Robotica (2014) volume 32, pp. 257–277. © Cambridge University Press 2013
doi:10.1017/S0263574713001148

Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning

Seung-kook Yun†* and Daniela Rus‡

†*SRI International, Menlo Park, CA 94025, USA*
‡*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*

(Accepted November 1, 2013. First published online: December 18, 2013)

SUMMARY
This paper presents decentralized algorithms for coverage with mobile robots on a graph. Coverage is an important capability of multi-robot systems engaged in a number of different applications, including placement for environmental modeling, deployment for maximal quality surveillance, and even coordinated construction. We use distributed vertex substitution for locational optimization and equal mass partitioning, and the controllers minimize the corresponding cost functions. We prove that the proposed controller with two-hop communication guarantees convergence to the locally optimal configuration. We evaluate the algorithms in simulations and also using four mobile robots.

KEYWORDS: Multi-robot systems; Modular robots; Distributed algorithms; Path planning on discrete spaces.

1. Introduction
Coverage is a core capability of multi-robot systems. In coverage, the goal is to compute a partition of the environment subject to the optimization criteria of the application. Many multi-robot system applications can be formulated as coverage problems, where the robot system deploys itself for maximal modeling or surveillance scope, for load balancing in traffic, or for uniform task progression in construction. All these applications can be modeled as graphs and can be parallelized. Most current solutions to multi-robot deployment and coverage are formulated for continuous convex environments and are centralized. We wish to have decentralized controllers to ensure good scalability properties in the number of robots. Moreover, we wish for the decentralized coverage controllers to have discrete formulations in order to support applications in arbitrarily shaped environments with obstacles, and expressivity for a broader class of problems.

Current approaches to discrete coverage problems adapt existing continuous methods to the discrete domain. Distributed coverage for multi-robot systems in continuous space has been studied to optimize locations of robots,[1,2] to find the best partition for vehicle routing,[3] and to distribute workload equally.[4] These prior results require a convex obstacle-free target area in the Euclidean space and a continuous weighting function on the target area called a density function. However, many applications in assembly, construction, transportation, and resource allocation have a discrete nature and thus would benefit from a discrete formulation of coverage algorithms. For example, we can use coverage to model building truss structures, where the construction elements are bars and connectors such as screws. This problem can be encoded with a continuous formulation.[4] The coverage algorithm computes sub-assemblies that are guaranteed to split the work evenly among the construction robots. This continuous formulation required a continuous blueprint of the desired object, which is accomplished by mapping the discrete blue print of the target object to a continuous function. This step introduces approximation errors, and could be eliminated if a discrete solution to the coverage problem was available. Discrete coverage on a graph could be used as the core computation for construction and other problems that are intrinsically discrete. Moreover, distributed coverage on a graph also provides

* Corresponding author. E-mail: seungkook.yun@sri.com

a solution to a broad class of coverage problems whose environments are non-convex and have obstacles.

In this paper we describe a decentralized algorithm for locational optimization,[1] where the robots deploy themselves at optimal locations with respect to a given cost function on a graph. The algorithm uses a Voronoi-based method which converges when robots reach the weighted Voronoi centroids. We then extend this algorithm to derive a decentralized equal-mass partitioning algorithm using a different cost function, which distributes equal node weights to each robot, and demonstrate this algorithm in the context of decentralized construction. The algorithm uses vertex substitution to sequentially find the best partitions by checking every possible movement of a single centroid (robot position in our case). We prove convergence of the algorithm to local minima in solution space and experimentally demonstrate that a large fraction of the solutions found by our algorithms are statistically close to the global optima. A surprising result is that two-hop communication rather than single hop communication to neighbors is required for the best performance.

The contributions of this paper are: (1) development of the distributed algorithm for coverage on a graph using the Voronoi tessellation, (2) evaluation of the algorithms on generic test graphs used for operations research, (3) the communication condition required for convergence, and (4) the hardware implementation of the algorithms.

1.1. Related work

This work builds on two areas of the previous work: distributed coverage and graph partitioning. Although a multi-robot coverage was extensively studied before in both a continuous domain[5] and a discrete graph,[6] the focus was on a centralized controller.[7] We are revisiting this area with a distributed controller, specifically focused on the coordinated construction.

Distributed coverage for cases where the *continuous* sensory function is known was proposed in Cortes *et al.*[1] The sensory function described weight or importance. Using Vornoi tessellations, robots compute deployment regions whose distributions are guaranteed to match the sensory function in a convex environment. Adaptive coverage with unknown sensory functions was proposed in Schwager *et al.*,[2] where the robots learn the sensory function as they move to achieve coverage and the system is guaranteed to converge to a locally optimal configuration in a convex environment. In equi-partitioning,[3,4] robots divide workload – integratd sensory information in each Voronoi partition – into equal amounts. Distributed coverage was also considered in the context of heterogeneous robots,[8] where aerial and ground vehicles collaborate, and power-aware coverage[9] of a sensor network where high powered sensors compensate for low-powered sensors in the network. These works were limited to a convex environment with smooth sensory functions.

Recently, the coverage algorithms have been extended to the coverage of a non-convex region under special circumstances.[8,10–13] The visibility-based deployment problem was addressed in Ganguli *et al.*,[10] where a team of robots solve the art-gallery problem. In Caicedo-Nunez and Zefran[11] a non-convex region is transformed to a convex region by a diffeomorphism. Pimenta *et al.*[8] use the geodesic distance measure for a non-convex region instead of Euclidean distance. Controlling mobile robots with proximity constraints was addressed for a known environment with obstacles in Ayanian and Kumar.[12] These solutions work for certain classes of non-convex environments; however, finding a general solution for non-convex environment remains unsolved.

Graph coverage and partitioning have been extensively studied to find the optimal locations of resources[14] and to distribute workload equally. For two excellent surveys, see Reese[15]and Fjallstrom.[16] These methods were used in robotics.

Gabriely and Rimon[17] and Durham *et al.*[18] use an environment discretized by grid cells, and a *centralized* algorithm based on spanning trees directs the robots to cover the environment. In Durham *et al.*,[18,19] a group of mobile robots are deployed to cover a discretized environment by gossip communication which requires the entire knowledge of neighbor partitions.

Our work is unique as the proposed algorithm is fully distributed, based on graph Voronoi tessellations, and evaluated on a set of generic graphs which have been used in order to test the algorithms for the p-median problem. Therefore, our algorithm can be used not only for a mobile robot application but also for solving the generic p-median problem in a distributed way.

[1] known as p-median problem.

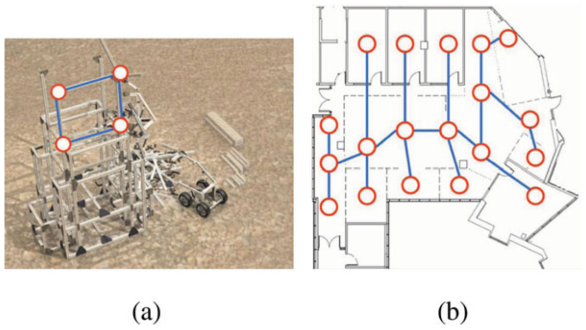


Fig. 1. (Colour online) Applications of distributed coverage on graph. (a) Concept art for the construction of a truss structure by mobile delivering robots and truss-climbing assembling robots. Trusses are discrete in nature, and we can model them as a set of nodes and edges (reprinted with permission from Jonathan Hiller, Cornell University, USA). (b) Coverage of a complicated non-convex region represented by a topological map. The blue robots are covering the 3rd floor of Stata Center at MIT. Each room is modeled as a node with corresponding importance as a weight.

