2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

Multi-Robot Assembly Sequencing via Discrete Optimization

Preston Culbertson, Saptarshi Bandyopadhyay, and Mac Schwager

***Abstract*— Multi-robot assembly has the potential to greatly reduce the cost and risk associated with the fabrication of large structures. Using teams of robots to perform assembly offers numerous advantages such as parallelism, robustness to single-agent failures, and flexibility in scheduling and task assignment. However, while previous work on multi-robot assembly focuses on generating feasible assembly plans and decentralized control strategies, we instead study the problem of planning optimal assembly sequences.**

To this end, we pose the problem of multi-robot assembly as a discrete optimization, specifically an integer linear program (ILP) or quadratic program (IQP), which aims to minimize the time to complete the assembly, or to minimize the distance traveled. We develop a model of multi-robot assembly that captures both geometric constraints and actuation constraints inherent to the problem. While the ILP and IQP can be solved exactly using commercial optimization software in a substantial amount of time, we also propose heuristic strategies which can be quickly computed, and can scale to structures of reasonable size. We also verify our methods empirically by comparing their performance on a variety of test structures.

I. Introduction

A. Motivation

Automated assembly remains one of the most important practical applications of robotics to date. Currently, robots are widely used in industrial environments, where they efficiently and reliably perform assembly operations to mass-produce commodities. However, while robotic assembly has transformed small-scale manufacturing, its benefits have not been brought to bear on large-scale assembly tasks such as the fabrication of buildings or infrastructure.

By leveraging recent advances in mobile robotics and multi-robot coordination, we hope to enable teams of robots to perform assembly on much larger scales, and in less structured environments, than is possible to date. Teams of mobile robots can work in parallel to perform multiple assembly operations at once, which can lead to large speedup in assembly time. Further, multi-robot assembly is inherently flexible, since *ad hoc* teams can form and dissolve as needed to perform various assembly and transport operations. Multi-robot teams are also robust to single-agent failures, ensuring assembly may continue despite individual robots requiring repair.

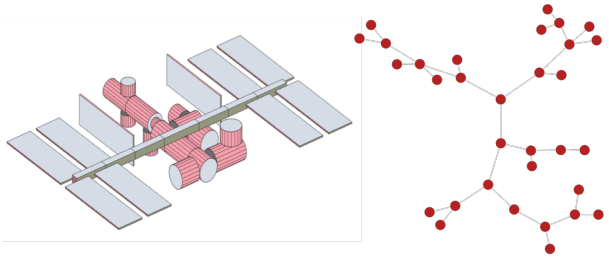


Fig. 1: Image of the International Space Station (ISS), and its assembly graph representation, which we use to plan multi-robot assembly sequences which minimize time and distance traveled during construction. We pose the planning problem as a discrete optimization, for which we propose heuristic strategies which can scale to structures of practical sizes such as the ISS.

By enabling multi-robot assembly of large structures, we hope to increase the safety of construction, which remains among the most dangerous occupations, while also reducing its cost and lead time. Further, there exist numerous environments inaccessible to humans where assembly tasks are crucial, such as the repair and fabrication of infrastructure in disaster environments, as well as the assembly of habitats and scientific instruments in space exploration.

B. Contribution

In this paper, we present a model of multi-robot assembly, in which a team of mobile robots collaborate to assemble a large structure. We pose the problem of assembly sequencing as an integer linear program (ILP) or integer quadratic program (IQP) which optimizes either the time to complete the assembly, or of the distance traveled by robots during construction, respectively. The program includes numerous geometric constraints to ensure planned assembly operations are physically feasible, as well as constraints which capture the limited actuation of individual assembly robots.

We argue that this approach is well-founded for planning multi-robot assembly of large, one-off structures such as buildings, since there exists plentiful "planning time" in which an assembly sequence may be computed and stored offboard, compared to the "execution time" which should be utilized as effectively as possible. A further advantage of this approach is its flexibility, since IPs are an expressive class of problems which can include logical and discrete variables. Thus, our model can be easily adapted to suit various construction scenarios through the addition of new variables and constraints.

We further leverage insights from our ILP and IQP to propose heuristic, optimal or near-optimal strategies for multi-robot assembly planning for a restricted class of structures. We also offer a proof of the optimality for one heuristic

P. Culbertson is with the Department of Mechanical Engineering, Stanford University, Stanford, CA, USA, pculbertson@stanford.edu.

S. Bandyopadhyay is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA, saptarshi.bandyopadhyay@jpl.nasa.gov.

M. Schwager is with the Department of Aeronautics and Astronautics, Stanford University, schwager@stanford.edu.

This research was supported in part by the NASA Space Technology Research Fellowship, Grant 80NSSC18K1180. Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

strategy in the free final time case. The performance of our exact solution, as well as our heuristic strategies, is evaluated empirically in a number of numerical examples.

II. Related Work

While (multi-)robotic assembly has been a fundamental problem for many years, there has been a recent surge of interest in distributed and decentralized approaches to robotic assembly. In [1], [2] the authors developed a termite-inspired swarm of ground robots which used local interaction rules to coordinate assembly of a large structure using tiles. This work is notable due to its limited assumptions on individual robot capabilities needed for assembly, as well as its use of techniques such as stigmergy to enable multi-robot coordination. In [3], the authors introduce a more efficient compiler, which uses posed the problem as a constraint satisfaction problem (CSP), and can scale up to structures with millions of bricks. Recently the authors in [4] proposed a reinforcement learning approach to this specific construction domain, in which they learn decentralized policies which aim to minimize the final time of assembly completion. While this approach is domain-specific, and provides few guarantees of its optimality or correctness, it shows an empirical speedup from the original algorithm in [2].

