Algorithmic Approaches to Reconfigurable Assembly Systems

Allan Costa
Department of Computer Science and Engineering
Massachusetts Institute of Technology
Cambridge, MA
allanc@mit.edu

Amira Abdel-Rahman
Center for Bits and Atoms
Massachusetts Institute of Technology
Cambridge, MA
amira.abdel-rahman@cba.mit.edu

Benjamin Jenett
Center for Bits and Atoms
Massachusetts Institute of Technology
Cambridge, MA
bej@mit.edu

Neil Gershenfeld
Center for Bits and Atoms
Massachusetts Institute of Technology
Cambridge, MA
neil.gershenfeld@cba.mit.edu

Irina Kostitsyna
Department of Mathematics and Computer Science
TU Eindhoven
Eindhoven, Netherlands
i.kostitsyna@tue.nl

Kenneth Cheung
Coded Structures Laboratory
NASA Ames Research Center
Moffett Field, CA
kenny@nasa.gov

Abstract—**Assembly of large scale structural systems in space is understood as critical to serving applications that cannot be deployed from a single launch. Recent literature proposes the use of discrete modular structures for in-space assembly and relatively small scale robotics that are able to modify and traverse the structure. This paper addresses the algorithmic problems in scaling reconfigurable space structures built through robotic construction, where reconfiguration is defined as the problem of transforming an initial structure into a different goal configuration. We analyze different algorithmic paradigms and present corresponding abstractions and graph formulations, examining specialized algorithms that consider discretized space and time steps. We then discuss fundamental design trades for different computational architectures, such as centralized versus distributed, and present two representative algorithms as concrete examples for comparison. We analyze how those algorithms achieve different objective functions and goals, such as minimization of total distance traveled, maximization of fault-tolerance, or minimization of total time spent in assembly. This is meant to offer an impression of algorithmic constraints on scalability of corresponding structural and robotic design. From this study, a set of recommendations is developed on where and when to use each paradigm, as well as implications for physical robotic and structural system design.**

1. Introduction

Automated, robotic processes for material deposition and manipulation have become central to state-of-the-art material and structural system production. Additive manufacturing can produce hierarchical, architected materials with nanometer-scale features on centimeter scale parts, resulting in novel properties unattainable with traditional engineering materials [1]. High performance, continuous fiber composite aircraft components can be made by automated tape laying, utilizing large gantries, tooling, and autoclaves [2]. However, even with the benefits of electronic digital control systems, some inherent limits exist due to the monolithic material system choice, including stochastic error detection and repair [3].

An alternative strategy is based on the assembly of discrete, building-block elements into larger functional structures, using reversible connections to allow for disassembly and reuse. These two properties, modularity and reconfigurability, have been appreciated throughout the history of engineered structures, and can be found in applications ranging from truss units for infrastructure to LEGO-style consumer products and toys. Modularity can reduce complexity, and reconfigurability can reduce material requirements for a variety of objectives. Combined, these attributes create significant potential for cost saving, adaptability, and scalability.

These benefits are particularly desirable in risk-averse, mass-constrained applications, such as space structures. In the relatively short history of space systems, we already see application of these principles in space stations and some large instrumentation [4], with the literature suggesting that further development of the approach may be key to building large scale infrastructure for long duration missions [5].

More recently, modular lattice elements with reversible connections have been assembled into structures with record setting mechanical properties [6], large-scale reconfigurability [7], and mission-adaptive performance [8]. These lattice elements, also termed voxels, can be mass produced with best-practice manufacturing, ensuring repeatability and affordability [9]. Further, it has been shown that simple, task-specific robots can be designed for assembly, inspection, and repair [10][11], and that the periodic, structured nature of the material system can be leveraged to reduce robot complexity. NASA's Automated Reconfigurable Mission Adaptive Digital Assembly Systems (ARMADAS) project [12] seeks to extend this material-robot system to the assembly of modular, reconfigurable space structures, as shown in Figure 1.

In this paper, we address the algorithmic challenges faced by a multi-robot system for reconfiguration of these modular, building-block structures. The use of discretized structures simplifies the algorithmic representation, and allows for physical error correction by discrete replacement. It is hoped that the use of discretized structures will also simplify the mechanical robotic assembly problem and demonstrate natural scalability.

U.S. Government work not protected by U.S. copyright

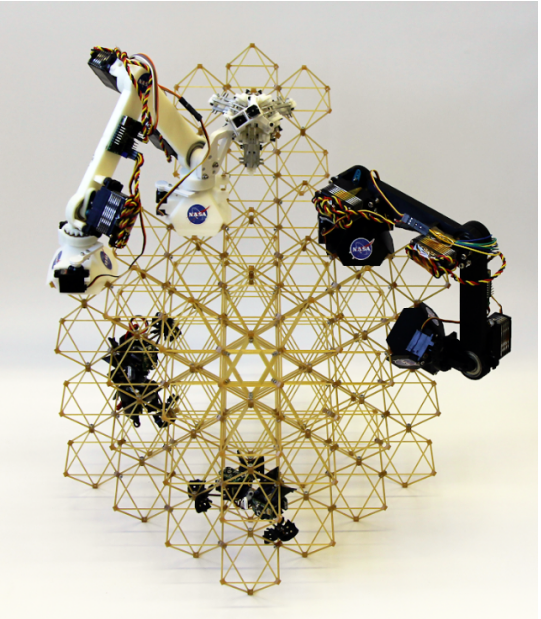

Figure 1. Modular reconfigurable 3D lattice structure [9] and mobile robots [10][11], showing the small size of the robots relative to the structure that they work on, and the parallel use of multiple robots.