1.2. Organization

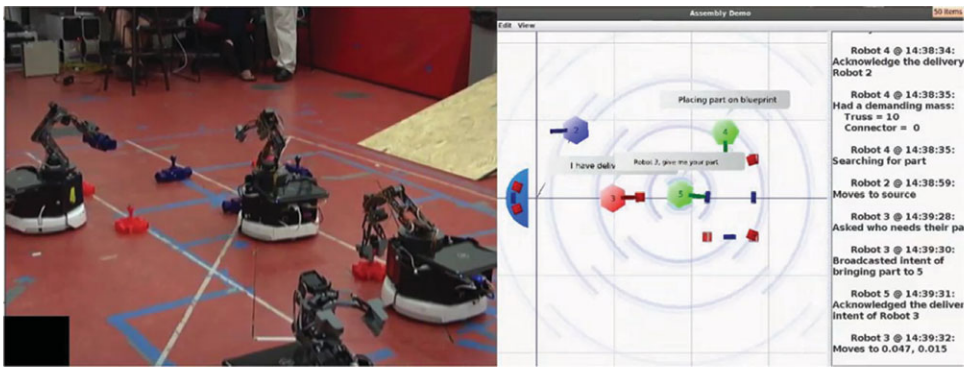
The paper is organized as follows. Section 2 introduces distributed coverage and coverage on graphs. We formulate the locational optimization problem on a graph in Section 3. Section 4 proposes the controller and the distributed vertex substitution algorithm. The algorithms are tested for two regularly spaced graph topologies as well as generic graphs from the OR library.[20] As extensions to the algorithm, distributed equal-mass partitioning on a graph is proposed and implemented in Section 5.

2. Coverage on a Graph

In distributed coverage on a graph, the goal is to partition the graph such that a set of mobile robots with local information only about the activities of the other robots in the system will place themselves to optimally cover the nodes of the graph according to the problem-specific metric. In this paper, we introduce two metrics: (1) locational optimization to optimize the sum of distances from each robot to its clustered node, and (2) equal-mass partitioning in order to equally assign a part of the graph to the robots. The robots, the environment, and the actions in such a system are all discrete. For example, we can apply this method to decentralized construction (see Fig. 1(a)) where robots cooperate to assemble a complex structure out of discrete components by dividing it into sub-assemblies. Our coverage algorithm can assign the sub-assemblies (sub-graphs) to each robot, and each robot constructs its own sub-assembly, given the delivery of source material.[21] Most decentralized coverage solutions for continuous spaces work with convex environments,[1–3] but many indoor and outdoor environments are not convex (for example, see Fig. 1(b)). Decentralized coverage on a graph can be used to cover continuous non-convex regions by modeling the non-convex region as a mesh network as in finite element methods (FEM) or a topological map. Both continuous and discrete methods are scalable in the number of robots and can handle robot failures.[1]

The discrete coverage on a graph algorithm has several advantages over the continuous approach, which are as follows:

1. The continuous coverage methods require a density function. The discrete density function that is natural in the discrete domain problem has to be processed to produce a continuous density function for the continuous approach; this step introduces approximation errors.
2. Using continuous methods adapted to the discrete domain often produce regions that consist of disconnected subsets; using coverage on a graph, we can guarantee that each computed coverage region stays connected.
3. The robot neighbors computed by continuous coverage methods adapted to the graph domain may not be physically reachable, while the neighbors computed by the discrete approach share Voronoi edges and are thus reachable.



```
Robot 4 @ 14:38:34:
Acknowledge the delivery
Robot 2

Robot 4 @ 14:38:35:
Had a demanding mass:
  Truss = 10
  Connector =  0

Robot 4 @ 14:38:35:
Searching for part

Robot 2 @ 14:38:59:
Moves to source

Robot 3 @ 14:39:28:
Asked who needs their pa

Robot 3 @ 14:39:30:
Broadcasted intent of
bringing part to 5

Robot 5 @ 14:39:31:
Acknowledged the deliver
intent of Robot 3

Robot 3 @ 14:39:32:
Moves to 0.047, 0.015
```

Fig. 2. (Colour online) A snapshot of robotic assembly experiment. The right side is GUI display. Two assembly robots (green octagons in GUI) and two delivery robots cooperate to build a given truss structure, which can be modeled as a weighted graph. The graph coverage algorithm is a suitable solution to find the optimal formation. The details of the experiments can be found in Bolger *et al.*[21]

4. The continuous methods for distributed coverage work with obstacle-free convex environments and special classes of concave environments, while the graph solution works for any type of convex or concave environment with obstacles.

3. Problem Formulation

We are given a team of n robots. They have to cover an undirected graph $G = (Q, E)$ with the configuration[2] $\{p_1, \ldots, p_n\}$, where Q is the set of vertices and E is its set of edges. Let $p_i \in Q$ be the vertex location of the ith robot. Let $d(\cdot,\cdot) : Q \times Q \to \mathbb{R}^+ \cup \{\infty\}$ denote the shortest distance between two vertices in the graph. The distance $d(s, t) = \infty$ when there is no path from s to t in the graph. The cost of each edge is strictly positive. Each vertex $q \in Q$ has associate with it a node-weight $\phi(q)$ denoting the importance the task at vertex q. We call $\phi : Q \to R^+$ the target density function.

The set of robot locations induces a Voronoi partition of G.[22] The intuition behind the partition is that the task at vertex q will be executed by the nearest robot to q. Thus, each robot in the system is allocated the tasks in its Voronoi partition V_i in G,

$$V_i = \left\{q \in Q | d(q, p_i) < d(q, p_j), \forall j \neq i\right\}. \tag{1}$$

Unlike Voronoi partitioning in a continuous space, we have to clarify the assignment of a node that has the same distance to multiple robots. We give priority to the robot with the minimum identity (ID) according to the following condition:

$$q \in V_i \Rightarrow i = \min\left\{j | d(q, p_i) = d(q, p_j)\right\}. \tag{2}$$

A controller reassigns p_i in order to minimize a certain cost function $\mathcal{H}(p_i)$ until the configuration reaches *local minimum*.

The distributed coverage algorithm makes the following assumptions:

1. The environment $(G, \phi(q))$ is given to each robot.
2. The node weight $\phi(q)$ is fixed.
3. The robots do *not* know the initial locations of other robots nor how many other robots there are in the system. Each robot is aware of information of only their neighbors which share an edge.
4. The robots *do* precompute the distance matrix D of G as a $|Q| \times |Q|$ symmetric matrix where the matrix element d_{ij} is $d(q_i, q_j)$.
5. There is no moving obstacle and no uncertainties.

Although the first assumption sounds very strong, it was generally accepted in the early stage of distributed coverage.[1] Since this work is one of the starting points for distributed coverage on

[2] A configuration in this work denotes a set of the robot locations.

a graph, we leave simultaneous exploring and covering as a future work which may be extended from Schwager *et al.*.[2]

Because of the third assumption, the robots can not precompute the optimal configuration where such optimality requires the knowledge of entire robots. The matrix D can be computed with $O(|Q|^3)$ runtime by the Floyd–Warshall algorithm.[23]

Next, we describe the partitioning problems studied in this paper: locational optimization and equal-mass partitioning. We focus on solving the locational optimization, and the solution is extended for equal-mass partitioning.

