Coordinating Construction of Truss Structures using Distributed Equal-mass Partitioning

Seung-kook Yun, Mac Schwager and Daniela Rus

Abstract This paper presents a decentralized algorithm for the coordinated assembly of 3D objects that consist of multiple types of parts, using a networked team of robots. We describe the algorithm and analyze its stability and adaptation properties. We instantiate the algorithm to building truss-like objects using rods and connectors. We implement the algorithm in simulation and show results for constructing 2D and 3D parts. Finally, we discuss briefly preliminary hardware results.

1 Introduction

We wish to develop cooperative robot systems for complex assembly tasks. A typical assembly scenario requires that parts of different types get delivered at the location where they are needed and incorporated into the structure to be assembled. We abstract this process with two operations: (1) tool and part delivery carried out by delivering robots, and (2) assembly carried out by assembling robots. In this paper, we consider how a team of robots will coordinate to achieve assembling the desired object. Tool and part delivery requires robots capable of accurate navigation between the part cache and the assembly location. Assembly requires robots capable of complex grasping and manipulation operations, perhaps using tools. Different assembling robots work in parallel on different subcomponents of the desired object. The delivering robots deliver parts (of different types) in parallel, according to the sequence in which they are needed at the different assembling stations. We consider the case where the parts are (a) rods of different lengths and (b) connectors for connecting the rods into truss-structured objects. The robots can communicate locally to neighbors. The delivering robots have the ability to find the correct part type in the part cache, pick it up, and deliver it to the correct spot for the assembling process requesting the part, and return to the part cache for the next round of deliveries. The

Seung-kook Yun, Mac Schwager and Daniela Rus
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA
e-mail: yunsk@mit.edu, schwager@mit.edu, rus@csail.mit.edu

assembling robots have the ability to receive the part from a delivering robot and incorporate it into the assembly.

We assume that the target object is given by a material-density function which encodes the object geometry and is known to all the robots. The construction process starts by a "coverage"-like process during which the assembling robots partition the target structure adaptively into sub-assemblies, such that each robot[1] is responsible for the completion of that section. To achieve this division, the robots locally compute a Voronoi partition, weighted by the mass of all the rods contained in the partition, and perform a gradient descent algorithm to balance the mass of the regions. The delivery robots also know the density function describing the target structure and the location of the parts. Each delivering robot carrying a part enters the assembling region and delivers the part to the region with the highest demanding mass. That is, the robot asks each assembling robot within communication range what is the current mass of the structure they have created and selects the site of the least completion. This ensures global and local balance for part delivery.

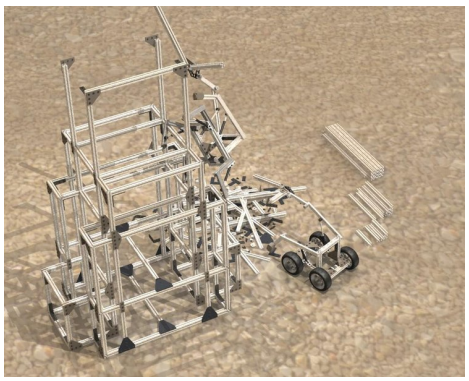


Fig. 1 Concept art for construction of a truss structure by mobile delivering robots and truss-climbing assembling robots. Reprinted with permission from Jonathan Hiller, Cornell University, USA.

We describe decentralized control algorithms for the partition, part delivery, and assembling steps. The algorithms are inspired by the approach in [2, 9, 7] and use *equal-mass partitioning* as the optimization criterion. The algorithms rely on local information only (e.g. neighbors exchange information about their local mass). The task allocation and part delivery algorithms are provably stable. They are adaptive to the number of delivering robots and assembling robots as well as to the amount of source material. We implemented these algorithms in simulation. Several 2D and 3D truss-structures were created using our algorithms. We have started a hardware implementation using iCreate robots extended with Meraki communication and a CrustCrawler 4-dof robot arm. The part delivery algorithm has been implemented on these robots to demonstrate the coordination infrastructure of the system and the correctness of the delivery algorithm. Assembly execution is under development.

1.1 Related Work

This work combines distributed coverage and robotic construction. We follow the notion of locational optimization developed by Cortes et al. [2], who introduced distributed coverage with mobile robots. The same optimization criteria was used in a distributed coverage controller for real-time tracking by Pimenta et al. [8]. In our previous work, Schwager [9] used adaptive coverage control in which networked

[1] The robot represents all the skills needed for each required assembly step; in some cases multiple robots will be needed, for example the connection of two rods with a screw is done by three robots, one robot holding each rod, and one robot placing the connector.

robots learn a sensory function while they are controlled for the locational optimization. This research inherits the distributed coverage concept, and pursues *equal-mass partitioning* in which every networked robot is controlled to have the same amount of construction (in our case, truss elements and connectors) to be built, rather than optimal sensing locations. Pavone et al. [7] have been independently working on equitable partitioning by the power diagram.

Algorithms and hardware have been developed for manipulator robots that climb and build a truss structure. SM^2, a truss-walking inspection robot, was developed for space station trusses [6]. Skyworker demonstrated truss-like assembly tasks [10]. Werfel et al. [11] introduced a 3D construction algorithm for modular blocks. Our previous work on truss assembling robots includes Shady3D [4, 5, 1] that utilizes a passive bar with active communication and may include itself in a truss structure, and is controlled by locally optimized algorithms. We also proposed a centralized optimal algorithm to reconfigure a given truss structure to a target structure [3]. This work introduces a framework in which robots are specialized as delivery and assembling robots, distributed algorithms control the assembly of a structure with multiple kinds of source materials.