Other approaches to multi-robot construction include [5], where a team of quadrotors use Finite State Controllers to execute a centrally-computed construction plan for fabricating cubic structures from self-assembling beams. [6], [7] propose a distributed, Voronoi-based algorithm for partitioning construction and part delivery tasks for among a team of robots which build truss structures. Further, [8] uses a CAD model to plan assembly, delivery, and tooling schedules for fabricating IKEA furniture, which are then executed by teams of ground robots. The authors in [9] seek to use local force measurements to determine the stability and strength of intermediate assemblies, and to use this information to stabilize and reinforce structures online.

While these works span a wide range of building materials, model fidelity, and centralization, they mostly seek to generate feasible assembly sequences under the constraints of their specific assembly models. In contrast, we propose framing multi-robot assembly planning explicitly as a discrete optimization, which can be solved centrally and executed in a distributed fashion. This framing is especially appropriate for high-risk or high-cost applications such as on-orbit assembly, or automated assembly of disaster relief infrastructure.

There has also been extensive work in developing both hardware and materials suitable for performing robotic assembly. [10] presents a highly accurate system which can weld and assemble wire mesh structures which act as reinforcement for concrete walls. As an alternative to assembly of pre-fabricated parts, [11] proposes "contour crafting," a material-deposition approach to construction in which a robot deposits ceramic material in layers (as in 3D printing) to build macro-scale structures. Further, the authors of [12] argue that robotic assembly motivates the development of "digital materials" which are specialized to the specific challenges of autonomous construction, cheaply mass produced, and reconfigurable. [13] provides an excellent survey of the current trends in autonomous construction research, both in hardware and software.

Of special note is the large body of work on in-space construction, which is motivated by the high cost of material transport, and dangerous environments which preclude the use of human labor. [14] investigates the use of a multi-robot team to fabricate habitats for human astronauts on other bodies such as the Moon. Other approaches such as [15], [16] aim to perform in-orbit assembly using satellites equipped with 3D printers, which can manufacture components *in situ*.

Our work draws inspiration from the extensive work by the operations research and artificial intelligence communities on assembly planning for single-robot manipulators in manufacturing environments. The authors of [17] formulate a disassembly sequence as a traversal of an AND/OR graph, which can list all possible disassembly sequences, and be searched for optimal sequences using AO*, a variant of the A* algorithm. In [18], the author develops a non-directional blocking graph, which can be used to determine infinitessimally-feasible disassembly motions without requiring the exponential space of the AND/OR graph. [19] provides an excellent review of assembly planning methods, including the breadth of optimization techniques applied to the problem in literature.

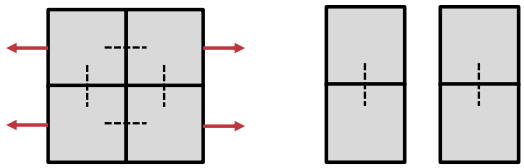
We note that in the assembly planning literature, the problem is to find feasible assembly sequences (via forward searches of large trees), whereas in this work we seek a feasible assembly sequence which is optimal with respect to some cost metric. We accomplish this by leveraging the recent advances in computing power and current availability of efficient commercial solvers for discrete optimization.

III. Model Description and Problem Formulation

The rest of the paper will proceed as follows. In Section III we present our model of multi-robot assembly, and pose the problem of assembly planning as a discrete optimization as an Integer Linear Program (ILP) or Integer Quadratic Program (IQP). In Section IV we provide technical details outlining our decision variables and constraints. Section V presents and analyzes heuristic strategies which can quickly generate feasible, near-optimal assembly sequences for many structures without solving the IP exactly. Finally, Section VI presents the results of our numerical experiments, which verify both the exact strategy resulting from our IP solution, as well as the performance of our heuristic strategies.

A. Model Definition

We will now outline our model of multi-robot assembly. We consider a set of N parts which must be assembled into the desired structure by a team of R robots. The final assembly is specified by a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with the vertex set $\mathcal{V} = \{1, 2, \ldots, N\}$ corresponding to the set of parts, and the edge set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, where edge $\{i, j\} \in \mathcal{E}$ iff there exists a mate between parts i and j in the final assembly. We also define the adjacency matrix $A(\mathcal{G})$, with $a_{ij} = 1$ if $\{i, j\} \in \mathcal{E}$, and zero otherwise. By definition, $a_{ii} = 0$ for all i, since parts may not be mated to themselves.



(a) (b)

Fig. 2: Parts being disassembled by moving along mating axes. Since parts are joined in a loop, no single part may be removed from this example; a subassembly must be created for disassembly.

Further, each mate $\{i, j\} \in \mathcal{E}$ has an associated direction D_{ij} along which part i must be moved to be attached to part j, as shown in Figure 2. We further let D be the number of unique mating directions in the assembly. While this model does not capture all possible attachment methods used in construction, it is a natural setting for parts which are connected via fasteners, snap fits, and press fits, and is expressive enough to capture geometries of sufficient interest for our work herein.