3.1. Locational optimization

Locational optimization has been extensively researched in operations research for a variety of optimization problems such as placing facilities to minimize costs (distances). For example, how should we locate post offices to minimize the total distance from inhabitants in the area? Recently, the locational optimization was revisited in robotics and control for distributed coverage of multi-robot systems.[1,2] In distributed coverage, a team of robots cover an area of interest to optimize a cost function.

In graph theory, this problem is called p-median (p should not be confused with the standard way of denoting the position of a robot by variable $\mathbf{p}$). The goal is to find the best set of medians (centroids) of the given graph, which results in the optimal clusters of the graph. The cost function is given as:

$$\mathcal{H}_L = \sum_{i=1}^{n} \sum_{q \in V_i} \phi(q) d(q, p_i), \tag{3}$$

which is similar to the cost function used in locational optimization in continuous space:[1]

$$\hat{\mathcal{H}} = \sum_{i=1}^{n} \int_{V_i} \phi(\mathbf{q}) \left\|\mathbf{p}_i - \mathbf{q}\right\|^2 d\mathbf{q}, \tag{4}$$

where $\mathbf{q}$ and $\mathbf{p}_i$ are now position vectors.

The p-median problem is NP-hard for a non-tree graph.[24] A great number of heuristic centralized solutions have been proposed.[15] Our approach implements distributed coverage of multi-robot system on a graph and is new in that it provides the following:

1. A distributed controller for a mobile robot system.
2. A geometry-based solution using graph Voronoi tessellation.

4. Decentralized Control Algorithms for Locational Optimization

In this section we describe the decentralized controllers that achieve locational optimization. Our solution is iterative and the partitions are only updated by physical movements of the robots. Our algorithm assumes local synchronization among neighbors, therefore it does not guarantee complete asynchronous control, however it provides parallel execution. In practice, the robot spends most of the time for physically moving, which takes much larger time than communication bandwidth, therefore asynchronous control can be used without causing instability (cyclic motion). We will extend the controller and apply it to equal-mass partitioning in Section 5.

Algorithm 1 shows the main control loop. Each robot has the following two states:

- *COMPUTE*: Compute the optimal node to relocate
- *MOVING*: Move to the optimal node.

$\mathcal{N}_i$ is the set of IDs of the neighbors of robot i. The neighbors are defined as robots whose graph Voronoi partition share edges with V_i. We assume the neighbors are always in communication range, which is commonly accepted in distributed coverage.[3]

[3] Work on the communication range issues can be found in Julian *et al.*[25]

Algorithm 1 Distributed controller.

```
Require: COMPUTE
 1: Communicate with $\mathcal{N}_i$
 2: Construct a new Voronoi partition by $\{p_i^*, p_{\mathcal{N}_i}^*, p_{\mathcal{N}_{\mathcal{N}_i}}^*\}$
 3: Find the new optimal $p_i^*$ (Algorithm 2)
 4: if $p_i \neq p_i^*$ then
 5:     state ← MOVING
 6: end if
Require: MOVING
 7: Move to $p_i^*$
 8: if $p_i = p_i^*$ then
 9:     state ← COMPUTE
10: end if
```

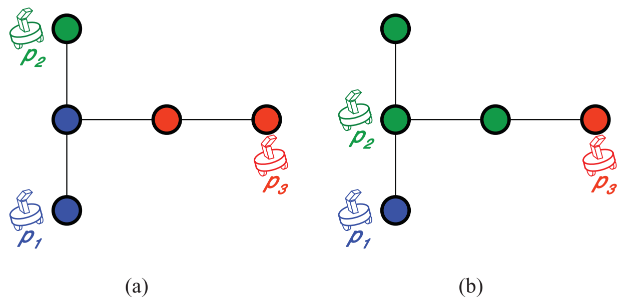


Fig. 3. (Colour online) An example shows why a robot needs to know information of the neighbors of the neighbors $\mathcal{N}_{\mathcal{N}_i}$. Each node and edge have unit weight and cost. Colors of nodes denote which robot they belong to. Remind that nodes which have the shortest distance to multiple robots belong to the robot with minimum ID.

In contrast to the distributed coverage controller in a continuous domain[1] where each robot requires only information about its neighbors $\mathcal{N}_i$, in the graph case each robot needs to know information about all the neighbors of its neighbors $\mathcal{N}_{\mathcal{N}_i}$ (excluding $\mathcal{N}_i$ and robot i) as well. This is because relocation of the robot on a graph cannot be infinitely small as in the controllers for the continuous domain. Therefore, the relocation may change the Voronoi region of $\mathcal{N}_{\mathcal{N}_i}$. Figure 3 shows an example. The initial graph's Voronoi tessellation is shown in Fig. 3(a). Each color represents its Voronoi region V_i. All the edges are unit distance long. If robot 2 moves downward by an edge as in Fig. 3(b), V_3 changes, although robot 3 was not a neighbor of robot 1 in the initial configuration. Because of this, unlike the continuous case of locational optimization, repeatedly moving each robot to its current centroid does not guarantee the decay of the cost function.

In the *COMPUTE* state, the robot communicates and receives information about the neighbors $\mathcal{N}_i$ and the neighbors of its neighbors $\mathcal{N}_{\mathcal{N}_i}$ to construct the graph Voronoi tessellation. p_i^* is the centroid of V_i denoting the desired location for robot i. Note that p_i^* can be different from the actual position p_i while the robot is moving to p_i^* in the *MOVING* state. This is a key to implement parallel execution of the control algorithm. Therefore, the graph of Voronoi tessellation should be built from the set of p^* (new centroid), not from p. After building the current Voronoi tessellation, each robot determines the optimal node for its relocation. Algorithm 2 finds the optimal node for locational optimization. The algorithms guarantee decay of the cost functions. If the found p_i^* is not p_i, the robot switches to the state *MOVING* and moves to p_i^*.

4.1. Why two-hop information?

Intuitively, we need two-hop communication because we need to access the location of the neighbors' neighbors. This can be done by two-hop communication (contact the neighbors to get their neighbors' locations). Alternatively, we can store the location of the neighbors with each node and use one-hop communication. The trade-off is that as the neighbors move, many updates (hence communications) may be necessary.

More specifically, we want relocation of robot i to change only the Voronoi regions of itself and its neighbors so that we can decouple the cost function as follows:

$$\mathcal{H}_L = \mathcal{H}_i + \mathcal{H}_{\setminus i},$$

where

$$\mathcal{H}_i = \sum_{l \in \{i\} \cup \mathcal{N}_i} \sum_{V_l} \phi(q) d(q, p_l),$$

and

$$\mathcal{H}_{\setminus i} = \mathcal{H}_L - \mathcal{H}_i.$$

$\mathcal{H}_i$ is a part of the cost function that can be changed by the relocation of robot i.[4] It includes only V_i and $V_{\mathcal{N}_i}$, where V_i is Voronoi partition of robot i. We want the remaining part $\mathcal{H}_{\setminus i}$ untouched while robot i is moving. To ensure this decoupling, robot i should know the locations of $\mathcal{N}_{\mathcal{N}_i}$ (neighbors of the neighbors). Note that we have shown that the relocation of a robot may change the Voronoi region of $\mathcal{N}_{\mathcal{N}_i}$. Given the locations $\mathcal{N}_{\mathcal{N}_i}$, the proposed *distributed vertex substitution* algorithm ensures no change in $\mathcal{N}_{\mathcal{N}_i}$.