We consider a system in which robots physically occupy a number of voxels' worth of spatial volume whether or not they are staying in one location or moving between adjacent locations. For the purposes of this paper, the robots are spatially constrained to the surfaces of the structure. Both of these constraints have computational implications for path planning and collision-detection. Finally, the structures are assumed to be three dimensional and isotropic in that assembly steps can proceed in any direction, we do assume, however, that the initial and final structure configurations have intersecting regions. The space that robots occupy necessarily changes during reconfiguration steps, and robots are generally presumed to travel between areas that are being disassembled and areas being constructed.

2. BACKGROUND

Mobile robotic fabrication is a popular topic, with many different approaches. We cover relevant prior art here, including material systems, sensing and positioning, construction and mobility algorithms, and hardware strategies.

One of the main appeals of mobile robotic fabrication is the ability to build structures larger than the build envelope of static robotic platforms. While large scale additive manufacturing has shown promise [13] [14], the size of the finished product is limited to the extents of the motion system. Mobile additive manufacturing has been tested at various scales [15] [16], and a key challenge is maintaining precision over long distances while utilizing a continuous, monolithic material system, ie: deposition of plastic, concrete, or metal.

An alternative approach is to use a discrete material system, ie: bricks, struts and nodes, or other forms of voxels. One main benefit of a discrete material system is its extensibility. Large scale structures can be decomposed into finite elements, and assuming a mobile robot can successfully place these elements, there should be no limit on scale. This assumption, however, relies on sensing, metrology, and localization.

Some systems use a global positioning system, such as a Vicon motion tracking system, to determine the location of the mobile robots and the state of the built object. This has been employed for aerial drone construction of brick [17] and strut and node systems [18], as well as wheeled, multi-DoF arms for coordinated assembly [19]. While this allows for arbitrary construction within the bounds of the global sensing system, it is ultimately limited to these bounds.

Other mobile robots leverage the modularity of the discrete construction system to achieve global precision from local metrology. Insect-inspired, wheeled robots can place passive bricks [20], or inchworm style platforms can traverse and manipulate active bricks [21]. Truss-based structures offer improved structural performance [22]. Our work builds on this approach, combining novel material properties for space applications.

We will now look at how mobile, discrete robotic construction systems use algorithms to address two main challenges: material deposition, or build, sequencing and robot path planning.

Within build sequencing, we can differentiate between local and global rules. Local rules describe where a block can or cannot be placed based on the configuration of neighboring blocks. The TERMES project, by Werfel et al, exemplifies this approach. Here, local rules are set up to avoid configurations where a subsequent block cannot be placed [23], or where a robot can not traverse the structure based on limited movements, ie: cliffs or canyons [24]. Globally, we seek to describe a desired geometry output in a language that can be executed by the robots. In TERMES, this was accomplished using an offline compiler which determined a "structpath", a one-way road which robots would enter and exit, depositing blocks, to achieve the final structure [20]. The Automatic Modular Assembly System (AMAS) applied both discrete and continuous gradient fields to simulate robots being guided from a material supply zone to a build front [25].

Path planning deals more directly with the motions of the robot. Objectives include decreasing total build time, optimizing multi-robot construction, and avoiding collision. Here we look at two different approaches to the system control architectures: centralized and distributed.

Centralized control architectures can be vulnerable to single points of failure, but also can achieve optimality more easily due to precomputing of a solution. This works for a relatively small number of robots with a relatively small number of parts, and has been implemented in several projects.

One distributed approach is biologically inspired by a variety of ant species that use stigmergy to deposit and follow pheromone trails [26]. These ants do not have a global view of their final destination, they only have a local view of the pheromone levels at each intersection, and by making small local decisions, they are able to achieve complex global goals. In the distributed model, we see use of gradient fields, or of a guided path, with local rules.

Lastly, in hardware, we see a number of options that relate to algorithm strategies. In AMAS, it is assumed that blocks can communicate [21]. This requires both mechanical and

electrical interconnects, which increases block complexity, but provides robust connectivity and avoid the need for wireless communication. In TERMES, blocks are passive, but markings are used by IR sensors on the robot for localization [27]. Typically, real-world experiments occur in a 1g environment, as opposed to microgravity. This affects how robots interact with the structure. TERMES relies on gravity, with continuous wheeled motion providing lateral motion, and the ability to climb up or down one block level. This simplifies hardware design, but results in a number of algorithmic constraints, such as the inability to climb vertical faces of a structure. Conversely, AMAS is designed to lock into any given face of the cubic blocks, which would allow for full 3D locomotion, though this was never fully physically implemented [21].