2 Problem Formulation

We are given a team of robots, n of which are specialized as part delivering robots and the rest are specialized as assembling robots. The robots can communicate locally with other robots within their communication range. The robots are given a target shape represented as a mass-density function ϕ_t. We wish to develop a decentralized algorithm that coordinates the robot team to deliver parts so that the goal assembly can be completed with maximum parallelism.

Suppose for now that the robots move *freely* in an Euclidean space (2D and 3D). This assumption makes sense when the robots move in the plane to achieve a planar assembly. However, for 3D assemblies, factors such as gravity and connectivity of structure, as well as 3D motion for the robots, must be considered. We will generalize in Section 5.

Algorithm 1 shows the main flow of construction in a *centralized* view. In the first phase, assembling robots spread in a convex and bounded target area $Q \subset \mathbb{R}^N (N = 2, 3)$ which includes the target structure. They find placements using a distributed coverage controller which assigns to each robot areas of the target structure that have approximately the same assembly complexity. In the second phase the delivering robots move back and forth to carry source components to the assembling robots. They deliver their components to the assembling robot with maximum *demanding mass*. The demanding mass is defined as the amount of a source component required for an assembling robot to complete its substructure. In this work, the source components include two types: truss elements and connectors. The truss elements are rods and they may be of different lengths. Details of the demanding mass for each type of the source components are presented in Section 4.1. After an assembling robot obtains a component from a delivering robot, it determines the optimal placement for this component in the overall assembly and moves there to assemble the component. The assembly phase continues until there is no source component left or the assembly structure is complete.

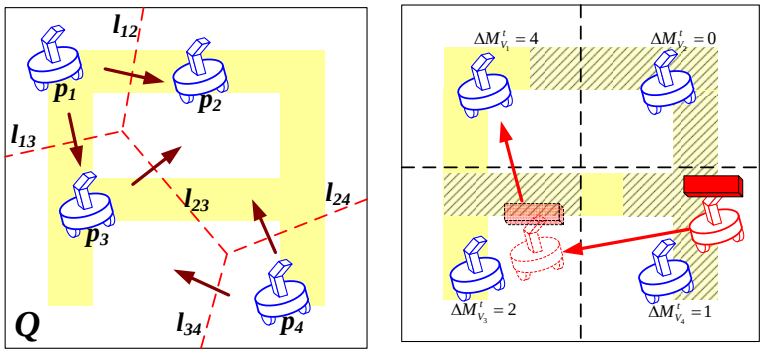
Algorithm 1 Construction Algorithm

1: Deploy the assembling robots in Q
2: Place the assembling robots at optimal task locations in Q (Section 3)
3: **repeat**
4: **delivering robots:** carry source components to the assembling robots (Section 4.2)
5: **assembling robots:** assemble the delivered components (Section 4.1)
6: **until** task completed *or* out of parts

2.1 Example

Figure 2.1(a) shows a construction system with 4 assembling robots. Intuitively, robot 1 and robot 4 move towards the other robots in order to expand their partition, whereas robot 2 moves away from the other robots because it has the largest area. The moving direction of the robots is determined by combining the normals to the Voronoi edges. Figure 2.1(b) shows the red delivering robot carrying a red truss element driven by the gradient of the demanding mass. The yellow region denotes the target density function ϕ_t. The hashed region denotes completed assembly. The demanding mass of a region can be thought of as the difference between the area of yellow regions and the area of hashed regions.

Suppose a delivering robot is in the region of robot 4. Among its neighbors (robot 2 and 3) the maximum demanding mass is with robot 3. Thus the delivering robot moves to robot3. The delivering robot finds that robot 1 has the maximum demanding mass among robot 3's neighbors, therefore it advances to robot 1 and delivers the truss component. Following the maximum demanding mass gives a local balance for the target structure.



(a) (b)

Fig. 2 Example of the equal-mass partitioning and delivery by the gradient of the demanding mass. 4 mobile manipulators (assembly robots) are displayed in a convex region Q that includes the A-shaped target structure. The yellow region has high density ϕ_t. The mass of a robot is the size of the total yellow region in its partition (Voronoi region.) $p_i (i = 1, 2, 3)$ denotes the position of the assembling robots and the red-dotted lines l_{ij} are shared boundaries of the partitions between two robots. $\Delta M^t_{V_i}$ is the demanding mass.

3 Task Allocation by Coverage with Equal-mass Partitions

This section describes a decentralized equal-mass partitioning controller which is inspired by distributed coverage control [2, 9]. The algorithm allocates to each assembling robot the same amount of assembly work, which is encoded as the same number of truss elements. This condition ensures maximum parallelism. We continue with a review of the key notation in distributed coverage, then give the mass optimization criteria and end the section with the decentralized controller.