Without two-hop information, we cannot decouple the cost function.

Next, we explain the details of the algorithms for locational optimization. The algorithm is based on *vertex substitution*. Vertex substitution[14] is known as a typical solution for the p-median problem.[15] We modify it to fit our problems and call the modified version *distributed vertex substitution*.

4.2. *Distributed vertex substitution algorithm*

Algorithm 2 describes the distributed vertex substitution algorithm for locational optimization. Each robot runs this algorithm independently as follows. Robot i picks a candidate node q_b in V_i as its next centroid p_i^*, and checks how the movement to q_b will change $\mathcal{H}_L$ by investigating all the possible changes to the Voronoi partitions. The algorithm returns the candidate with the maximal reduction of $\mathcal{H}_L$. If there is no candidate that decreases $\mathcal{H}_L$, the algorithm returns null.

Given the position set $P_i = \{p_i^*, p_{\mathcal{N}_i}^*, p_{\mathcal{N}_{\mathcal{N}_i}}^*\}$, where $p_{\mathcal{N}_i}^*$ is the list of the centroids of the neighbors and $p_{\mathcal{N}_{\mathcal{N}_i}}^*$ is the list of the centroids of the two-hop neighbors, let D_i^Q be the $|Q| \times |P_i|$ sub-matrix of D whose rows and columns match P_i (see line 2 of Algorithm 2). The algorithm finds the optimal node for substituting the current position among q_b, the candidate node for the next centroid location of robot i in V_i (line 2 of algorithm 2). The algorithm then checks how the substitution of p_i by q_b will affect the Voronoi tessellation and the cost function, by examining how the nodes $q_j \in \{V_i \cup V_{\mathcal{N}_i} \cup B_{\mathcal{N}_i}\}$ will change, where $B_{\mathcal{N}_i}$ is a set of vertices that do not belong to $V_{\mathcal{N}_i}$ but share edges with other vertices. Therefore, it represents the nodes of $\mathcal{N}_{\mathcal{N}_i}$ that can be affected by the movement of robot i.

In line 4–7 of Algorithm 2, d_{jk} is the minimum distance from q_j to the robots (robot i, $\mathcal{N}_i$, and $\mathcal{N}_{\mathcal{N}_i}$), r_k is the ID of the robot with the minimum distance that is located at node k, and d_{js} is the distance from q_j to the second closest robot (closest except for robot r_k). Accordingly, q_s is the node of the second closest robot.

We consider whether q_j belongs to robot i *before* substitution of p_i^* by q_b. If so, the substitution may lead to the following three cases:

1. $d(q_j, q_b) < d(q_j, p_i)$,
2. $d(q_j, q_s) > d(q_j, q_b) > d(q_j, p_i)$,
3. $d(q_j, q_b) > d(q_j, q_s) > d(q_j, p_i)$.

In the first case, $q_j \in V_i$ after the substitution, therefore $\mathcal{H}_L$ decreases by $\phi(q_j)(d(q_j, q_b) - d(q_j, p_i))$, which is denoted as ${}_j\Delta_{bi}$ in line 2 in Algorithm 2. In the second case, still $q_j \in V_i$, however q_j is further away from robot i now. Therefore, $\mathcal{H}_L$ increases by the same amount

[4] Note that $\mathcal{H}_L \neq \sum_i \mathcal{H}_i$. Depending on a number of neighbors and configurations, movement of robot i can affect $\mathcal{H}_L$ differently. However, the amount of change in $\mathcal{H}_i$ contributes to the same amount of change in $\mathcal{H}_L$.

$\phi(q_j)(d(q_j, q_b) - d(q_j, p_i))$. For the last case, q_j will belong to the second closest robot and $\mathcal{H}_L$ increases by $\phi(q_j)(d(q_j, q_s) - d(q_j, p_i))$.

If q_j does not belong to robot i before the substitution, the cost function increases by $\phi(q_j)(d(q_j, q_s) - d(q_j, p_i))$ only when the closest robot r_k to q_j is in $\mathcal{N}_i$. If $r_k \notin \mathcal{N}_i$, we may change $V_{\mathcal{N}_{\mathcal{N}_i}}$. Therefore, we do not consider q_b as a substitute for p_i^*. This guarantees that the algorithm will only change the neighboring Voronoi regions.

The final node for substitution is chosen to reduce the cost function most among all Δ_{bi}, which is the sum of each ${}_j\Delta_{bi}$, where $|Q_j|$ is the number of q_j. Δ_b, the minimum of Δ_{bi}, must be negative, otherwise the algorithm returns null, that is, robot i does not move.

Algorithm 2 Distributed vertex substitution algorithm for locational optimization.

```
1: $D_i^Q = D(:, p_i^* \cup p_{\mathcal{N}_i}^* \cup p_{\mathcal{N}_{\mathcal{N}_i}}^*)$
2: for $q_b = \{q \in V_i | q \neq p_i\}$ do
3:    for $q_j = \{q \in V_i \cup V_{\mathcal{N}_i} \cup B_{\mathcal{N}_i} | q \neq p_{\mathcal{N}_i}\}$ do
4:       $d_{jk} \leftarrow \min(\mathrm{row}(D_i^Q, j))$
5:       $r_i \leftarrow$ ID of the robot $i$
6:       $r_k \leftarrow$ ID of the robot at $k$
7:       $d_{js} \leftarrow$ 2nd smallest row$(D_i^Q, j)$
8:       if $r_i = r_k$ then
9:          if $d(q_j, q_b) \leq d(q_j, p_i)$ or $(d(q_j, q_b) > d(q_j, p_i)$ and $d(q_j, q_s) > d(q_j, q_b))$ then
10:            ${}_j\Delta_{bi} = \phi(q_j)(d(q_j, q_b) - d(q_j, p_i))$
11:         else
12:            ${}_j\Delta_{bi} = \phi(q_j)(d(q_j, q_s) - d(q_j, p_i))$
13:         end if
14:      else
15:         if $d(q_j, q_b) < d_{jk}$ then
16:            if $r_k \notin \mathcal{N}_i$ then
17:               discard $q_b$
18:            end if
19:            ${}_j\Delta_{bi} = \phi(q_j)(d(q_j, q_b) - d_{jk})$
20:         end if
21:      end if
22:   end for
23:   $\Delta_{bi} = \sum_{j=1}^{|Q_j|} {}_j\Delta_{bi}$
24: end for
25: $\Delta_b = \min \Delta_{bi}$
26: if $\Delta_b \geq 0$ then
27:    return $\varnothing$
28: else
29:    return $q_{b^*}$ for $\Delta_b$
30: end if
```

4.3. Analysis

The runtime of Algorithm 2 is $O(n|Q|^2)$ due to two loops, where $|Q|$ is the number of nodes in G. Given the locations of the robots connected by a two-hop communication, we prove Algorithm 2's convergence to a critical configuration where the robots do not reconfigure anymore according to the distributed vertex substitution. Note that the critical configurations depend on the control algorithm. They can be different from a critical configuration with the original vertex substitution algorithm[14] (not distributed).