This paper is one of the very few papers that presents different path planning algorithms for the same system to compare the performance and optimality of each approach. The use of passive building blocks ensures the scalibity of the structures to build large and relatively inexpensive structurally sound systems. The algorithmic approaches discussed are not only able to build a larger family of shapes than similar robotic construction approaches [27], it is one of the first to discuss algorithms for the reconfiguration (parallel dissassembly and assembly) of structures from one arbitrary shape to another.

3. Model

We now describe the approach based on a discrete lattice material system and mobile robots. A key attribute of the system is the discretization of space by the use of finite voxels or building blocks. We also make use of a strong assumption of a reliable robot synchronization, whereby time is taken as discretized in time steps for the entire system.

We represent a (discrete) position in space with an integer vector and a boolean value that indicates the existence of a voxel in that position. We refer to the set of space cells that contain voxels as a *lattice*. We refer to the state of the lattice and of the robots at a given time step as a *configuration*, which we represent as $\Gamma(\cdot)$. We then denote $\Gamma(\vec{r})$ as the indicator variable for the existence of a voxel at position $\vec{r}$.

The example geometry that we employ here, for clarity, is a simple cubic arrangement of voxels. Each voxel has exactly six faces in which robots can possibly step. A face is said to be *exposed* if there is no voxel in the adjacent position. Robots operate only on the surface of the lattice, and move only between faces that are exposed.

A reconfiguration routine considers both initial Γ_0 and target Γ_f configurations. A region of space can be classified as part of assembly or disassembly if it is modified, an intersection, or void otherwise. Formally, given Γ_0 and Γ_f:

$$\text{type}[\vec{r}] = \begin{cases} \text{Assembly} & \text{if } \Gamma_f(r) - \Gamma_0(r) = 1, \\ \text{Disassembly} & \text{if } \Gamma_f(r) - \Gamma_0(r) = -1, \\ \text{Intersection} & \text{if } \Gamma_f(r) = \Gamma_0(r) = 1, \\ \text{Void} & \text{otherwise} \end{cases}$$

Here we only consider problems for which intersections exist. Such a requirement considerably reduces the complexity of the problem by limiting the space in which robots can operate. Any reconfiguration with empty intersections can be reduced into a series of subproblems with intersecting regions, which can be solved by the methods presented here. We also constrain the model to conserve total size: the size of assembly equals the size of disassembly - we do not consider voxel sources or sinks for clarity (though the experienced reader will see that minor modifications in these algorithms are required to accommodate those scenarios).

We refer to a voxel that can be added or removed, for a given time step t, as a *frontier*. These are defined by heuristics that can be mechanical (e.g., robotic or structural requirements for addition or removal) or computational (e.g., precomputed orderings).

We consider many of a single type of robot, each able to perform M moves. These moves include locomotion, voxel detachment, voxel attachment, giving or receiving a voxel from another robot, and staying stationary. Robots have these moves limited to the directions, tangential to the voxel face they occupy and along one of any Cartesian axial direction.

Every move is abstracted by a number of relative positions that must be free in order for the move to be possible (Figure 2), and similarly by positions that must have voxels. A move is considered to be possible only if those two conditions hold. Note that a robot that is stationary also has space requirements.

We refer to the set of moves executed by each robot at a given time step t as a *move step*. We refer to the time-ordered list of all move steps as a *move plan*, Π, which describes a reconfiguration together with an initial configuration. A configuration Γ is said to be *unreachable* or *unfeasible* from Γ_0 if there's no valid Π that is able to reach Γ starting from Γ_0. Note that $|\Pi| = T =$ *Assembly Time* is the number of time steps required for the reconfiguration.

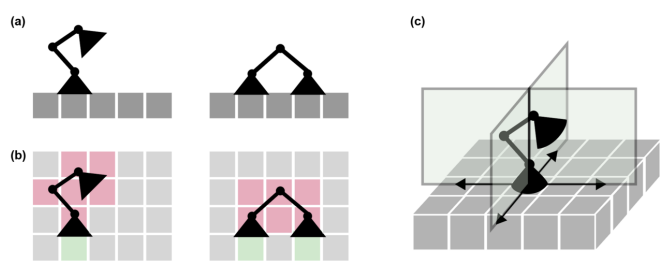

Figure 2. (a) Two possible moves that a robot can perform, steady and single step. (b) Those moves are only possible if the positions in red are free and the positions in green have voxels. (c) When moving, robots can choose from four possible directions.

4. Algorithm Selection

The objective function pursued here is to guarantee the reconfiguration $\Gamma_0 \rightarrow \Gamma_f$ while possibly minimizing total traveled distance, ensuring fault-tolerance, and/or proceeding in structurally optimal ways. We present two substantially different approaches, below.

We consider that centralized algorithms focus on building the move plan as a precomputed optimal solution, whereas distributed algorithms never have a single entity that holds or processes the complete move plan. For centralized algo-

rithms, we will describe different implementations of a state space search that minimizes assembly time.

Distributed and swarm algorithms have weak guarantees of optimality, and are never able to evaluate or parameterize a full state. In the extreme case of a cellular automaton, a finite set of rules describe how robots operate based on localized states. These algorithms are unaffected by the size of the reconfiguration. We describe an cellular automata-like algorithm in which robots use a precomputed map of gradient fields to make local decisions.