3.1 Equal-mass partitioning

Suppose n assembling robots cover region Q with a configuration $\{\mathbf{p}_1, ..., \mathbf{p}_n\}$, where $\mathbf{p}_i$ is the position vector of the i^{th} robot. Given a point $\mathbf{q}$ in Q, the nearest robot to $\mathbf{q}$ will execute the assembly task at $\mathbf{q}$. Each robot is allocated the assembly task that included its Voronoi partition V_i in Q.

$$V_i = \{\mathbf{q} \in Q | \, \|\mathbf{q} - \mathbf{p}_i\| \leq \|\mathbf{q} - \mathbf{p}_j\| \, , \forall j \neq i\} \tag{1}$$

The target density function ϕ_t is the density of truss elements, and it is fixed during the construction phase. Given V_i, we define its mass property as the integral of the target density function in the area.

$$M_{V_i} = \int_{V_i} \phi_t(\mathbf{q}) d\mathbf{q} \tag{2}$$

We wish for each robot to have the same amount of assembly work. We call this *equal-mass partitioning*. The cost function can be modeled as the product of all the masses:

$$\mathcal{H} = \mathcal{H}_0 - \prod_{i=1}^{n} M_{V_i}, \tag{3}$$

where $\mathcal{H}_0$ is a constant and the bound of the product term as:

$$\mathcal{H}_0 = \left(\frac{1}{n} \sum_{i=1}^{n} M_{V_i} \right)^n = \left(\frac{1}{n} \int_Q \phi_t(\mathbf{q}) d\mathbf{q} \right)^n. \tag{4}$$

The cost function is continuously differentiable since each M_{V_i} is continuously differentiable [8]. Minimizing this cost function leads to equal-mass partitioning, because of the relationship between the arithmetic mean and the geometric mean.

$$\frac{1}{n} \sum_{i=1}^{n} M_{V_i} \geq \sqrt[n]{\prod_{i=1}^{n} M_{V_i}}, \tag{5}$$

where the equality holds only if all the terms are the same. Therefore the prefect equal-mass partitioning makes the cost function zero. Using the cost function in

(5), we have developed a decentralized controller that guarantees $\mathcal{H}$ converges to a local minimum.

3.2 Controller with Guaranteed Convergence

We wish for the controller to continuously decrease the cost function: $\dot{\mathcal{H}} \leq 0, t > 0$. Differentiating $\mathcal{H}$ yields

$$\dot{\mathcal{H}} = \sum_{i=1}^{n} \frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} \dot{\mathbf{p}}_i. \tag{6}$$

When $\mathcal{N}_i$ is a set of neighbor robots of the i^{th} robot, each term of the partial derivatives is

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = - \sum_{j=i,\mathcal{N}_i} \frac{\partial M_{V_j}}{\partial \mathbf{p}_i} \prod_{k=\{1,\ldots,n\},k\neq j} M_{V_k} \tag{7}$$

$$= - \prod_{l\notin\{i,\mathcal{N}_i\}} M_{V_l} \sum_{j=i,\mathcal{N}_i} \frac{\partial M_{V_j}}{\partial \mathbf{p}_i} \prod_{k\in\{i,\mathcal{N}_i\},k\neq j}^{n} M_{V_k} \tag{8}$$

where

$$\frac{\partial M_{V_i}}{\partial \mathbf{p}_i} = \sum_{j\in\mathcal{N}_i} \mathcal{M}_{ij}, \quad \frac{\partial M_{V_j}}{\partial \mathbf{p}_i} = -\mathcal{M}_{ij} \tag{9}$$

$\mathcal{M}_{ij}$ is computed along the sharing edges (sharing faces in 3D) l_{ij} between V_i and V_j as in [8]:

$$\mathcal{M}_{ij} = \int_{l_{ij}} \phi_t(\mathbf{q}) \frac{\partial \mathbf{q}_{l_{ij}}}{\partial \mathbf{p}_i} \mathbf{n}_{l_{ij}} d\mathbf{q} = \int_{l_{ij}} \phi_t(\mathbf{q}) \frac{\mathbf{q}-\mathbf{p}_i}{\|\mathbf{p}_i-\mathbf{p}_j\|} d\mathbf{q} \tag{10}$$

where $\mathbf{n}_{l_{ij}}$ is a normal vector to l_{ij} as

$$l_{ij} = V_i \cap V_j, \quad \mathbf{n}_{l_{ij}} = \frac{\mathbf{p}_j-\mathbf{p}_i}{\|\mathbf{p}_i-\mathbf{p}_j\|}. \tag{11}$$

We can rewrite equation 6 as

$$\dot{\mathcal{H}} = -\sum_{i=1}^{n} \prod_{l\notin\{i,\mathcal{N}_i\}} M_{V_l} \sum_{j=i,\mathcal{N}_i} \frac{\partial M_{V_j}}{\partial \mathbf{p}_i} \prod_{k\in\{i,\mathcal{N}_i\},k\neq j}^{n} M_{V_k} \dot{\mathbf{p}}_i \tag{12}$$

Let $\boldsymbol{J}_i$ denote the part of the partial derivative term $\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i}$ which is related with the set $\{i,\mathcal{N}_i\}$.

$$\boldsymbol{J}_i = \sum_{j=i,\mathcal{N}_i} \frac{\partial M_{V_j}}{\partial \mathbf{p}_i} \prod_{k\in\{i,\mathcal{N}_i\},k\neq j}^{n} M_{V_k} \tag{13}$$

Note that $\boldsymbol{J}_i$ is a vector. Given a velocity control for each robot, the decentralized controller that achieves task allocation is given by the control law:

$$\dot{\mathbf{p}}_i = k\frac{\boldsymbol{J}_i}{\|\boldsymbol{J}_i\|^2 + \lambda^2}, \tag{14}$$

where k is a positive control gain and λ is a constant to stabilize the controller even around singularities where $\|\boldsymbol{J}_i\|^2 = 0$.