Let Ω be the set of all possible configurations with n robots, and let $M \subset \Omega$ be the set of critical configurations in which the robots do not reconfigure any more ($\Delta_b \geq 0$ for $\forall i$), given the distributed vertex substitution control algorithm. The critical configurations can be local minima or saddle points. Note that our algorithm cannot guarantee convergence to the global optimum. Let $\Delta\mathcal{H}_L$ be the total change of the cost function after running Algorithm 2 for all the robots.

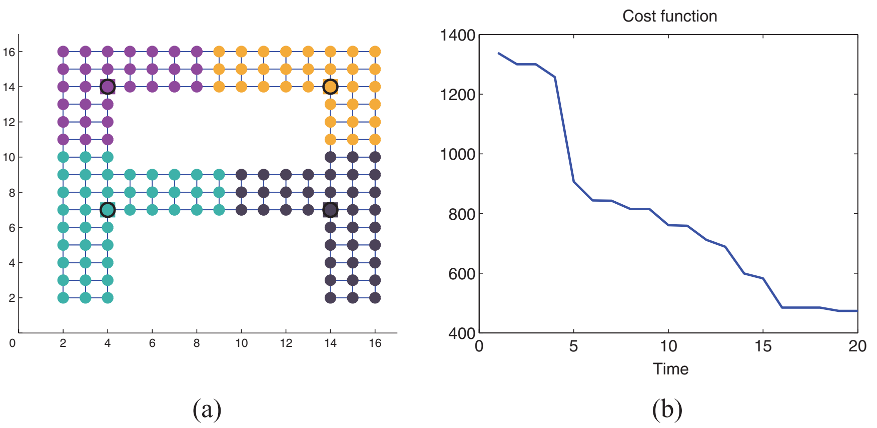


Fig. 4. (Colour online) Simulation result from coverage on the small bridge. Nodes are filled circles and edges are black solid lines connecting the nodes. The circles enclosed by the black outline are robot locations. Each color represents a Voronoi region that belongs to the same colored robot. (a) The final configuration of locational optimization on the small bridge by four robots. (b) The cost function $\mathcal{H}_L$.

Theorem 1. Ω and M are bounded and invariant sets.

Proof. The number of possible configuration is $\binom{|Q|}{n}$. Therefore, Ω has a finite number of configurations, and it is bounded. Also, it is invariant since it contains every possible set. Considering M, given the controller, robots do not move when $\Delta\mathcal{H}_L = 0$. Therefore, once a configuration yields $\Delta\mathcal{H}_L = 0$, this configuration remains constant. □

Theorem 2. Every configuration in Ω converges to M.

Proof. Let $\mathcal{H}_0$ be the minimum of $\mathcal{H}_L$ in Ω. A configuration with $\mathcal{H}_0$ is the global optimum. $\mathcal{H}_0$ is the lower bound of $\mathcal{H}_L$, and the configuration with $\mathcal{H}_0$ should be in M since the robot should not move when their configuration is the global optimum.

Let ϵ be the *smallest negative change* in $\mathcal{H}_L$ between every possible pair of configurations in Ω. Since Ω has a finite number of configurations, ϵ is also finite. Therefore, given any configuration with the proposed controller, $\Delta\mathcal{H}_L$ is either 0 or less than ϵ. If $\Delta\mathcal{H}_L = 0$, the configuration is in M. If not, the configuration converges to M within a finite number of runs $T < \frac{\mathcal{H}_L - \mathcal{H}_0}{\epsilon}$, because the cost function decreases at least by ϵ and it is lower bounded by $\mathcal{H}_0$.

Since M is invariant, any configuration in Ω converges to the critical configuration. □

4.4. *Implementation and evaluation in simulation*

We implemented Algorithms 1 and 2 and tested them on a suit of regularly shaped graph topologies as well as general graph test sets designed for p-median problem. To our knowledge, our work is the first approach to test a control algorithm for mobile robots on the general graph test sets. In all tests, the initial configurations of the robots are randomly selected. We show the statistical performance of our partitioning algorithms and the resultant partitions. Unfortunately, the general graph test sets[20] do not provide any topological information.

4.4.1. Evaluation on regularly spaced graphs. We report on the results of the implementation of two graphs with similar topology but different sizes representing blue prints of bridge structures. The first structure shown in Fig. 4(a) has 144 nodes and 240 edges. The second structure is shown in Fig. 6(a). It has 384 nodes and 649 edges. Coverage by 2 to 10 robots is tested for the small bridge graph, while coverage by 2 to 15 robots are simulated for the big bridge graph. For each robot team size we simulate the algorithm 20 times using randomly initialized configurations. The simulations terminate when the Voronoi partitions no longer change.

Note that the first structure represents a mesh-network of the non-convex shape with the uniform density function shown as the yellow region in Fig. 5. Therefore, the resultant partitions are approximated partitions from continuous distributed coverage which does not exist yet. The finer mesh network will yield more precise approximation.

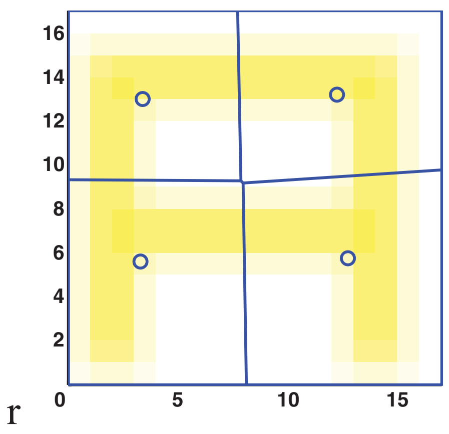
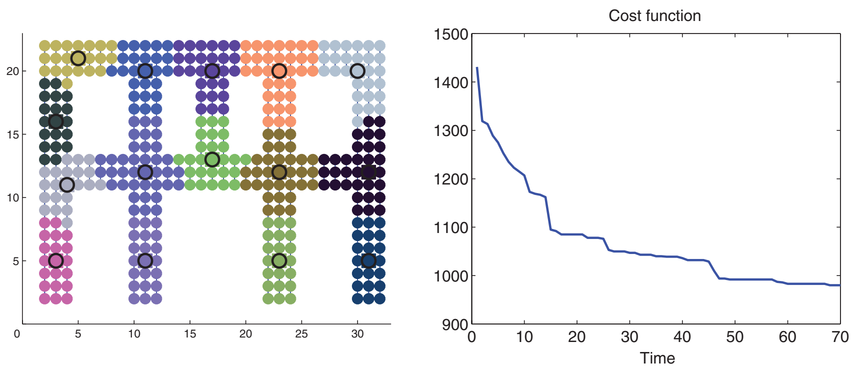


Fig. 5. (Colour online) The resultant Voronoi regions by locational optimization with the continuous density function. The yellow region denotes high-density area, while the white region has low density. The blue circles are robots.



(a) (b)

Fig. 6. (Colour online) Simulation result from 15 robot coverage on the big bridge. (a) The final configuration of locational optimization on the big bridge. (b) The cost function $\mathcal{H}_L$.