Centralized: State Space Search

In the State Space Search formulation, we consider the set S of all possible configurations achievable from Γ_0. If the space isn't bounded, $|S|$ is infinite. In the state exploration graph, there's an edge between nodes corresponding to Γ_i and Γ_j if and only if there is a valid move step that allows the transformation $\Gamma_i \rightarrow \Gamma_j$. The starting node corresponds to an initial configuration Γ_0 and the target node is a goal configuration Γ_f.

Choosing to use heuristics, we use $A^* search$ to explore the graph. This formulation allows encoding and minimization of a number of different objectives: weights can be added to move steps to avoid robotic actions that are costly, heuristic costs can be added to configurations that have undesirable mechanical properties, and/or probabilities of failures can be parameterized to minimize the chances of robotic mistakes. A resulting move plan Π can be considered to holistically consider all such parameters. In the implementation described here, we only consider heuristic costs that guide the search, but let edge costs be one in order to achieve minimum assembly time.

If a robot can stay in a voxel in D different ways, and if we bound the space by a cube of size n, the size of the graph is:

$$|V| = O((\mathrm{D}+1)^{n^3})$$

And there are

$$|E|/|V| = O(M^R)$$

outgoing edges from each node. It is hence crucial that we make good usage of time and memory space.

A number of heuristics are used to prune the exploration tree. Trivially, configurations in which voxels are being added to regions that are not assembly, or which has robots stepping out of bounds, are promptly pruned in the exploration.

An initial heuristic approach for A^* involved moving the center of mass of voxels that are removed to the center of mass of the assembly region, e.g. minimizing such distances. A later approach involved orienteering robots to gradient fields as in the distributed case. The fastest and most successful heuristic so far, however, involves computing the min-cost of a dynamic bipartite matching.

We describe this heuristic now (Figure 3). We denote the sets of robots without and with voxels R^0 and R^*, respectively. D and A denote the set of assembly and disassembly voxels, and DF and AF their respective fronts. Given a configuration, we build the corresponding bipartite graph $B : (U, V)$, where $U = R^* \cup D$ and $V = R^0 \cup A$. There's an edge connecting every node in R^* and AF, R^0 and DF, and A and D. Edge weights denote the (surface-based) distance between their

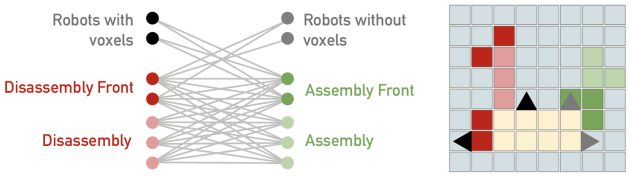

Figure 3. A bipartite graph as used to compute our heuristic, and a corresponding possible configuration.

corresponding positions. The value of our heuristic is then the min-cost max-matching of B.

As voxels are removed from the disassembly front, their corresponding nodes are removed from the bipartite graph. The same is done for voxels added in the assembly front. In the target configuration, the bipartite graph has the smallest number of nodes and no edges, and hence the min-cost max-matching reaches a minimum of zero. Trying to minimize this heuristic hence serves as a guide for our state exploration. It's important to note that a search might go through local minima such that the final solution might not be the optimal one if we don't explore an additional number of configurations in the A^* priority queue. Increasing the number of additional states we explore is required to finding an optimal solution, but also makes our algorithm computationally unfeasible as the number of robots increase. We hence upper bound this number for computational purposes, using state space search as an approximation algorithm for increasing problem size and number of robots.

For efficiency in implementation, we only store in program memory the initial and target configurations, as well as a small buffer with the top-scored configurations. In our implementation, an initial bipartite graph is built and dynamically updated as we explore the states. As we explore the graph, we save only the hash values of the explored configurations, and for edges we only keep the corresponding move steps. If we need to explore a configuration Γ that is not available in memory, we backtrack through the hash values to reconstruct the move plan that generates Γ through Γ_0.

Distributed: Gradient Fields

In this approach, we assume that there's no centralized authority able to compute the full solution online. Rather, robots have a precomputed map of the reconfiguration, and use both this map and local configuration to come up with a decision.

Computing these levels draws a comparison with physics: we assign positive charges to region of space in which voxels are to be added, and negative charges to regions of voxels to be removed. The field is computed over the surface of the structure (in opposition to the full 3D space), and the resulting field lines describe paths that robots are able to follow.

Robots can be visualized as charges following those lines. Interaction with other robots and the presence of local potential deadlocks induce robots to flip charges and navigate against the fields, adding to the complex behavior of assembly. The resulting movement searches for frontier voxels to be removed or frontier regions that can have a voxel being added. When the potential gradient doesn't point to a particular direction, the robot picks a direction at random. Finally, robots have a short memory buffer that optimizes for

exploration not to continue to regions that have been visited. The program halts when no robots carry voxels, and robots following lines towards disassembly can't find any voxels to be removed.