Note that all the equations can be computed in a distributed way, since they only depend on the variables of the neighboring robots.

Theorem 1. *The proposed controller guarantees that $\mathcal{H}$ converges to either a local maximum or a global maximum.*

Proof. The proposed control input $\dot{\mathbf{p}}_i$ yields

$$\dot{\mathcal{H}} = -k\sum_{i=1}^{n} \frac{\|\boldsymbol{J}_i\|^2}{\|\boldsymbol{J}_i\|^2 + \lambda^2} \prod_{l\notin\{i,\mathcal{N}_i\}} M_{V_l}. \tag{15}$$

Since k and M_{V_l} are positive, each term of $\dot{\mathcal{H}}$ is always negative. In addition, the cost function is differentiable, and trajectories of robots are bounded in Q. Therefore, the controller keeps the cost function decreasing unless all the $\boldsymbol{J}_i$ are empty vectors (relocating the robots does not change the cost function), which implies a local minimum.[2]

4 Delivery and Assembly Algorithms

Once the assembling robots are in place according to the equal-mass partitioning controller, construction may begin. State machines drive the delivering robots and the assembling robots. During construction we wish to distribute the source components (truss elements and connectors) to the assembling robots in a balanced way. Global balance is asymptotically achieved by a probabilistic target selection of delivering robots that uses ϕ_t as a probability density function. For local balance, the delivering robots are driven by the gradient of demanding mass defined as the remaining structure to be assembled by the robot. Robots with more work left to do get parts before robots with less work left. Each assembling robot waits for a new truss element or connector and assembles it to the most demanding location in *its Voronoi region*. Therefore, construction is purely driven by the density functions regardless of the amount of the source components and it can be done without an

[2] Pavone et. al [7] also developed equitable partitioning using power diagrams that are weighted generalized Voronoi diagrams. They used a different cost function as the average of inverse of the masses. They targeted a different application in the space of the multi-vehicle routing.

explicit drawing of the target structure. We ensure that all the processes of the controllers work in a distributed way and each robot needs to communicate only with neighbors. Details of the control algorithms are explained next.

4.1 Assembly Algorithm

Each assembling robot operates using a state machine as shown in Figure 3. The robot has the following states:

- IDLE
- WAITING: waiting for a new component
- MOVING: moving to the optimal location to add the part
- ASSEMBLING: adding the component to the assembly

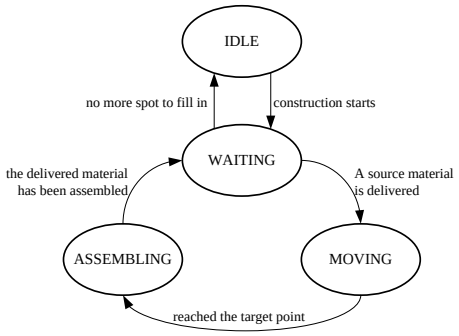


Fig. 3 The state machine for an assembling robot. Each assembling robot waits for the delivery of a source component, moves the component to the optimal spot and adds it to the structure. The robot's task is complete when there is no demanding mass left.

Each robot has a graph representation $G_i = (R_i, E_i)$ of the already built substructure. The graph is composed of sets of nodes and edges in the Voronoi region. For simplicity of exposition, we assume truss elements of two sizes: the unit-box size, and the unit box diagonal. The extension to multiple sizes is trivial. We design the density function according to a grid. The unit length of the grid is the length of the truss element. Vertices of the grid have density values equal to the number of truss elements at the vertex. The density of the intermediate points in the space is interpolated. The interpolated value is used in the coverage implementation only. We can generalize this cost function to be a continuous function that encodes the geometry of the object. The demanding mass is defined uniquely for each component type. As for a truss element, the demanding mass $\Delta M^t_{V_i}$ is computed as:

$$\Delta M^t_{V_i} = \int_{V_i} \phi_t(\mathbf{q}) d\mathbf{q} - \int_{V_i} \rho(\mathbf{q}) d\mathbf{q}, \tag{16}$$

where $\rho(\mathbf{q})$ is the density function of the built structure, which increases as a robot assembles truss elements. Note $\phi_t(\mathbf{q})$ of the target shape is fixed. Therefore, a bigger demanding mass means that more elements should be included in that area. The demanding mass for connectors $\Delta M^c_{V_i}$ is the number of required connectors Φ^c for the current structure G_i. Note that $\Delta M^c_{V_i}$ is a function of $\phi(\mathbf{q})$. The demanding masses drive a delivering robot according to gradients as in (Section 4.2). If a structure is composed of other components, we can define the demanding mass for each material.