Figure 4(a) shows the resultant Voronoi regions of the small bridge obtained using Algorithm 2 with four robots. The graph of the cost function is shown in Fig. 4(b). We see that the cost function decreases over time. By comparison, Fig. 5 shows the solution computed by a distributed coverage controller for a continuous domain, where robots move not only on the target structure but also in free space (white region). The weighting function is continuously defined by interpolating the node weights. The distributed controller in Cortes *et al.*[1] is used for distributed coverage in Fig. 5. The cost function for the continuous domain is shown in Eq. (4). The final locations of the robots look almost identical. However, we can clearly see that our algorithm for graph coverage ensures fully connected V_i and neighbors whose regions are physically connected. The distributed controller in the continuous domain may result in V_i with separated parts and physically non-connected neighbors. In Fig. 5, we can see that the upper-right and the lower-left robots are neighbors, although they are not connected by the target structure.

Figure 6(a) shows the final Voronoi regions computed by Algorithm 2 with 15 robots. The result matches our intuition to locate the robots at the joints of the bridge.

We used two centralized methods to compare our solution with centralized solutions capable of computing global optima: integer programming and Lagrangian relaxation heuristics. Integer programming could not handle the problem size. We used MATLAB and SCIP 1.2.0[5] on 64bit Quad CPU Q9550. The software failed to compute the global optimum even for the smaller graph in Fig. 4(a) because the computation load was too high.

[5] SCIP claims, it is currently one of the fastest non-commercial mixed integer programming (MIP) solvers.

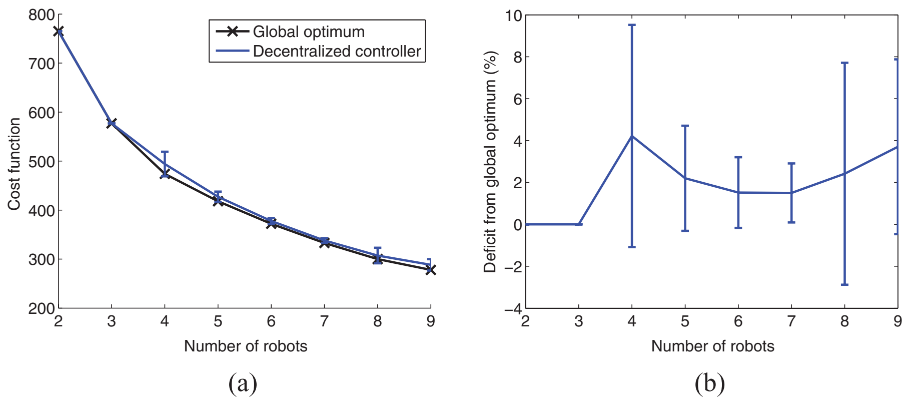


Fig. 7. (Colour online) Performance comparison with the global optimum for the graph in Fig. 4(a). Data are obtained from locational optimization on the big bridge by 2–15 robots. (a) The global optimum and the resultant cost function value from the locational optimization controller. (b) Percentage of deviation from the global optimum. Mean and error-bars (unit σ) are shown.

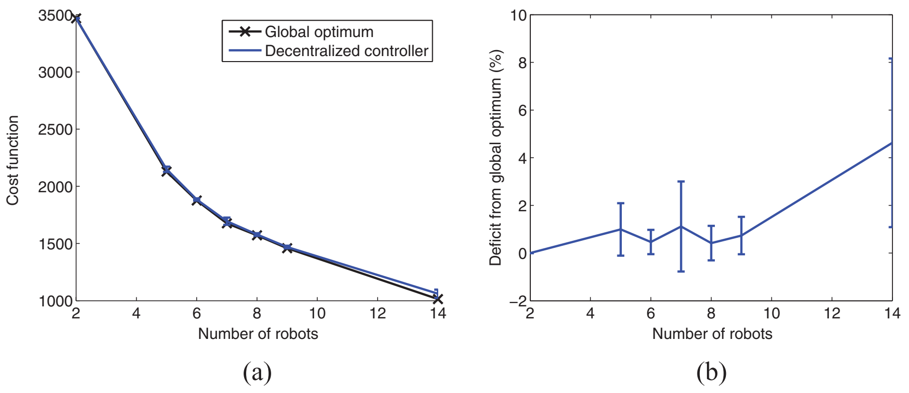


Fig. 8. (Colour online) Performance comparison with the global optimum for the graph in Fig. 6(a). Data are obtained from locational optimization on the big bridge by 2–15 robots. (a) The global optimum and the resultant cost function value from the locational optimization controller. (b) Percentage of deviation from the global optimum. Mean and error-bars are shown.

Lagrangian relaxation does not guarantee the computation of a global optimum, but once it finds a solution, it can determine whether it is a global optimum or not. We used this method to evaluate the solutions computed by our method. Lagrangian relaxation produces the global optima for eight times out of nine cases for the graph in Fig. 4(a), and for seven of 14 cases we tested for the graph in Fig. 6(a).

Figures 7 and 8 compare the solution computed with Algorithm 2 with the results using Lagrangian relaxation (centralized method) and identify the distance between the local and global optima. Deviation from the guessed global minimum slowly increases as the number of robots increases, however it remains within 10%.

4.4.2. Evaluation on generic graphs. In order to evaluate the performance on a general graph, we implement our algorithm on 25 graphs in OR library,[20] which have been used for the p-median problem. The tested graphs have six sets, each of which has three to five graphs with the same number of nodes and edges but different topologies. First four graphs are the smallest (100 nodes and 200 edges) in the OR library, and the last three graphs are the largest (900 nodes and 16,200 edges). Our algorithm has been run for 20 times for each graph. The performance is shown in Table. I.[6] $|Q|$ is the number of nodes, $|E|$ is the number of edges, n is the number of robots, and $\mathcal{H}_0$ is the optimal cost.

In most cases, the average deviation from the global optimum stays within 10% unless n is over roughly 30% of $|Q|$. The larger deficit from a larger n is expected since the robots can be easily

[6] Since the OR library does not give any locational information of the nodes, we can not display the partitions.

Table I. Performance comparison with the global optimum for the graph in OR library.[20]

Filename in OR library	$\lvert Q \rvert$	$\lvert E \rvert$	n	$\mathcal{H}_0$	Average cost	Standard deviation	Average deviation from the global optimum	Average iterations per robot
pmed1.txt	100	200	5	5819	5970	4.7%	2.5%	1.9
pmed2.txt			10	4250	4415	3.8%	3.8%	1.6
pmed3.txt			20	3034	3321	5.0%	9.5%	1.0
pmed4.txt			33	1355	1628	5.3%	20.1%	0.7
pmed6.txt	200	800	5	7824	7824	0.0%	0.0%	2.3
pmed7.txt			10	5631	5728	1.8%	1.7%	1.9
pmed8.txt			20	4445	4732	3.0%	6.4%	1.6
pmed9.txt			40	2734	3013	3.0%	10.2%	1.0
pmed11.txt	300	1800	5	7696	7787	1.9%	1.2%	2.7
pmed12.txt			10	6634	6788	1.7%	2.3%	2.0
pmed13.txt			30	4374	4655	2.1%	6.4%	1.4
pmed14.txt			60	2968	3226	2.1%	8.7%	1.0
pmed15.txt			100	1729	2033	4.1%	17.6%	0.6
pmed16.txt	400	3200	5	8162	8198	1.5%	0.4%	2.7
pmed17.txt			10	6999	7195	2.2%	2.8%	2.0
pmed18.txt			40	4809	5063	1.3%	5.2%	1.4
pmed19.txt			80	2845	3133	1.8%	10.1%	1.0
pmed20.txt			133	1789	2108	3.2%	17.8%	0.6
pmed21.txt	500	5000	5	9138	9289	2.1%	1.6%	2.0
pmed22.txt			10	8577	8680	1.6%	1.1%	2.3
pmed23.txt			50	4619	4870	1.8%	5.4%	1.3
pmed24.txt			100	2961	3269	1.3%	10.0%	0.9
pmed38.txt	900	16,200	5	11,060	11087	0.9%	0.2%	2.5
pmed39.txt			10	9423	9464	0.9%	0.4%	2.4
pmed40.txt			90	5128	5226	1.3%	5.8%	1.3

stuck to local minimum when only a small number of nodes (small number of choices for the next move) belong to each robot. In terms of convergence speed, the algorithm converges within just three iterations per robot in all the cases whereas the Lagrangian relaxation heuristics often requires several hundreds of iterations, each of which may take very long time without a guarantee of finding the global optimum .[26]