Finally, we assume the parts are assembled at one of S construction sites which are located at points $\mathbf{s}_i$ in $\mathbb{R}^n$ for either $n = 2$ or 3. At the beginning of the assembly process, parts are located in a "depot" located at the origin, from which robots may retrieve parts for assembly. Then, over each timestep of the construction process, robots grasp, transport, and join parts at various sites, until they complete the assembly. Figure 3 shows a schematic of the proposed construction layout.

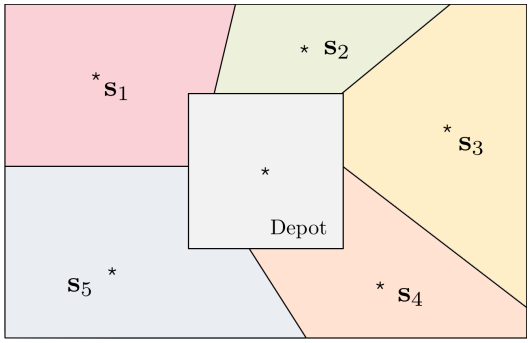


Fig. 3: Schematic of proposed construction site. Robots transport parts from the depot to assembly sites located at $\mathbf{s}_i$, where they join parts to create subassemblies.

Since we are interested in high-level construction planning, our system dynamics are highly discretized, where one timestep of our construction process is long enough for robots to grasp, transport, and attach parts. Further, our construction model is synchronous, meaning all robots operate on the same clock, performing only one construction operation per timestep.

In this paper, we formulate the planning problem as "assembly-by-disassembly," meaning we plan optimal disassembly sequences which are then reversed to give the optimal assembly plan. Since we consider the problem problem of deterministic, open-loop planning, the reversed disassembly sequence retains its optimality for assembly.

B. Problem Formulation

Given our assembly model and system architecture, we now formulate the assembly planning problems to be solved in this paper. While there exist a large number of feasible assembly sequences, we aim to optimize our assembly plan with respect to some cost function. In this work we consider two different costs of assembling a structure: final assembly time, and total distance traveled.

For most assembly problems, we aim to minimize final time, since we want the structure to be completed as quickly as possible. The benefits of multiple robots are obvious in this case, since using a team of robots enables construction tasks to occur in parallel, which can greatly speed construction. However, other construction tasks such as in-orbit assembly place low value on optimizing the final time of the construction process in favor of minimizing fuel usage.

1) Minimum-Time Assembly Planning: The minimum time objective can be captured using a set of binary variables $\mathbf{p} \in \mathbb{B}^T$, where $p_t = 1$ if there are any parts outside the depot at time t, and $\mathbb{B} = \{0, 1\}$. This can be expressed using the constraint

$$p_t = \vee_{i=1}^{N} \vee_{j=1}^{S} x_{i,j,t}, \tag{1}$$

where $\mathbf{x} \in \mathbb{B}^{N \times S \times T}$ are variable describing part placements, with $x_{i,j,t} = 1$ if part i is in location j at time t, and $\vee$ is the logical OR. We let $x_{i,0,t} = 1$ when part i is located at the depot at timestep t. This yields the cost function

$$J_{\text{time}} = \sum_{t=1}^{T} p_t. \tag{2}$$

We further note that the logical constraint (1) can be implemented using linear inequalities via a big-M formulation [20]. Thus, the minimum-time assembly problem is given by

$$\begin{aligned} J^*_{\text{time}} = \quad & \text{minimize} && \sum_{t=1}^{T} p_t, \\ & \text{subject to} && \text{geometric constraints (6)-(7),} \\ & && \text{movement constraints (8)-(9),} \\ & && \text{connectivity constraints (10)-(13),} \\ & && \text{cost constraint (1).} \end{aligned} \tag{3}$$

As we show in Section IV, all constraints are linear in the decision variables, which take integral values. Since we optimize a linear cost with linear constraints, (3) is an Integer Linear Program (ILP), which can be efficiently solved using commercial solvers.

2) Minimum-Travel Assembly Planning: When we seek to minimize the total distance traveled by parts during assembly, this cost may be captured by considering the matrix $\mathbf{L} = [\mathbf{0}, \mathbf{s}_1, \ldots, \mathbf{s}_S]$, i.e. the concatenation of the position vectors of the depot and assembly sites. The distance traveled by part i in time t is simply

$$||\mathbf{L}(\mathbf{x}_{i,:,t} - \mathbf{x}_{i,:,t+1})||_2,$$

where $||\cdot||_2$ denotes the ℓ_2-norm, and $\mathbf{x}_{i,:,t} \in \mathbb{B}^S$ is the binary vector of location variables corresponding to part i's position at time t.

Thus, to express our cost as the distance traveled by all parts, we use the objective

$$J_{\text{dist}} = \sum_{i=1}^{N}\sum_{t=1}^{T} ||\mathbf{L}(\mathbf{x}_{i,:,t} - \mathbf{x}_{i,:,t+1})||_2^2, \tag{4}$$

which is quadratic in the decision variables x. This yields the problem

$$\begin{aligned} J^*_{\text{dist}} = \quad & \text{minimize} \quad \sum_{i=1}^{N}\sum_{t=1}^{T} ||\mathbf{L}(\mathbf{x}_{i,:,t} - \mathbf{x}_{i,:,t+1})||_2^2, \\ & \text{subject to} \quad \text{geometric constraints (6)-(7)}, \\ & \qquad\qquad\ \ \text{movement constraints (8)-(9)}, \\ & \qquad\qquad\ \ \text{connectivity constraints (10)-(13)}. \end{aligned} \tag{5}$$

Since our objective is quadratic in our decision variables, and our constraints are linear, (5) is an Integer Quadratic Program (IQP), which again may be solved efficiently using a commercial solver.

