could pick any of them and the other would be clear (Lemma 10). Because \mathcal{C} only contains a single pinned cube, we pick R to be either L_1 or L_2 such that the other one is clear. Therefore, there is always a clear low component. While a clear low component exists, we can execute operations **(g-i)** if it is small, or perform operations one dimension lower if it is big and does not contain the origin. This is a contradiction and therefore there can be at most one low component L .

Because no operations of type **(h)** or **(i)** are possible on L , its clearing pillar must contain p . Furthermore, no operations are possible on L in one dimension lower, so L must contain its own origin by recursion. Because there are no possible operations of type **(g)** on L and because it is big enough, this means that L also contains the global origin. ◀

If none of the operations **(a-g)** are possible, every clear low component either contains the origin, or is too small to do so.

The algorithm terminates when no clear low component (and hence only the root low component) remains. Recall that d is the maximum distance from the root R of $\mathcal{LH}_\mathcal{C}$ over all vertices. We are left with two cases. Either no high component remains ($d = 0$), or there is at most one high component ($d = 1$), which consists of entirely finished cubes.

As stated before, all operations **(a-i)** not only work in 3 dimensions, they also work in 2 dimensions when instead of prioritizing reducing the z -coordinate, we prioritize reducing the y -coordinate. Moreover, these operations never move the origin. Therefore, we can now run the exact same operations on the bottom layer in 2D, until the root component is finished. If there is still a high component, it stays connected via the origin. We end up with a finished configuration.

Running time. Recall that the *potential* of a cube $c = (c_x, c_y, c_z)$ is $\Pi_c = w_c(c_x + 2c_y + 4c_z)$, where the *weight* w_c depends on the coordinates of c in the following way. If $c_z > 1$, then $w_c = 5$, if $c_z = 1$, then $w_c = 4$. If $c_z = 0$, then w_c depends on c_y . If $c_y > 1$, then $w_c = 3$, if $c_y = 1$, then $w_c = 2$ and lastly, if both $c_z = c_y = 0$, then $w_c = 1$. The potential of the complete configuration is the sum of potential of the individual cubes. Moreover, a sequence of m moves is *safe* if the result is a configuration \mathcal{C}' inside $B_\mathcal{C}$, such that $\Pi_{\mathcal{C}'} < \Pi_\mathcal{C}$ and $m = O(\Pi_\mathcal{C} - \Pi_{\mathcal{C}'})$. Each operation of type **(a-i)** strictly reduces the potential function. Moreover, each of the operations **(a-g)** is trivially safe. We have shown that operations **(h)** and **(i)** are also safe (see Lemma 12).

Because all operations are safe and reduce the potential, the algorithm performs at most $O(\Pi_\mathcal{C}) = O(X_\mathcal{C} + Y_\mathcal{C} + Z_\mathcal{C})$ moves. For the problem of reconfiguring the cubes into a finished configuration, this is worst-case optimal. An example achieving this bound is a configuration consisting of a path of cubes in a bounding box of equal side lengths w tracing from the origin to the opposite corner of the bounding box. To see that, note that any finished cube at position (x, y, z) requires there to exist $n \geq x \cdot y \cdot z$ cubes, so at least one of x , y , and z is at most $n^{1/3}$ for any candidate finished position. There are $\Omega(n)$ cubes (x', y', z') that are initially $\Omega(w - n^{1/3}) = \Theta(n) = \Theta(x' + y' + z')$ away from any such potential finished position.

4 Conclusion

We presented an in-place algorithm that reconfigures any configuration of cubes into a compact canonical shape using a number of moves proportional to the sum of coordinates of the input cubes. This result is asymptotically optimal. However, just as many other algorithms in the literature, our bounds are amortized in the sense that we make use of a number of dedicated cubes which help other cubes move by establishing the necessary connectivity in their neighborhood. This is in particular the case with our pillar shoves, that need some additional cubes to gather at the pillar, to then move up and down the pillar to facilitate moves. These extra moves are charged to one cube in the pillar reducing its coordinates. In the literature such cubes are referred to as *helpers*, *seeds*, or even *musketees* [2, 7, 13, 17].

Such helping cubes are in many ways in conflict with the spirit of modular robot reconfiguration: ideally each module should be able to run the same program more or less independently, without some central control system sending helpers to those places where they are needed. The input-sensitive Gather&Compact algorithm in 2D by Akitaya et al. [4] does not require amortized analysis and gives a bound on the number of moves for each square in terms of the perimeters of the input and output configurations. The question hence arises whether it is possible to arrive at sum-of-coordinates bounds either in 2D or 3D without amortization? For example, is there a compaction algorithm in which each cube in the configuration that starts at position (x, y, z) performs at most $O(x + y + z + a)$ moves, where a is the average L_1 -distance that cubes lie from the origin?