Algorithm 2 shows the details of the state machine. When construction starts, an assembling robot initializes the parameters R, E, ρ, Φ^c and changes its state to

WAITING. Once a new truss element is delivered, the robot finds the optimal place to add it to the structure using Algorithm 3. Since we want the structure to gradually grow, the optimal edge is chosen among a set of edges E_1 that are connected to G. Let E_2 be a set of edges that have maximum demanding mass in E_1. The demanding mass of an edge can be computed as the sum of masses of two nodes defining the edge. Each node of the edges in E_2 should have a density value greater than the threshold preventing the robot from assembling the component outside the target structure. In order to achieve a spreading-out structure, priority is given to unconnected edges. If no such edge exists, we choose another seed edge that is not connected to G and has the maximum demanding mass. This jump is required in case that the robot covers substructures which are not connected to each other. If the delivered material is a connector, the optimal location is a node $v \in \Phi^c$ that is connected to the largest number of edges in E. The state machine sets a target location $\mathbf{t}$ according to the optimal location and changes the state to MOVING. In the MOVING state, an assembling robot moves to the target location $\mathbf{t}$ and changes the state to ASSEMBLING when it arrives. Finally, a robot assembles the delivered material and updates the parameters. It adds a node of the optimal edge to Φ^c if the node $\notin \Phi^c$ and is connected to other edges. If the material is a connector, the robot removes the node from Φ^c. The state switches to WAITING again.

Algorithm 2 Control Algorithm of assembling robots

STATE: IDLE
$R = \varnothing, E = \varnothing$
$\rho(\mathbf{q}) = 0, \Phi^c = \varnothing$
state=WAITING
STATE: WAITING
if truss delivered **then**
  e=findOptimalEdge(R, E, ϕ_t, ρ) (Alg. 3)
  if $e \neq \varnothing$ **then**
    $\mathbf{t} = \mathbf{q}_{(node_1(e)+node_2(e))/2}$
    state=MOVING
  else
    state=IDLE
  end if
end if
if connector delivered **then**
  $v \leftarrow \Phi^c$
  $\mathbf{t} = \mathbf{q}_v$
  state=MOVING
end if
STATE: MOVING
if reached $\mathbf{t}$ **then**
  state=ASSEMBLING
else
  move to $\mathbf{t}$
end if
STATE: ASSEMBLING
assemble the material
if the material = truss **then**
  update $\rho(e)$
  if $node_2 \in R$ and $node_i \notin \Phi^c$ **then**
    $\Phi^c \leftarrow node_i$
  end if
  $E \leftarrow e$
  $R \leftarrow \{node_1(e), node_2(e)\}$
end if
if the material = connector **then**
  $\Phi^c = \Phi^c - \{v\}$
end if
state=WAITING

4.2 Delivery Algorithm

delivering robots operate by a state machine as shown in Figure 4. Each robot has the following states:

- IDLE

Algorithm 3 Finding the Optimal Edge to Build

$E_1 = \varnothing, E_2 = \varnothing, E_3 = \varnothing$
if $E_1 = \varnothing$ **then**
  $e_{opt} = \text{argmax}_e(\phi_t(e) - \rho(e)) \cap (\phi_t(e) > \Delta\phi_{threshold})$
else
  $E_1 \leftarrow e, (e \notin E, node(e) \in R)$
  $E_2 \leftarrow \text{argmax}_{e \in E_1}(\phi_t(e) - \rho(e)) \cap (\phi_t(e) > \Delta\phi_{threshold})$
  if $E_2 = \varnothing$ **then**
    $e_{opt} = \text{argmax}_e(\phi_t(e) - \rho(e)) \cap (\phi_t(e) > \Delta\phi_{threshold})$
  else
    $E_3 \leftarrow e, (e \in E_2, \{node_1(e), node_2(e)\} \in \{R_i, R_{j,j \in \mathcal{N}_i}\})$
    if $E_3 \neq E_2$ **then**
      e_{opt}= random($E_2 - E_3$)
    else
      e_{opt}= random(E_2)
    end if
  end if
end if
return e_{opt}

- ToSOURCE: moving to get a new element
- ToTARGET: moving to a picked point at the target area Q
- ToASSEMBLY: delivering the element to an assembling robot

Algorithm 4 describes the details of the state machine.[3] Given an initially empty state, a delivering robot changes its state to ToSOURCE and moves to $\mathcal{S}$ (the source location). At $\mathcal{S}$, the robot picks a source component if one exists. Otherwise, it stops working. The state is switched to ToTARGET and the robot moves to a randomly chosen point in Q following the probability density function ϕ_t. Therefore, materials are more likely to be delivered to an area with a denser ϕ_t. After arrival at the chosen point, the robot changes the state to ToASSEMBLY and moves following the gradient of the demanding mass ΔM_{V_i} of assembling robots. Delivery by the gradient of the demanding mass yields a *locally* balanced mass distribution. Note that the global balance is maintained by the randomly chosen delivery with density ϕ_t. When the robot meets the assembling robot with the maximum demanding mass, it checks

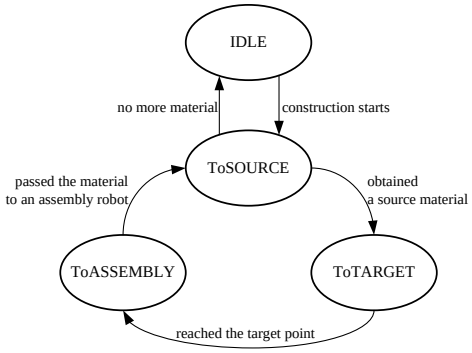


Fig. 4 The state machine for a delivering robot. A delivering robot repeatedly passes source components from the source location to an assembling robot. The initialization of construction causes the delivering robots to start moving. The robots finish working when there is no more source material left at the source location or the assembly is complete.