IV. Constraint Definitions

We now describe in detail the constraints of the ILP and IQP defined above. Frequently used symbols are summarized in Table I below.

Symbol	Definition
$\mathcal{G}$	Graph of desired assembly
R	Number of robots
N	Number of parts
S	Number of assembly sites
T	Final timestep
$\mathbf{x}$	Part locations (Sec. III-B)
$\mathbf{a}$	Part adjacency (Sec. IV-A)
$\mathbf{d}$	Part directions (Sec. IV-A)
$\mathbf{b}$	Mate disconnections (Sec. IV-A)
$\mathbf{r}$	Robot assignments (Sec. IV-B)
$\mathbf{c}$	Part connectivity(Sec. IV-C)

TABLE I: List of frequently used symbols

A. Geometric Constraints

To track the assembly process at each timestep, we introduce a set of binary variables $\mathbf{a} \in \mathbb{B}^{N\times N\times T}$, where T is the index of the final timestep, and $\mathbb{B} = \{0, 1\}$, with $a_{i,j,t} = 1$ iff part i is directly mated to part j at timestep t. Since we plan by disassembly, the initial adjacency variables $A_0 = A(\mathcal{G})$, i.e. the adjacency matrix of the completed structure, and $A_T = 0$, i.e. a completely disassembled structure.

However, there exist geometric constraints on which parts may be detached at each timestep; specifically, parts must move away from each other along their mating axis to allow disassembly. Thus, we seek to constrain $a_{i,j,t} - a_{i,j,t+1} = 1$ iff part i can be removed from part j, or vice versa.

To accomplish this, we introduce binary variables $\mathbf{d} \in \mathbb{B}^{N\times 2D+1\times T-1}$, where $d_{i,k,t} = 1$ iff part i moves in direction k at time t. Since only one part need move to allow disassembly, we include a null direction D_0 to model parts which remain stationary during a disassembly step.

We thus introduce another variable $\mathbf{b} \in \mathbb{B}^{N\times N\times T-1}$, which indicates if the parts move in the correct relative directions to allow the mate $a_{i,j,t}$ to be disconnected. Logically, we can express this as

$$b_{i,j,t} := d_{i,D_{i,j,t}} \wedge (d_{j,0,t} \vee d_{j,D_{j,i,t}}) \tag{6}$$

meaning $b_{i,j,t} = 1$ iff part i moves away from part j along their docking axis and part j remains immobile or moves away from part i along the same axis. Further, using big-M, we can express this logical constraint as a series of linear constraints between $\mathbf{b}$ and $\mathbf{d}$. We can now enforce our geometric constraint on our transition dynamics by constraining

$$a_{i,j,t} - a_{i,j,t+1} \leq b_{i,j,t} + b_{j,i,t}, \tag{7}$$

for all i, j, t, meaning the mate between i and j may only be deactivated at time t if the parts move such that i can be removed from j or vice versa.

B. Movement Constraints

We first constrain parts located in the depot to be disconnected from all other parts (i.e. there can be no subassemblies in the depot). We achieve this by imposing the constraint

$$x_{i,0,t} \implies \sum_{j=1}^{N} a_{i,j,t} = 0, \quad \forall i, t, \tag{8}$$

where the implication constraint may again be implemented using linear inequalities via big-M.

Further, part movements are limited by the number of robots which we have available to perform manipulation and joining tasks. Since we assume our robots are much smaller than the total assembly, we model their actuation limits by requiring one robot to be attached to each part which changes locations in the next timestep. Thus, we require one robot per part which is taken to the depot; moving a subassembly of m parts requires a team of m robots.

We capture this mathematically using a set of binary variables $\mathbf{r} \in \mathbb{B}^{R\times N\times T-1}$, where $r_{m,n,t} = 1$ iff robot m is assigned to part n at time t. We can thus implement our actuation constraints by constraining

$$x_{i,j,t+1} - x_{i,j,t} \leq \sum_{k=1}^{R} r_{k,i,t} \tag{9}$$

for all i, t.

C. Connectivity Constraints

There further exist important constraints on our assembly path which depend on the connectivity of the assembly graph at each timestep. Specifically, since connected components of the assembly graph correspond to subassemblies, any parts belonging to the same connected component must have the same direction. Further, since we constrain each construction site to have at most one subassembly, we must constrain parts in different subassemblies at each time to occupy different construction sites.

We introduce a variable $\mathbf{c} \in \mathbb{B}^{N\times N\times T}$, with $c_{i,j,t} = 1$ iff there exists a path between parts i and j at time t.

Specifically, if we consider A_t to be an adjacency matrix corresponding to an intermediate assembly graph $\mathcal{G}_t = \{\mathcal{V}, \mathcal{E}_t\}$, where $\mathcal{E}_t \subset \mathcal{E}$ is the set of mates still active at time t, then c_t corresponds to the transitive closure of A_t.