We implement this process in our system, assuming that inert voxels are unable to store information. A first step is needed where there is a global compiler that turns the voxel map into a discrete gradient field where each voxel face now has a value that indicates how close it is to the assembly and disassembly front. In the intersecting regions of the reconfiguration, a simple equation describes how the potential field is calculated:

$$\Phi(v) = \sum_{c \in \text{assembly}} \frac{1}{d(v,c)} - \sum_{c \in \text{disassembly}} \frac{1}{d(v,c)}$$

where $d(v, c)$ describes the breadth-first-search distance from voxel c to face v. The potential is considerably trickier in the assembly and disassembly regions, where the ordering of voxel additions and removals is crucial. We use different methods for computing those depending on the geometry of the reconfiguring shapes. Those methods include: ordering the potential in a frontier-based breadth-first-search of voxels from the intersecting regions; ordering from inside-out based on the geometrical skeleton of the structure; using values drawn from functional representations of the geometries, etc. We found that different methods of calculating those allow for different reconfigurations to be performed, while causing potential field deadlocks in other ones. Further work will explore finding one such heuristic able to generalize to a number of different complex geometries, or designing an algorithm to decide which heuristic to use given the geometry problem.

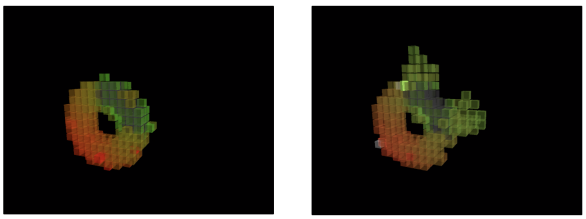

Figure 4. The distributed algorithm makes usage of complex gradient fields to guide robotic movement and action. Robots use local field information and can be compared charged particles moving under a force field.

A critical assumption to the correctness of our distributed algorithm is the capacity of robots to hand voxels to one another. It is often the case in our algorithm that robots carrying voxels and robots free of voxels traverse the same field lines in opposite directions; voxel exchange is critical for conflict resolution in this scenario, and when the number of robots increases, a bucket-brigade behavior starts to appear, which increases the voxel transport efficiency. For more complex conflict resolution and bottlenecks, we define a right-of-pass based on each robot's unique number identifier to break the symmetry. Robots that can't move stand still until they are given right-of-pass.

5. Implementation and Testing

The algorithms were implemented in software using Java. A class `Configuration` abstracts the state of voxels and robots, and classes `MoveStep` and `MovePlan` abstract changes in these states. Configurations can be encoded by either three dimensional bit matrices or by functional representations (frep) that describe voxel positions. This voxel map is used together with a list of robot positions to generate configurations.

A simulator interface is used to generate the behavior of robots under specific algorithms and conditions. This interface allow us to assign probabilities of failure to robots and voxels. A simulation outputs a file that discretely describes all the steps (including potential failures) of the robots until assembly is completed. We analyze the reconfiguration process through a visualizer that can be invoked on this file

We limit our tests to comparing state space search with distributed automata. We test our algorithms in 7 different scenarios, using 4 different shapes (Figure 5) with varying number of robots. The shapes and number of moved voxels are as follows: a) The rotation of a rectangle, 12 voxels. b) Linear translation of rectangle, 25 voxels. c) 2D sand timer. 50 voxels. This shape is used to test bottlenecks and funneling effects in structures. d) Translation of a large cube, 144 voxels.

We also impose different restrictions to robotic movement. Degrees of freedom of 2 are such that robots are limited to a single plane of movement, without depth. For example, we might impose this restriction to a robot in shape (a), so that it can only move in the border of the structure. This restriction is equivalent to requiring that the robot in Fig. 2 (c) be constrained to a single plane to move. This contrasts with degrees of freedom of value 3, in which both planes (i.e. all four directions) can be chosen, and robots can move over the whole surface of the structure.

Centralized algorithms have strong guarantees of optimality on structure assembly or total assembly time. This phenomena is apparent in figures 6 and 7, where the state search algorithm outperforms the distributed algorithm in terms of minimizing both the travel distance and assembly time, hence making the system less prone to failure. It is also apparent that the difference in performance between both methods also increases when the number of voxels, robots or degree of freedom increases. The state search total assembly time is 60% less than the distributed algorithm for test 1 and 230% less for test 7.

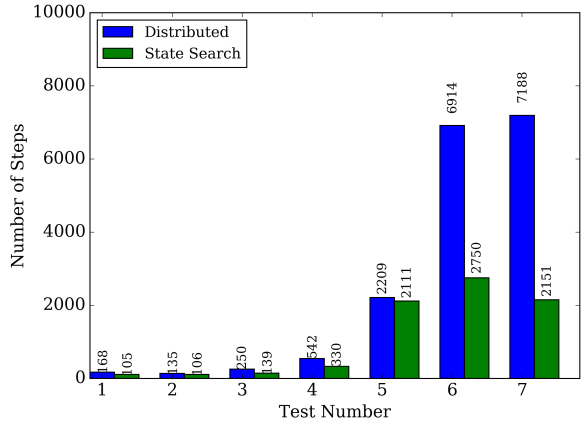

Figure 7. Comparison between the distributed and state search total distance traveled for the test cases.