[3] The assembly and the delivery algorithms provably guarantee completion of the correct target structure. In the interest of space, the proof is omitted. Empirical results in Section 6 shows correctness of the algorithms since all the simulations with different initial conditions end up with the same final structure.

if the state of the assembling robot is WAITING and passes the material. The state changes to ToSOURCE and the robot repeats delivery.

Algorithm 4 Control Algorithm of delivering robots

STATE: IDLE
state = ToSOURCE
$\mathbf{t} = \mathcal{S}$
STATE: ToSOURCE
if reached $\mathbf{t}$ **then**
  if source material remains **then**
    pick a material element
    $\mathbf{t} = \mathbf{q}, \mathbf{q} \sim \phi_t(\mathbf{q})$
    state = ToTARGET
  else
    state = IDLE
  end if
else
  move to $\mathbf{t}$
end if
STATE: ToTARGET
if reached $\mathbf{t}$ **then**
  state=ToASSEMBLY
else
  move to $\mathbf{t}$
end if
STATE: ToASSEMBLY
communicate with robot r_i s.t. $\mathbf{q} \in V_i$
$deliveryID = \mathrm{argmax}_{(k=i, j \in \mathcal{N}_i)} \Delta M_{V_k}$
$\mathbf{t} = p_{deliveryID}$
if reached $\mathbf{t}$ & *state* of r_i = WAITING **then**
  pass the material
  state = ToSOURCE
  $\mathbf{t} = \mathcal{S}$
else
  move to $\mathbf{t}$
end if

5 Adaptation

We briefly discuss the adaptive features of the construction algorithm. Proofs, details and implementation will be in future work.

Theorem 2. *Continuous coverage during construction compensates for failure of robots*

In the proposed framework for robotic construction, a failure of an assembling robot is critical since the robot covers a unique region. Control that uses equal-mass partitioning continuously during the construction makes the remaining robots automatically compensate for the failed assembling robot. The assembling robots reconstruct the Voronoi regions when the surrounding network of the robots has changed (in implementation, assembling robots keep contact with the neighbor robots.) The assembling robots also need to update the parameters such as the graph of the built structure, the demanding mass, etc. The delivering robots achieve this transparently.

Theorem 3. *The algorithms are adaptive to construction in order.*

The construction algorithm is also adaptive to a time varying density function. This property has a nice side-effect: it enables construction *in order*, with connectivity constraints in 3D. For example, the robots can build a structure from the ground up by revealing only part of ϕ_t that is connected to the current structure.

Theorem 4. *The proposed algorithms can be extended for reconfiguration from an existing structure.*

The goal structure might change after or during construction. The construction algorithm can be to adapt the robots to build a new goal structure from the current structure. Equal-mass partitioning can be used with difference of the target density functions, assuming the assembling robot is capable of disassembly. The delivering robots grab source material from the part of the current structure that is unnecessary for the new goal structure.

6 Implementation

Algorithm 2, 4 and the *equal-mass partitioning* were implemented for building 2D and 3D structures. We use side truss elements and connectors that lie at a single source location. We have built several structures using these algorithms.

6.1 Building an A-shaped bridge

The first simulation demonstrates the construction of a bridge from a single source location of trusses and connectors. The density function ϕ_t and the final Voronoi regions resulting from using the equal-mass partitioning controller for 4,6, and 10 assembling robots are shown in Figure 5. We use a discrete system so that ϕ_t is defined at every node (integer points). The unit length is the length of a truss element. At an arbitrary point $\mathbf{q}$, $\phi_t(\mathbf{q})$ is interpolated from 4 surrounding nodes by barycentric interpolation. The interpolation ensures continuity of ϕ that is required for the cost function $\mathcal{H}$. The robots are deployed from randomly selected starting positions. Figure 5 shows that each robot has approximately the same area of the yellow region. As expected, the masses converge to the same value as shown in Figure 6(b), and the cost function $\mathcal{H}$ approaches zero as in Figure 6(a). A little jitter in the masses and the cost function graphs comes from discrete numerical integrals.

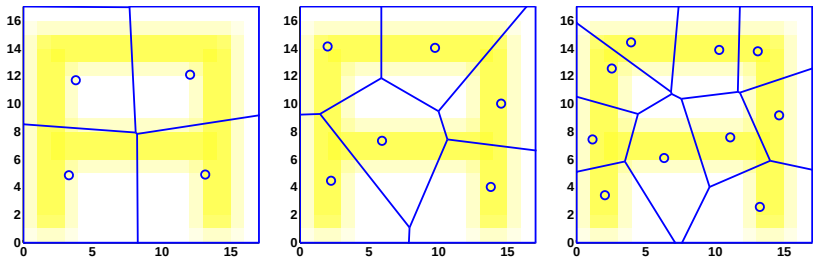
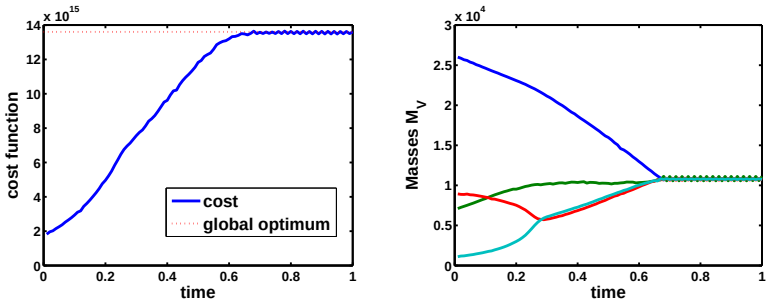


Fig. 5 Density function for an A-shaped bridge and coverage by the equal-mass partitioning. The blue circles are assembling robots. Yellow regions have dense ϕ_t.