The values of c_t can be determined from the adjacency matrix a_t at time t, using $c_t = (a_t + I)^n$, i.e. by repeatedly applying the adjacency matrix to consider all n-hop walks on the graph. But, this relation is nonlinear and thus intractible for many integer programming solvers. We can instead compute the transitive closure using intermediate variables and the Floyd-Warshall algorithm [21], which requires recursive applications of logical functions, which can be modeled with linear constraints.

We thus introduce the intermediate variable $\boldsymbol{\chi} \in \mathbb{B}^{N\times N\times N+1\times T}$, where $\chi_{i,j,k,t} = 1$ iff nodes i and j of $\mathcal{G}_t$ can be connected using intermediate nodes up to k, per the Floyd-Warshall algorithm. We impose the constraint

$$\chi_{i,j,0,t} = \begin{cases} a_{i,j,t}, & i \neq j \\ 1, & \text{otherwise}, \end{cases} \tag{10}$$

since directly connected nodes require no intermediate nodes for their connection, and nodes are self-connected by definition.

Further, each set of intermediate variables can be defined recursively, via

$$\chi_{i,j,k+1,t} := \chi_{i,j,k-1,t} \vee (\chi_{i,k,k-1,t} \wedge \chi_{k,j,k-1,t}), \tag{11}$$

which states that nodes i and j can be connected with intermediate nodes $\{1, \dots k\}$ iff they were connected using $k-1$ nodes, or if k acts as a separating node. Finally, we simply let $c_{i,j,t} = \chi_{i,j,N,t}$, which denotes all connected nodes which can use the full vertex set as intermediate nodes.

We note that this number of variables grows with N^3, which limits the tractability of the exact IP solutions in practice. However, these constraints are also crucial to the problem, since many important properties of assembly planning (e.g. actuation constraints, placement) hold at the subassembly level, which requires us to understand the connected components of the graph.

To constrain disconnected parts to occupy separate sites, we simply impose the constraint

$$x_{i,k,t} + x_{j,k,t} \leq 1 + c_{i,j,t} \tag{12}$$

for all i, j, k, t. Thus, if the nodes nodes i and j are disconnected at time t (i.e. $c_{i,j,t} = 0$), then only one of them may occupy each site k.

Finally, we must impose that connected components must move together; otherwise, parts could have arbitrary motions. We thus impose

$$d_{i,k,t} - d_{j,k,t} \leq 1 - c_{i,j,t+1}, \tag{13}$$

for all i, j, k, t.

V. Heuristic Strategies

As discussed previously, while the assembly sequencing problems of interest may be solved exactly by solving the IPs outlined above, the solution time scales poorly in the number of parts, which can cause the exact problem to become intractable even for moderately-sized structures ($N \approx 30$).

Thus, it is crucial to develop optimal or near-optimal heuristic strategies which can scale to structures with large numbers of parts. In this section, we propose two heuristic strategies for assembling a restricted class of structures, as well as a proof optimality for the minimum-distance, free final time case.

For both strategies, we consider the problem of building acyclic assemblies, i.e. assemblies whose assembly graphs contain no self-loops. While this is a significant restriction, we choose to study acyclic structures because they contain removable single parts at every timestep. Disassembling cycles, even over one timestep, requires the planner to choose feasible subassemblies for removal. This is itself a combinatorial problem subject to connectivity constraints, i.e. one of similar hardness to the full problem.

A. Greedy Disassembly

We first propose a heuristic strategy we term greedy disassembly, in which robots remove as many free parts as possible at each timestep. The algorithm is summarized in Alg. 1. Initially, the assembly is placed at the assembly location which is closest to the depot. Robots are then assigned to "leaf" parts in the structure, i.e. those with only one neighbor; assignment continues until there are no robots or free parts remaining. The robots then remove their assigned part, and leave it at the depot. The process continues until all parts have been disassembled.

Algorithm 1: Greedy algorithm for disassembly

```
function Greedy(s, R);
  input : List of assemblies s, and number of robots R
  output: Number of robots used robs, list of parts
          rem to be removed
initialize robs, rem;
for every assembly a in s do
    for every part p in a do
        if robs < R and neighbors(p) = 1 then
            append p to rem;
            robs ++;
        end
    end
end
```

Importantly, since we consider the disassembly problem for acyclic structures, this heuristic remains persistently feasible (i.e. there exists at least one removable part at each timestep), since an acyclic graph must have at least one vertex with one or fewer neighbors, and removing this node from the graph yields another acyclic graph. Further, since robots only remove single parts, there always exist feasible disassembly directions.

We term this strategy "greedy" as it optimizes the number of parts sent to the depot in the current timestep; it is myopic with respect to time, since assembly may proceed more

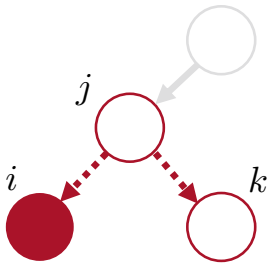


Fig. 4: Expansion step of fast disassembly. Upon expanding part i, its parent j, and all descendants (i.e. k) are added to the current subassembly. Since i is a leaf node, this expansion requires a free site and two additional robots. Part j is now added to the list of expansion candidates.

quickly by creating subassemblies which can be disassembled in parallel.