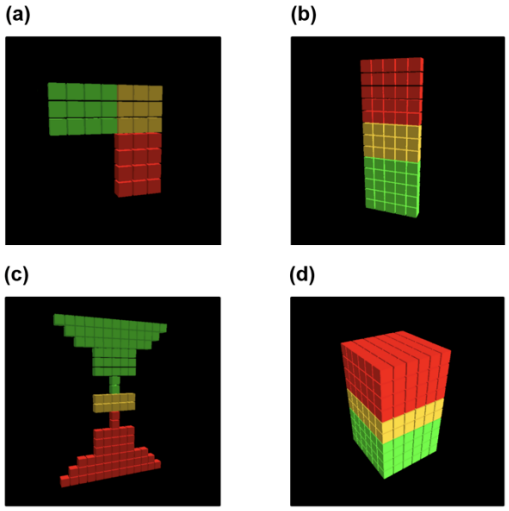

Figure 5. Reconfigurations used for comparing state space search and distributed algorithms. Yellow voxels indicate intersecting regions between the initial and target configurations. Green and red voxels indicate the assembly and disassembly regions, respectively.

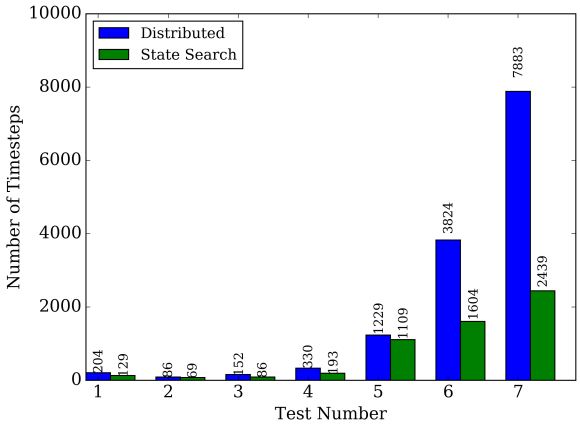

Figure 6. Comparison between the distributed and state search total elapsed time for the test cases.

Nevertheless, even if centralized algorithms often give us optimal solutions, since these solutions are precomputed, any robotic or structural failure will require further expensive recomputation. It is important to notice that when the size of the problem increases, fully centralized approaches become unfeasible and we turn to approximations for state space search. Moreover, we don't allow voxel exchange in our centralized approach, yet this is a critical assumption in the distributed case. These two factors explain figures 8 and 9 where, when the number of robots increases in relation to the number of voxels, the solutions that both solve complex collisions and perform better that the distribution algorithm become rare.

Distributed algorithms are unaffected by the size of the reconfiguration, and are generally oblivious to global situations such as robotic or structural failures. Figure 10 shows how the total assembly time for shape (b) drastically decreases when the number of robots increases while the total travel distance

Table 1. **Test Cases**

Description	1	2	3	4	5	6	7
Number of Translated Voxels	12	12	12	25	50	144	144
Reconfiguration Shape	(a)	(a)	(a)	(b)	(c)	(d)	(d)
Degrees of Freedom	3	3	2	3	2	3	3
Number of Robots	1	2	2	2	2	2	1

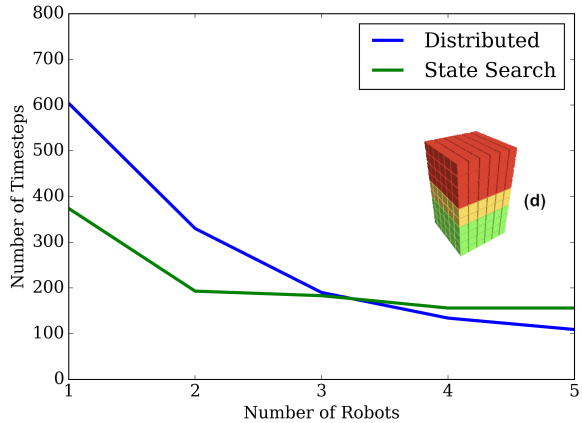

Figure 8. Comparison of total elapsed time between the distributed and the state algorithm on shape (d) for 1 to 5 randomly placed robots.

remains the same. However, when the density of robots increases, the systems becomes saturated and more collisions occur between the robots, increasing both the assembly time and total travelled distance. This phenomena counteracts the savings of adding extra robots, stabilizing the total assembly time in relation of number of robots.

6. Conclusion

From the conducted study, we were able to draw some conclusions regarding the feasibility and optimally of using a number of algorithms for the reconfiguration of discretely assembled structures. In general, it is important to note that the effectiveness of each algorithm is heavily dependent on the objective function, and there isn't a single absolute approach that would have the optimum performance under all conditions (Figure 11). For instance, failure rate is an important factor when trying to chose a control algorithm. For centralized systems, the precomputed solutions have to be reevaluated more frequently. A windowed precomputation minimizes risk of losing too much of a precomputed solution, but might result in solutions that are far from optimal. A high number of robots compared to number of voxels implies greater amounts of possible system deadlocks, and might require centralized approaches to find solutions. Centralized approaches were shown to perform considerably better than distributed automata in small reconfigurations. However, an increasing number of robots and voxels requires centralized approaches to be bounded and approximated, which can result in worse results from a greedy distributed approach. Still, even for those cases centralized approaches can be used for finding solutions that require holistic understanding of the system (e.g., structural requirements), compared to distributed. Further studies need to be done in mixed algorithms in which large scale distribution is combined with localized centralized computations.