References

- 1 Zachary Abel and Scott Duke Kominers. Universal reconfiguration of (hyper-)cubic robots. *arXiv e-Prints*, 2011. [arXiv:0802.3414v3](https://arxiv.org/abs/0802.3414v3).
- 2 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovic, Robin Y. Flatland, Matias Korman, Belén Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The $O(1)$ musketees. *Algorithmica*, 83(5):1316–1351, 2021. [doi:10.1007/S00453-020-00784-6](https://doi.org/10.1007/S00453-020-00784-6).
- 3 Hugo A. Akitaya, Erik D. Demaine, Andrei Gonczi, Dylan H. Hendrickson, Adam Hesterberg, Matias Korman, Oliver Korten, Jayson Lynch, Irene Parada, and Vera Sacristán. Characterizing universal reconfigurability of modular pivoting robots. In *Proc. 37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *LIPICs*, pages 10:1–10:20, 2021. [doi:10.4230/LIPICs.SoCG.2021.10](https://doi.org/10.4230/LIPICs.SoCG.2021.10).
- 4 Hugo A. Akitaya, Erik D. Demaine, Matias Korman, Irina Kostitsyna, Irene Parada, Willem Sonke, Bettina Speckmann, Ryuhei Uehara, and Jules Wulms. Compacting squares: Input-sensitive in-place reconfiguration of sliding squares. In *Proc. 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 227 of *LIPICs*, pages 4:1–4:19, 2022. [doi:10.4230/LIPICs.SWAT.2022.4](https://doi.org/10.4230/LIPICs.SWAT.2022.4).
- 5 Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O’Rourke, Val Pinciu, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhler. Efficient constant-velocity reconfiguration of crystalline robots. *Robotica*, 29(1):59–71, 2011. [doi:10.1017/S026357471000072X](https://doi.org/10.1017/S026357471000072X).
- 6 Greg Aloupis, Sébastien Collette, Mirela Damian, Erik D. Demaine, Robin Flatland, Stefan Langerman, Joseph O’Rourke, Suneeta Ramaswami, Vera Sacristán, and Stefanie Wuhler. Linear reconfiguration of cube-style modular robots. *Computational Geometry*, 42(6):652–663, 2009. [doi:10.1016/j.comgeo.2008.11.003](https://doi.org/10.1016/j.comgeo.2008.11.003).
- 7 Matthew Connor and Othon Michail. Centralised connectivity-preserving transformations by rotation: 3 musketees for all orthogonal convex shapes. In *Proc. 18th International Symposium on Algorithmics of Wireless Networks (ALGOSENSORS 2022)*, volume 13707 of *LNCS*, pages 60–76. Springer, 2022. [doi:10.1007/978-3-031-22050-0_5](https://doi.org/10.1007/978-3-031-22050-0_5).

- 8 Adrian Dumitrescu and János Pach. Pushing squares around. *Graphs and Combinatorics*, 22:37–50, 2006. doi:10.1007/s00373-005-0640-1.
- 9 Sándor P. Fekete, Phillip Keldenich, Ramin Kosfeld, Christian Rieck, and Christian Scheffer. Connected coordinated motion planning with bounded stretch. In *Proc. 32nd International Symposium on Algorithms and Computation (ISAAC 2021)*, volume 212 of *LIPICs*, pages 9:1–9:16, 2021. doi:10.4230/LIPICs.ISAAC.2021.9.
- 10 Daniel Feshbach and Cynthia Sung. Reconfiguring non-convex holes in pivoting modular cube robots. *IEEE Robotics and Automation Letters*, 6(4):6701–6708, 2021. doi:10.1109/LRA.2021.3095030.
- 11 Robert Fitch, Zack Butler, and Daniela Rus. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and System (IROS 2003)*, volume 3, pages 2460–2467, 2003. doi:10.1109/IROS.2003.1249239.
- 12 Ferran Hurtado, Enrique Molina, Suneeta Ramaswami, and Vera Sacristán. Distributed reconfiguration of 2D lattice-based modular robotic systems. *Autonomous Robots*, 38:383–413, 2015. doi:10.1007/s10514-015-9421-8.
- 13 Othon Michail, George Skretas, and Paul G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. In *Proc. 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *LIPICs*, pages 136:1–136:15, 2017. doi:10.4230/LIPICs.ICALP.2017.136.
- 14 Tillmann Miltzow, Irene Parada, Willem Sonke, Bettina Speckmann, and Jules Wolms. Hiding sliding cubes: Why reconfiguring modular robots is not easy. In *Proc. 36th International Symposium on Computational Geometry, (SoCG 2020, Media Exposition)*, volume 164 of *LIPICs*, pages 78:1–78:5, 2020. doi:10.4230/LIPICs.SOCG.2020.78.
- 15 Joel Moreno and Vera Sacristán. Reconfiguring sliding squares in-place by flooding. In *Proc. 36th European Workshop on Computational Geometry (EuroCG)*, pages 32:1–32:7, 2020.
- 16 Irene Parada, Vera Sacristán, and Rodrigo I. Silveira. A new meta-module design for efficient reconfiguration of modular robots. *Autonomous Robots*, 45(4):457–472, 2021. doi:10.1007/s10514-021-09977-6.
- 17 Frederick Stock, Hugo Akitaya, Matias Korman, Scott Kominers, and Zachary Abel. A universal in-place reconfiguration algorithm for sliding cube-shaped robots in quadratic time. In *Proc. 40th International Symposium on Computational Geometry (SoCG)*, 2024. To appear.
- 18 Cynthia R. Sung, James M. Bern, John Romanishin, and Daniela Rus. Reconfiguration planning for pivoting cube modular robots. In *Proc. IEEE International Conference on Robotics and Automation (ICRA 2015)*, pages 1933–1940, 2015. doi:10.1109/ICRA.2015.7139451.