B. Fast Disassembly

We now propose a second heuristic strategy for minimum-time disassembly of acyclic structures, which aims to improve upon the myopic behavior of the greedy algorithm. The algorithm is summarized in Algs. 2 and 3. We term this strategy "fast" since it attempts to improve the greedy solution to render it less myopic with respect to time.

This algorithm leverages a number of insights about the disassembly problem to improve our performance. On one hand, when structures have low numbers of free parts relative to the number of robots, the myopic strategy results in poor utilization rates of the team. Thus, our algorithm aims to increase robot utilization by forming subassemblies.

However, creating large subassemblies can also be counterproductive. We thus constrain the number of parts per subassembly to be below the threshold Θ_i, for

$$\Theta_i = \left\lfloor \frac{|a_i|}{s(a_i) + m(a_i) + 1} \right\rfloor,$$

where $|a_i|$ is the number of parts contained in an assembly of interest a_i, $s(a_i)$ is the proposed number of single parts being taken from a_i, and $m(a_i)$ is the proposed number of subassemblies to be taken from a_i. Intuitively, if all subassemblies have size Θ_i, then a_i will be split into equal portions at the next timestep.

The algorithm works as follows: the policy is first initialized to the greedy solution, where we attempt to remove all free parts. If assembly robots remain idle after greedy assignment, the algorithm aims to create valid subassemblies by "growing" inward from the original free part. To accomplish this we maintain a list of "threads," candidate subassemblies which are iteratively expanded until assigning all robots are assigned, or until all assemblies cannot be expanded without exceeding the part size threshold.

Figure 4 shows a diagram of an "expansion step" for various candidate parts. We first direct the edges of the assembly graph such that, for each undirected edge $\{i, j\} \in \mathcal{E}$, we direct $i \to j$ if part j would be removed before part i under the greedy algorithm. On expanding a part, the part's parent, and all the parent's descendants, are added to the current subassembly. Since the parent part may have other subassemblies or singletons downstream, we update these lists on every expansion step. Subassemblies are assigned greedily to the closest free sites, in order of size.

Algorithm 2: Fast algorithm for disassembly

```
function Fast(s, R, S);
  input : List of assemblies s, number of robots R,
          and sites S
  output: List of parts to be removed singles, list
          of subassemblies to be created threads
robs, singles ← Greedy(s, R);
threads ← [ ], sites ← |s|;
while robs < R do
    robsPrev ← robs;
    for every p in singles do
        growSubassem(p);
    end
    for every t in threads do
        growSubassem(t);
    end
    if robs = robsPrev then break;
end
```

C. Analysis

We now offer analysis of our heuristics' performances on the problems presented in Section III-B. We first prove the optimality of greedy disassembly for the minimum-travel, free final time case, and then discuss the performance of our heuristic policies for the minimum-time problem.

Theorem 1 (Optimality of Greedy Disassembly)**.** *Greedy disassembly, as described in Alg. 1 is an optimal strategy for minimum-travel assembly (5) with free final time.*

Proof. Consider a relaxation of (5) which only imposes the boundary conditions and (8), which restricts connected parts from being placed in the depot. The relaxed problem has optimal cost $J^* = N||\mathbf{s}^*||_2^2$, where $\mathbf{s}^* = \arg\min_i ||\mathbf{s}_i||_2$, since each part must start at a non-depot site.

Now let us consider the cost of greedy disassembly in the free final time case. At each time t the robots will remove K_t free parts and transport them to the depot, incurring a stage cost of $K_t||\mathbf{s}^*||_2^2$, since the full assembly is located at the closest assembly site.

Since robots remove only single parts, there exist feasible removal directions for all parts, and since the graph is acyclic, there always exist free parts to be removed at each timestep. Finally, since the final time is unconstrained, the removal process may continue until some time T^*, when all parts have been disassembled.

Thus, the cost of performing greedy disassembly is $J^* = \sum_{t=1}^{T^*} K_t||\mathbf{s}^*||_2^2 = N||\mathbf{s}^*||_2^2 = J^*$. Ergo, our policy attains a cost equal to the optimal cost of the relaxed problem, while being feasible for the full planning problem. Thus, greedy removal is optimal for the minimum-travel, free final time case. □

However, while greedy disassembly is optimal for the free final time case, it can perform poorly in optimizing the

Algorithm 3: Subassembly expansion

```
function growSubassem(p);
  input : Index p of part to be expanded
a ← p.assem;
q ← p.parent;
thresh = floor(|parts(a)| / (|singles(a)|+|threads(a)|+1));
if p in singles then
    expand ← [(sites < S) ∧
     (robs + numAdded(q) ≤ R) ∧
     (size(q) ≤ thresh)];
    if expand then sites++;
else
    expand ← [(robs + numAdded(q) ≤ R) ∧
     (size(q) ≤ thresh)];
end
if expand then
    push q to threads;
    robs += numAdded(q);
    for every descendant d of q do
        if d in singles then remove d;
        if d in threads then
            remove d;
            sites --;
        end
    end
end
```

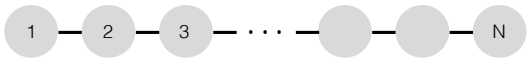


Fig. 5: Structure of N parts with chain topology.

final disassembly time. Consider a structure with a "chain" topology, i.e. all parts are mated sequentially along one axis, as shown in Figure 5. In this case, even with R, S arbitrarily large, the greedy policy will only remove two parts per timestep, resulting in a final disassembly time of $T^* = \left\lceil \frac{N}{2} \right\rceil$. In contrast, the fast policy will create subassemblies at each timestep to increase the number of free parts, as well as robot utilization. Further, as we show in Section VI, the fast policy empirically returns policies which are optimal or near-optimal for the minimum final time case.