5. Application to Equal-Mass Partitioning

Using a different goal with a corresponding cost function, we can extend and apply Algorithms 1 and 2 to equal-mass partitioning. In equal-mass partitioning we wish to partition an environment so that all partitions have the same weight. The environment has an associated weighting function. This problem is important in decentralized construction where we seek to identify sub-assemblies that can be aggregated in approximately the same time period,[4] and in vehicle routing where we want each vehicle to cover the same workload along its route.[3] In this section we introduce the problem, describe the distributed algorithm, and evaluate the algorithms by simulations and experiments.

5.1. Equal-mass partitioning

The goal of equal-mass partitioning is to divide a given space into components with equal amount of workload. Mass can be viewed as the physical measure of the weight associated with each region, or as an abstract measure. Prior centralized solutions include Baron *et al.*[27] Recently, two groups introduced distributed controllers for equal-mass partitioning in the continuous domain.[3,4]

Equal-mass partitioning is related to *graph partitioning*, where the goal is to find subsets of a graph with equal node weights and minimum total weights of edges crossing between subsets. Graph partitioning is also NP-hard,[28] and has many heuristic algorithms.[16] Distributed and geometric solutions were proposed by Refs. [29–32]. Our work specializes this problem by adding the following two constraints:

1. Each graph node belongs to the nearest robot.
2. A robot can relocate itself only in its Voronoi region.

Many existing algorithms either arbitrarily assign a node to a partition (robot) or relocate centroids to any nodes. In the equal partition problem it is not important that we obtain the minimum edge cut, since the cost of the edge cut does not affect the cost function that drives the controller. We focus on dividing a graph into subsets with equal node weights.

More formally, given the Voronoi partition V_i, we define its mass as the sum of the target density function in the area,

$$M_{V_i} = \sum_{V_i} \phi(q). \tag{5}$$

If all the nodes have the same unit node weight ϕ, then M_{V_i} is the number of nodes in V_i. The cost function is given by:

$$\mathcal{H}_E = \sum_{i=1}^{n} \frac{1}{M_{V_i}}. \tag{6}$$

Note that $\mathcal{H}_E$ is minimized only if $M_{V_1} = M_{V_2} = \ldots = M_{V_n}$.

5.2. *Control algorithm*

The distributed vertex substitution algorithm is also used for equal-mass partitioning. The setup of the algorithm inherits Algorithm 2, and the local cost function $\mathcal{H}_{E_i}$ is defined as:

$$\mathcal{H}_{E_i} = \sum_{l=\in\{i\}\cup\mathcal{N}_i} \frac{1}{M_{V_l}}. \tag{7}$$

Note that decay of $\mathcal{H}_{E_i}$ directly leads to the decay of the total cost function $\mathcal{H}_E$. We check how the Voronoi regions change. The change to the masses is ${}_j\Delta_l$, where $l = i, l \in \mathcal{N}_i$. For equal-mass partitioning, lines 10–12 and 19 of Algorithm 2 are replaced by ${}_j\Delta_i \leftarrow_j \Delta_i \pm \phi(q_j)$, where ${}_j\Delta_i$ is the change of the mass for robot i. Its sign is decided according to gain or loss of mass. Let $\hat{\mathcal{H}}_{E_i}$ be the changed $\mathcal{H}_{E_i}$ by substituting p_i^* to q_b. The optimal node for substitution is chosen so that it minimizes $\hat{\mathcal{H}}_{E_i}$. The substitution to q_b that may lead to change $V_{\mathcal{N}_{\mathcal{N}_i}}$ is discarded as in Algorithm 2. The minimum is smaller than $\mathcal{H}_{E_i}$, otherwise the algorithm returns null.

5.3. *Implementation and evaluation in simulation*

The vertex substitution algorithm for equal-mass partitioning was implemented and tested on a suite of graphs, including the graphs in Figs. 4 and 6, and the generic graphs in OR library.[20]

5.3.1. Evaluation on regularly spaced graphs. Originally our continuous version of equal-mass partitioning controller[4] was developed for equally distributing workload to a team of robots for construction (see Fig. 2, which likely consists of a number of regular structures). Therefore, we start the evaluation of the algorithm on two regularly shaped graphs representing feasible blue prints of bridge-like structures. Figure 9 shows the resultant partitions and the data for the small bridge. We see the masses converge to approximately the same value as in Fig. 9(b). The resultant Voronoi regions from the distributed controller in a continuous domain is shown in Fig. 10,[4] and they look similar as well. Note that Algorithm 2 guarantees that V_i is fully connected, while the distributed controller in Yun[4] does not.

Figure 11(a) shows the final Voronoi regions by the equal-mass partitioning controller with 15 robots on the big bridge. The masses converge as shown in Fig. 11(b).

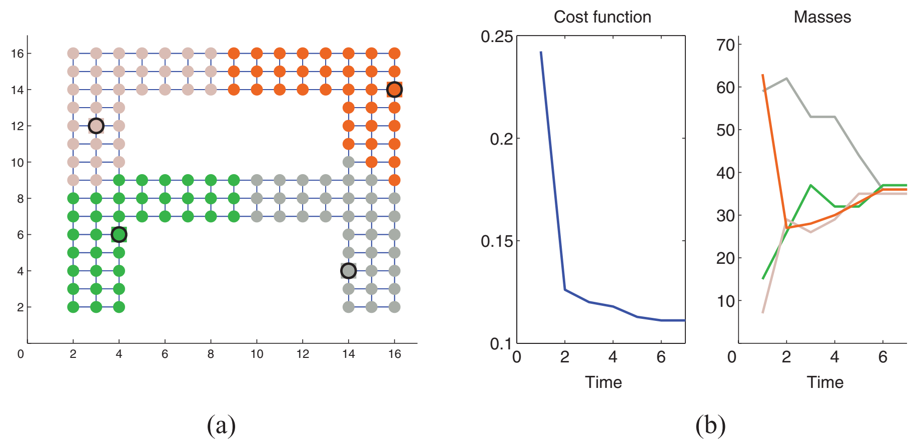


Fig. 9. (Colour online) Simulation result from four robot coverage on the small bridge. (a) The final configuration of equal-mass partitioning on the small bridge. (b) The cost function $\mathcal{H}_E$ and the masses M_{V_i}.