Figure 7 shows snapshots from the simulation after partitioning. We use 4 robots for truss delivery and 4 robots for connector delivery. They deliver source materials which have 250 side truss elements and 150 connectors. The area with high den-



(a) (b)

Fig. 6 Result from the equal-mass partitioning controller for 4 assembling robots. (a) Cost function $\mathcal{H}$ (b) Masses of four assembling robots

sity is gradually filled with truss elements and connectors. Because the controller uses equal mass partitioning and the gradient of the demanding mass, the assembling robots maintain almost the same ΔM_V all the time. Therefore, each Voronoi region has a balanced amount of truss elements. Note that the control algorithms do *not* depend on the amount of the source truss elements. With fewer elements, we obtain a thinner structure, while the availability of more truss element yields a denser structure. At the end of the simulation, the assembling robot that has built the least amount of the truss component has assembled 58 truss elements while the robot with the maximum amount has assembled 63. The robot with the minimum number of connectors assembled 33 connectors and the robot with the maximum number assembled 38.

Figure 8 shows the demanding masses for a truss part and a connector. All four curves are completely overlapped, meaning all the substructures have been balanced at all time. The demanding mass for a connector oscillates since it depends on the already built substructure.

6.2 Constructing an Airplane

Figure 9 shows snapshots of building an airplane. 3D grids are used and the target density functions are given and computed in the grids.

6.3 Experiment

We have implemented Algorithm 4 using a team of robots. The robots are networked using the Meraki mesh networking infrastructure. The robots (Figure 10) combine an iRobot platform for navigation with a Crust Crawler 4-dof arm and use a Vicon system for location feedback. The setup allowed us to verify the coordination and computation required by part delivery. Currently, we have a single type of source

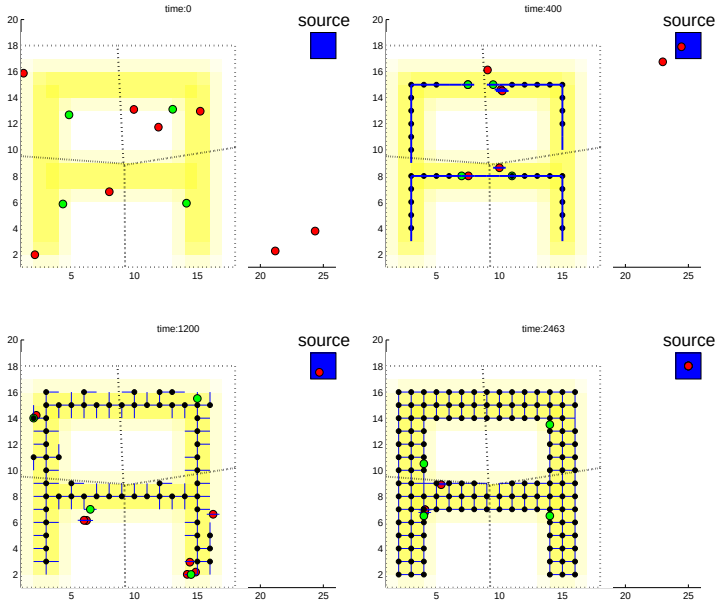
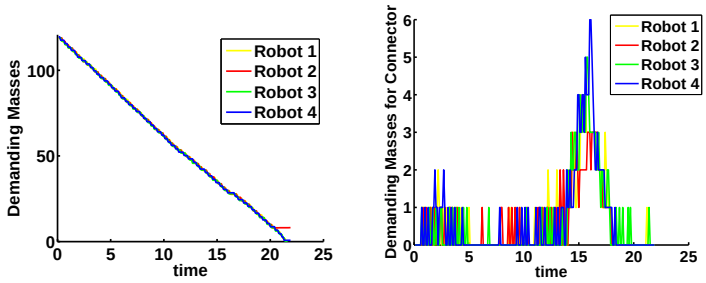


Fig. 7 Snapshots of simulation. Green circles denote assembling robots and red circles denote delivering robots. The blue line is a truss elements and the black dot is a connector. The blue box is the source location. The dotted lines in Q are boundaries of the Voronoi regions.



(a) (b)

Fig. 8 (a) Demanding masses for a truss part and (b) a connector. 4 assembling robots and 8 delivering robots are used. The assembly time is set to ten times the velocity. All the graphs are almost overlapped.