VI. Numerical Experiments

In this section, we present numerical experiments which study the performance of both our ILP and IQP solutions, as well as the heuristic strategies proposed in Section V. All computations were performed onboard a Google Cloud virtual machine with 8 vCPUs and 30GB of memory. We implemented our integer programs in Julia 1.1 using JuMP [22], which we solved using IBM CPLEX 12.8. Our source code is available at http://www.github.com/pculbertson/ip-assembly-planning.

For our test cases, we considered five structures of varying topology and size: a nine-part planar lattice, a LEGO® model of the Hubble space telescope, the ISS, a chain of 50 parts, and a random tree of 500 parts. While these examples vary greatly in their scale and complexity, their joining mechanisms and geometry can be captured well using our assembly model. Figure 6 shows images of the structures, along with their assembly graph representations.

Tables II & III summarize the results of our numerical experiments below. We computed both exact and heuristic assembly plans for all three structures, and compare their objective values as well as computation time. A video visualizing both optimal and heuristic assembly plans is included with this submission.

method	Lattice	Hubble	ISS	Chain	Random
ILP	**4** (10^0s)	**5** (10^3s)	—	—	—
Greedy	—	7 (10^{-2}s)	9 (10^{-2}s)	26 (10^{-2}s)	32 (10^{-1}s)
Fast	—	7 (10^{-2}s)	9 (10^{-2}s)	11 (10^{-2}s)	30 (10^{-1}s)

TABLE II: Numerical results for minimum-time assembly sequencing. Table values are final times returned by each solution method, with optimal values in bold. Computation times are reported in parenthesis in seconds.

method	Lattice	Hubble	ISS	Chain	Random
IQP	**58** (10^2)	**20** (10^3)	—	—	—
Greedy	—	**20** (10^{-2})	**32** (10^{-2})	**50** (10^{-2})	**500** (10^{-2})
Fast	—	60 (10^{-2})	120 (10^{-2})	1079 (10^{-2})	1036 (10^{-2})

TABLE III: Numerical results for minimum-travel, free final time assembly sequencing. Table values are distance objectives returned by each solution method, with optimal values in bold. Computation times are reported in parenthesis in seconds.

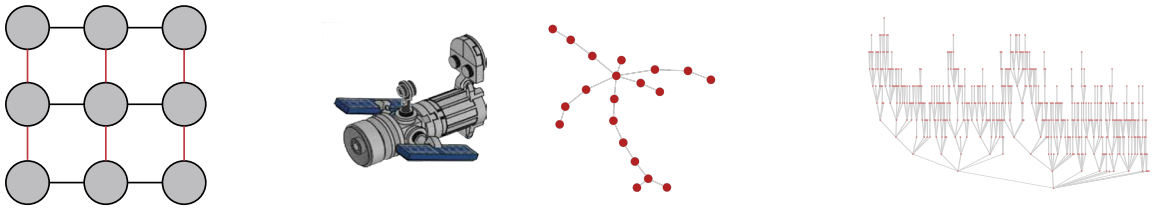
For each case, we let $S = 4$, with assembly sites located at $\mathbf{s_1} = [0, 1]$, $\mathbf{s_2} = [-1, -2]$, $\mathbf{s_3} = [-2, 2]$, and $\mathbf{s_4} = [2, -2]$. For the lattice and Hubble cases, we let R = 4, while we use $R = 6$ for the ISS, $R = 10$ for the chain case, and $R = 20$ for the random tree.

While we can compute optimal assembly plans via our IP solution, the number of integer variables scales poorly with the number of parts, causing the problem to become intractable as the assembly grows larger. Thus, the IP was unable to scale beyond the Hubble case, where $N = 20$. Further, while our heuristic solutions were easily computable, they are unable to handle cyclic structures (such as the lattice). We also report the distance traveled by the fast disassembly heuristic, to demonstrate that it incentivizes part travel to optimize the final time.

We also studied how additional robots can speed construction. Figure 7 plots the final time for both heuristics fabricating the chain, with $N = 50$. We note that the greedy heuristic cannot improve its performance after $R = 2$, due to the topology of the structure. On the other hand, the fast heuristic continues to improve its final time, albeit with diminishing returns. We note also the suboptimality of the fast heuristic, since the final time should be nonincreasing with respect to R. We believe slight increases in final time are due to the fast heuristic's optimism about the value of creating subassemblies.

VII. Conclusions & Future Work

In summary, we posed the problem of multi-robot assembly as a discrete optimization which captures both the geometric and actuation constraints inherent to the problem.



(a) Lattice, ($N = 9$) (b) LEGO® Hubble Telescope ($N = 20$) (c) Random Tree ($N = 500$)

Fig. 6: Images of structures studied in our numerical experiments, with their assembly graph representations, if different. Our test cases also included the ISS (Fig. 1), and a 50-part chain (Fig. 5).