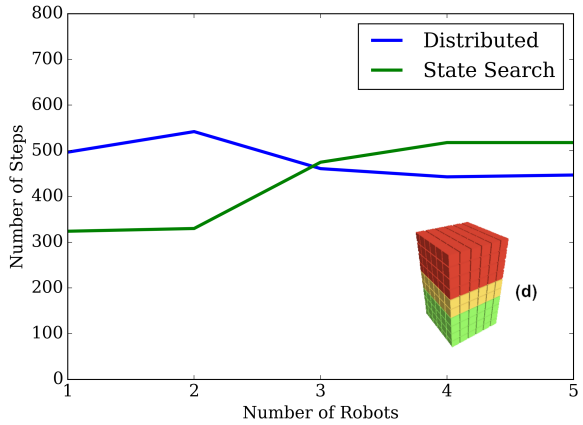

Figure 9. Comparison of total traveled distance between the distributed and the state algorithm on shape (d) for 1 to 5 randomly placed robots.

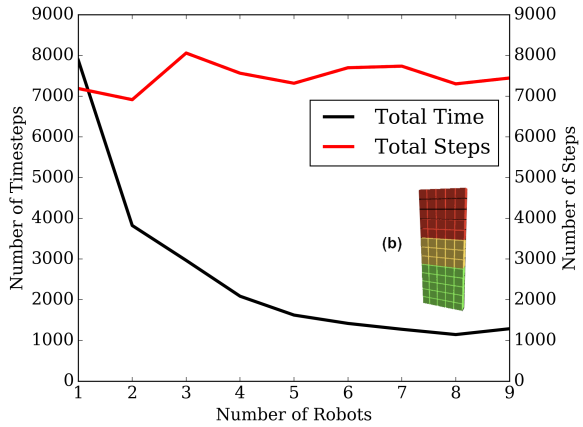

Figure 10. Total assembly time for the distributed algorithm on shape (b) for 1 to 8 randomly placed robots showing how the total time drastically decreases with time.

In conclusion, the paper pointed out and discussed the limitations of each algorithm in relation to different problem conditions. Finding an optimal algorithm is itself a multi-objective optimization problem for different geometries, robot densities and problem sizes, and we addressed variations of these situations with representative algorithms and test cases.

ACKNOWLEDGMENTS

This research was supported by the Automated Reconfigurable Mission Adaptive Digital Assembly Systems (ARMADAS) project through NASA STMD Game Changing Development (GCD) Program, the NASA Space Technology Research Fellowship (NSTRF) Grant NNX14AM40H, the NASA International Internship program, the MIT International Science and Technology Initiative (MISTI), and the Center for Bits and Atoms (CBA) consortia funding.

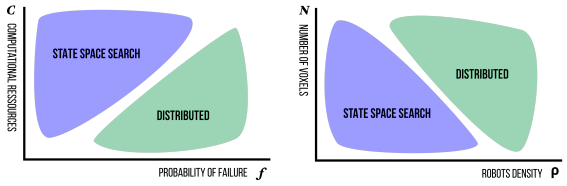

Figure 11. Conceptual Ashby-style plot of the optimum performance of each algorithm in relation to four problem parameters: computational resources, probability of failure, total number of voxels and robot density.

REFERENCES

[1] T. A. Schaedler and W. B. Carter, "Architected cellular materials," *Annual Review of Materials Research*, vol. 46, pp. 187–210, 2016.

[2] Z. August, G. Ostrander, J. Michasiow, and D. Hauber, "Recent developments in automated fiber placement of thermoplastic composites," *SAMPE J*, vol. 50, no. 2, pp. 30–37, 2014.

[3] D. Lévesque, C. Bescond, M. Lord, X. Cao, P. Wanjara, and J.-P. Monchalin, "Inspection of additive manufactured parts using laser ultrasonics," in *AIP Conference Proceedings*, vol. 1706, no. 1. AIP Publishing, 2016.

[4] W. K. Belvin, W. R. Doggett, J. J. Watson, J. T. Dorsey, J. E. Warren, T. C. Jones, E. E. Komendera, T. Mann, and L. M. Bowman, "In-space structural assembly: Applications and technology," in *3rd AIAA Spacecraft Structures Conference*, 2016, p. 2163.

[5] S. A. Jefferies, D. C. Arney, C. A. Jones, F. Stillwagen, P. Chai, C. D. Hutchinson, M. Stafford, R. Moses, J. A. Dempsey, E. Rodgers *et al.*, "In-space assembly capability assessment for potential human exploration and science applications," in *AIAA SPACE and Astronautics Forum and Exposition*, 2017, p. 5359.

[6] K. C. Cheung and N. Gershenfeld, "Reversibly assembled cellular composite materials," *science*, p. 1240889, 2013.

[7] B. Jenett, D. Cellucci, C. Gregg, and K. Cheung, "Meso-scale digital materials: modular, reconfigurable, lattice-based structures," in *ASME 2016 11th International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers, 2016, pp. V002T01A018–V002T01A018.