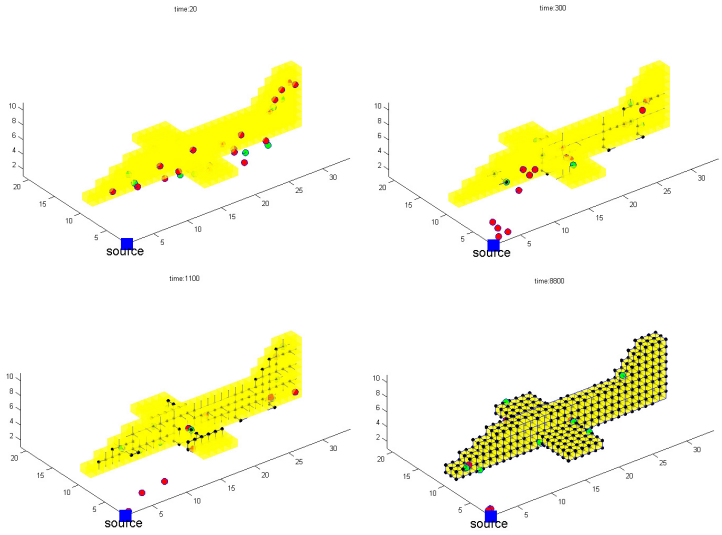


Fig. 9 Snapshots of building an airplane pyramid. There are 10 assembling robots, 20 delivering robots for truss parts and 10 robots for connectors. 1575 truss parts and 656 connectors are assembled.

component as the screw in Figure 10. Two delivering robots have performed 20 times of delivery to three assembling robots.

7 Conclusion

We propose a framework for distributed robotic construction. Robots with specialized tasks (assembly and delivery of various parts) cover the target structure which is given by a density function, and perform their tasks with only local communication. To divide the structure in equally-sized substructures, the equal-mass partitioning controller is introduced, guaranteeing convergence of the cost function that is the product of the all the masses. An intuitive control criteria with probabilistic deployment and a gradient of the demanding masses is proposed to maintain a balance among the substructures. Implementation with two kinds of source materials (truss and connector) shows that the proposed algorithms assign an equal amount of construction work to the assembling robots, and effectively construct the target structures. This work has opened many interesting questions which we are pursuing as part of our on-going work. We are currently expanding the hardware experiment.

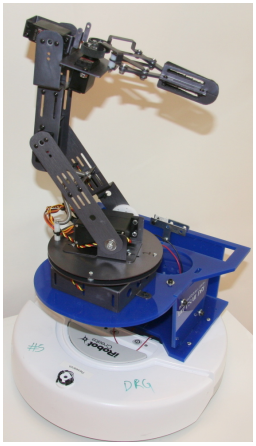


Fig. 10 iRobot platform with Crust Crawler 4-dof arm.

1. **Goal-driven structure** A target structure can be given as an abstract goal such as connecting two points, not as a density function. In this case, each assembling robot should make the locally best decision of how to build a partial structure.
2. **Connectivity in sub-structure** Assembling robots may be constrained to work only on the truss structure in practice. In this case, connectivity through each Voronoi region is critical, since a robot may not reach its own region if some part of the region is separated. We need to incorporate this constraint in distributed coverage.

Acknowledgements This project has been supported in part by The Boeing Company, the U.S. National Science Foundation, NSF grant numbers IIS-0513755, IIS-0426838, CNS-0520305, CNS-0707601, the MAST project, MURI SWARMS project grant number W911NF-05-1-0219, and Emerging Frontiers in Research and Innovation (EFRI) grant #0735953. Seung-kook Yun is supported in part by Samsung Scholarship. We are grateful for this support.

References

1. Carrick Detweiler, Marsette Vona, Yeoreum Yoon, Seung-kook Yun, and Daniela Rus. Self-assembling mobile linkages. *IEEE Robotics and Automation Magazine*, 14(4):45–55, 2007.
2. J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. 20(2):243–255, 2004.
3. Seung kook Yun, David Alan Hjelle, Hod Lipson, and Daniela Rus. Planning the reconfiguration of grounded truss structures with truss climbing robots that carry truss elements. In *Proc. of IEEE/RSJ IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.
4. Seung kook Yun and Daniela Rus. Optimal distributed planning for self assembly of modular manipulators. In *Proc. of IEEE/RSJ IEEE International Conference on Intelligent Robots and Systems*, pages 1346–1352, Nice, France, Sep 2008.
5. Seung kook Yun and Daniela Rus. Self assembly of modular manipulators with active and passive modules. In *Proc. of IEEE/RSJ IEEE International Conference on Robotics and Automation*, pages 1477–1482, May 2008.
6. MC Nechyba and Y. Xu. Human-robot cooperation in space: SM^2 for new spacestation structure. *Robotics & Automation Magazine, IEEE*, 2(4):4–11, 1995.
7. Marco Pavone, Emilio Frazzoli, and Francesco Bullo. Distributed algorithms for equitable partitioning policies: Theory and applications. In *IEEE Conference on Decision and Control*, Cancun, Mexico, Dec 2008.
8. L. C. A. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. S. Pereira. Simultaneous coverage and tracking (scat) of moving targets with robot networks. In *Proceedings of the Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Guanajuato, Mexico, December 2008.
9. Mac Schwager, Daniela Rus, and Jean-Jacques E. Slotine. Decentralized, adaptive control for coverage with networked robots. *International Journal of Robotics Research*, 28(3):357–375, March 2009.
10. S. Skaff, P. Staritz, and WL Whittaker. Skyworker: Robotics for space assembly, inspection and maintenance. *Space Studies Institute Conference*, 2001.
11. Justin Werfel and Radhika Nagpal. International journal of robotics research. *Three-dimensional construction with mobile robots and modular blocks*, 3-4(27):463–479, 2008.