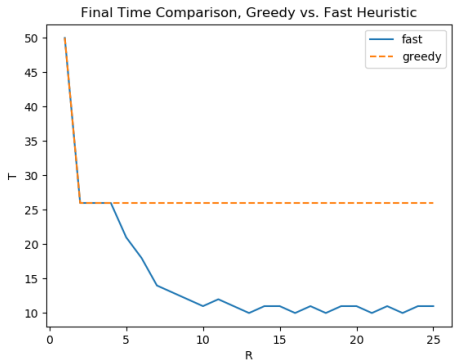


Fig. 7: Final time comparison of greedy and fast heuristics, for a chain of $N = 50$ parts, as a function of R. We note that adding robots can decrease the final time significantly while using the fast heuristic, while exhibiting diminishing returns.

We pose the problem as an ILP or IQP, which we solve exactly, as well as providing scalable heuristic strategies for acyclic structures. We verify our approach numerically by studying its performance on assembling a number of test structures.

This work offers numerous paths for future work. Our most immediate goal is to use our heuristic strategies to "warm start" the IP solver, to speed up the exact solution. We also aim to develop heuristics for cyclic structures which are significantly faster than the full IP. We are also interested in reformulating our connectivity constraints (12)-(13) without intermediate variables which grow in N^3. Finally, we are interested in developing decentralized and asynchronous assembly strategies, while also bounding their optimality with respect to the exact problem defined in this work.

VIII. Acknowledgements

The authors would like to thank Federico Rossi for his helpful discussions on integer programming.

References

[1] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, pp. 754–758, Feb. 2014.

[2] K. Petersen, R. Nagpal, and J. Werfel, "TERMES: An autonomous robotic system for three-dimensional collective construction," in *Robotics: Science and Systems VII*, Robotics: Science and Systems Foundation, June 2011.

[3] Y. Deng, Y. Hua, N. Napp, and K. Petersen, "Scalable compiler for the TERMES distributed assembly system," in *Distributed Autonomous Robotic Systems*, pp. 125–138, Springer International Publishing, 2019.

[4] G. Sartoretti, Y. Wu, W. Paivine, T. K. S. Kumar, S. Koenig, and H. Choset, "Distributed reinforcement learning for multi-robot decentralized collective construction," in *Distributed Autonomous Robotic Systems*, pp. 35–49, Springer International Publishing, 2019.

[5] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction with quadrotor teams," *Autonomous Robots*, vol. 33, pp. 323–336, June 2012.

[6] S. Yun, M. Schwager, and D. Rus, "Coordinating construction of truss structures using distributed equal-mass partitioning," in *Springer Tracts in Advanced Robotics*, pp. 607–623, Springer Berlin Heidelberg, 2011.

[7] D. Stein, T. R. Schoen, and D. Rus, "Constraint-aware coordinated construction of generic structures," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Sept. 2011.

[8] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "IkeaBot: An autonomous multi-robot coordinated furniture assembly system," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, May 2013.

[9] N. Melenbrink and J. Werfel, "Local force cues for strength and stability in a distributed robotic construction system," *Swarm Intelligence*, vol. 12, pp. 129–153, Nov. 2017.

[10] M. Lussi, T. Sandy, K. Dorfler, N. Hack, F. Gramazio, M. Kohler, and J. Buchli, "Accurate and adaptive in situ fabrication of an undulated wall using an on-board visual sensing system," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2018.

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

[12] N. Gershenfeld, M. Carney, B. Jenett, S. Calisch, and S. Wilson, "Macrofabrication with digital materials: Robotic assembly," *Architectural Design*, vol. 85, pp. 122–127, Sept. 2015.

[13] H. Ardiny, S. Witwicki, and F. Mondada, "Construction automation with autonomous mobile robots: A review," in *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, IEEE, Oct. 2015.

[14] A. Stroupe, A. Okon, M. Robinson, T. Huntsberger, H. Aghazarian, and E. Baumgartner, "Sustainable cooperative robotic technologies for human and robotic outpost infrastructure construction and maintenance," *Autonomous Robots*, vol. 20, pp. 113–123, Mar. 2006.

[15] R. P. Hoyt, "SpiderFab: An architecture for self-fabricating space systems," in *AIAA SPACE 2013 Conference and Exposition*, American Institute of Aeronautics and Astronautics, Sept. 2013.

[16] J. Kugler, J. Cherston, E. R. Joyce, P. Shestople, and M. P. Snyder, "Applications for the archinaut in space manufacturing and assembly capability," in *AIAA SPACE and Astronautics Forum and Exposition*, American Institute of Aeronautics and Astronautics, Sept. 2017.

[17] A. C. Sanderson, H. Zhang, and L. S. Homem de Mello, "Assembly sequence planning," *AI Mag.*, vol. 11, pp. 62–81, Apr. 1990.

[18] R. H. Wilson, *On Geometric Assembly Planning*. PhD thesis, Stanford University, Stanford, CA, USA, 1992. UMI Order No. GAX92-21686.

[19] P. Jiménez, "Survey on assembly sequencing: a combinatorial and geometrical perspective," *Journal of Intelligent Manufacturing*, vol. 24, pp. 235–250, Aug. 2011.

[20] J. P. Vielma, "Mixed integer linear programming formulation techniques," *SIAM Review*, vol. 57, pp. 3–57, Jan. 2015.

[21] S. Warshall, "A theorem on boolean matrices," *Journal of the ACM*, vol. 9, pp. 11–12, Jan. 1962.

[22] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, pp. 295–320, Jan. 2017.