[8] B. Jenett, S. Calisch, D. Cellucci, N. Cramer, N. Gershenfeld, S. Swei, and K. C. Cheung, "Digital morphing wing: active wing shaping concept using composite lattice-based cellular structures," *Soft robotics*, vol. 4, no. 1, pp. 33–48, 2017.

[9] C. E. Gregg, J. H. Kim, and K. C. Cheung, "Ultra-light and scalable composite lattice materials," *Advanced Engineering Materials*, 2018.

[10] B. Jenett and K. Cheung, "Bill-e: Robotic platform for locomotion and manipulation of lightweight space structures," in *25th AIAA/AHS Adaptive Structures Conference*, 2017, p. 1876.

[11] B. Jenett and D. Cellucci, "A mobile robot for locomotion through a 3d periodic lattice environment," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5474–5479.

[12] K. Cheung, "Nasa gcd armadas," *https://gameon.nasa.gov/projects/automated-reconfigurable-mission-adaptive-digital-assembly-systems-armadas/*, Accessed 10/1/2018.

[13] C. E. Duty, V. Kunc, B. Compton, B. Post, D. Erdman, R. Smith, R. Lind, P. Lloyd, and L. Love, "Structure and mechanical behavior of big area additive manufacturing (baam) materials," *Rapid Prototyping Journal*, vol. 23, no. 1, pp. 181–189, 2017.

[14] B. Khoshnevis, D. Hwang, K.-T. Yao, Z. Yeh *et al.*, "Mega-scale fabrication by contour crafting," *International Journal of Industrial and Systems Engineering*, vol. 1, no. 3, pp. 301–320, 2006.

[15] S. J. Keating, J. C. Leland, L. Cai, and N. Oxman, "Toward site-specific and self-sufficient robotic fabrication on architectural scales," *Science Robotics*, vol. 2, no. 5, p. eaam8986, 2017.

[16] X. Zhang, M. Li, J. H. Lim, Y. Weng, Y. W. D. Tay, H. Pham, and Q.-C. Pham, "Large-scale 3d printing by a team of mobile robots," *Automation in Construction*, vol. 95, pp. 98–106, 2018.

[17] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems*, vol. 34, no. 4, pp. 46–64, 2014.

[18] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction with quadrotor teams," *Autonomous Robots*, vol. 33, no. 3, pp. 323–336, 2012.

[19] M. Dogar, R. A. Knepper, A. Spielberg, C. Choi, H. I. Christensen, and D. Rus, "Multi-scale assembly with robot teams," *The International Journal of Robotics Research*, vol. 34, no. 13, pp. 1645–1659, 2015.

[20] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[21] Y. Terada and S. Murata, "Automatic modular assembly system and its distributed control," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 445–462, 2008.

[22] F. Nigl, S. Li, J. Blum, and H. Lipson, "Autonomous truss reconfiguration and manipulation," *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/13, 2013.

[23] J. Werfel and R. Nagpal, "Three-dimensional construction with mobile robots and modular blocks," *The International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 463–479, 2008.

[24] J. K. Werfel, K. Petersen, and R. Nagpal, "Distributed multi-robot algorithms for the termes 3d collective construction system." Institute of Electrical and Electronics Engineers, 2011.

[25] Y. Terada and S. Murata, "Modular stucture assembly using blackboard path planning systems," in *International Symposium on Automation and Robotics in Construction*, 2006, pp. 852–857.

[26] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[27] J. Werfel and R. Nagpal, "Extended stigmergy in collective construction," *IEEE Intelligent Systems*, vol. 21, no. 2, pp. 20–28, 2006.

BIOGRAPHY

Allan Costa *is an undergraduate at the Massachusetts Institute of Technology, where he is pursuing his bachelor's degree in computer science engineering, and physics. He has worked with MIT's Center for Bits and Atoms and NASA Ames Research Center, and is currently researching abroad at ETH Zurich.*

Amira Abdel-Rahman *is a graduate student researcher at the Massachusetts Institute of Technology's Center for Bits and Atoms, where she is pursuing her Ph.D. on generative computational tools for discrete structures and materials. She holds a Master's degree from Harvard Graduate School of Design.*

Benjamin Jenett *is a graduate student researcher at the Massachusetts Institute of Technology's Center for Bits and Atoms, where he is pursuing his Ph.D. on automated assembly for large aerospace structures. He was formerly a NASA Space Technology Research Fellow and is currently in the NASA Pathways program.*

Irina Kostitsyna *is an assistant professor in the Applied Geometric Algorithms group in the Department of Mathematics and Computer Science. Her research interests lie in the field of computational geometry, both in its theoretical and applied aspects. She holds a Ph.D. from Stony Brook University.*

Neil Gershenfeld *is the Director of MITs Center for Bits and Atoms. Dr. Gershenfeld has a BA in Physics with High Honors from Swarthmore College, a Ph.D. in Applied Physics from Cornell University, honorary doctorates from Swarthmore College, Strathclyde University and the University of Antwerp.*

Kenneth Cheung *received his Ph.D. from the Center for Bits and Atoms at the Massachusetts Institute of Technology. He helps to run the ARC Coded Structures Laboratory (CSL), which conducts research on the application of building block based materials and algorithms to aeronautical and space systems.*