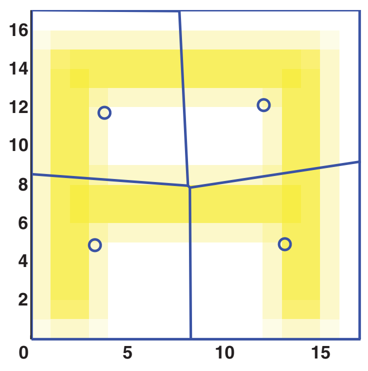


Fig. 10. (Colour online) The resultant Voronoi regions by equal-mass partitioning with the continuous density function. The distributed controller proposed in Yun *et al.*[4] is used.

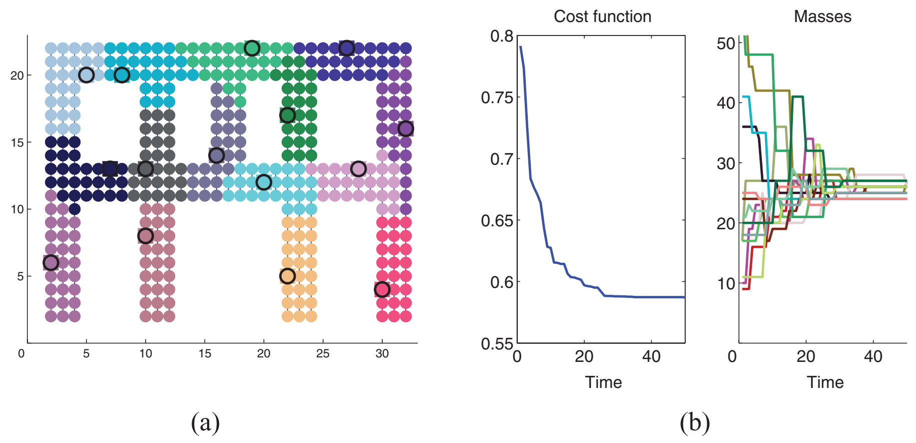
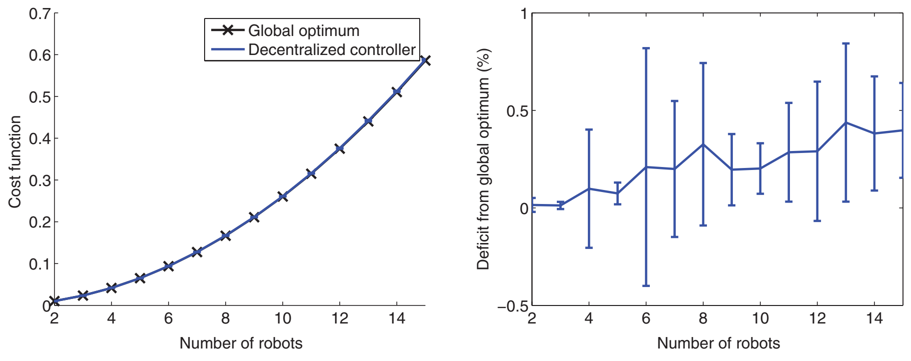


Fig. 11. (Colour online) Simulation result from 15 robot coverage on the small bridge. (a) The final configuration of equal-mass partitioning on the small bridge. (b) The cost function $\mathcal{H}_E$ and the masses M_{V_i}.

For the equal-mass partitioning problem, we know the lower bound of the global optimum of $\mathcal{H}_E$:

$$\mathcal{H}_{E_{\text{opt}}} = \frac{n^2}{M_G}, \tag{8}$$



(a) (b)

Fig. 12. (Colour online) Performance comparison with the global optimum. Equal-mass partitioning on the big bridge is implemented with 2–15 robots. (a) The global optimum and the resultant cost function from the equal-mass partitioning controller. (b) Percentage of deviation from the global optimum. Mean and error-bars are shown.

where M_G is the total mass of G.[7] Therefore, we can compare the result from our controller with $\mathcal{H}_{E_{\text{opt}}}$ as in Fig. 12. The two plots for the *ideal* global optimum and for our controller are almost identical. Figure 12(b) confirms that the deviation from the global optimum is less than 1% and is independent of how many robots were used in the test.

5.3.2. Evaluation on generic graphs. The same graphs for the p-median problem in OR library[20] are used for the evaluation of our equal-mass partitioning algorithm.[8] We use the graphs which have more than 10 nodes per robot in order to ensure enough number of node assignments for each robot. We assume a unit weight to each node. The equal-mass partitioning algorithm has been run 100 times for each graph. The performance is shown in Table. II, in which $\frac{M_G}{n}$ is the ideal optimal mass for each robot, "average deviation" is the average of mass deviations from the optimal mass, and "std. dev. of avg. dev." is the standard deviation of the average deviation.

In most cases, the average deviation from the global optimum stays within a very small amount of node weights. The small standard deviations prove the stability of the control algorithm. As in Table I, the algorithm converges within two iterations per robot in most of the cases.

5.4. Hardware experiment for equal-mass partitioning

Our hardware system consists of four iCreate mobile robots as shown in Fig. 13. Specifications of each component are in Table III. We do not use the arm in this experiment. Poses of all the robots are captured by a Vicon motion capture system which broadcasts 3D poses over a mesh network. The robot has three communication protocols: IR, UDP, and xBee, which are used for communication with the smart parts, other robots, and motion capture system, respectively. We equipped each robot with a small Dell Inspiron Mini 10s netbook which runs a Java-based controller. The robots receive precise location information from a Vicon motion capture system providing the 2D positions and the rotational heading with accuracy to the millimeter and milli-radian respectively at 10 Hz using a commercial xBee radio frequency (RF) wireless mesh network. Between the robots, a UDP multicast channel on the local network is implemented with a singe WLAN router. The UDP packets contain a logical time-stamp, a robot ID number, their current positions, and their current target robot. The robots also broadcast their states.

A* navigation algorithm, which updates every second, has the robots navigate to approach a destination node. More details of the system can be found in Bolger *et al.*[21]

Using this four-robot hardware platform we implemented the equal-mass partitioning algorithm and evaluated it using two classes of graphs: a planar square graph consisting of 900 nodes (see Fig. 14) and a planar graph that captures the geometry of an A-shaped bridge (and thus has geometric concavities) consisting of 724 nodes (see Fig. 16). The size of each of these structures is approximately

[7] It is possible that the configuration with $\mathcal{H}_{E_{\text{opt}}}$ does not exist.
[8] Note that there is no example graph set for equal-mass partitioning since this is a new problem.

Table II. Equal-mass partitioning: performance comparison with the global optimum for the graph in OR library.[20]

Filename in OR library	$\|Q\|$	$\|E\|$	n	$\frac{M_G}{n}$	Average deviation	Std. dev. of avg. dev.	Average iterations per robot
pmed1.txt	100	200	5	20	1.3	0.7	1.6
pmed2.txt			10	10	1.3	0.8	1.6
pmed6.txt			5	40	2.2	2.1	1.7
pmed7.txt	200	800	10	20	2.2	1.4	2.1
pmed8.txt			20	10	2.1	0.8	1.7
pmed11.txt			5	60	3.1	4.4	2.0
pmed12.txt	300	1800	10	30	3.1	3.2	1.8
pmed13.txt			30	10	2.2	0.5	2.0
pmed16.txt			5	80	4.1	7.1	1.9
pmed17.txt	400	3200	10	40	2.7	2.1	1.9
pmed18.txt			40	10	2.0	0.5	1.9
pmed21.txt			5	100	3.1	1.4	1.75
pmed22.txt	500	5000	10	50	2.8	1.7	1.9
pmed23.txt			50	10	2.2	0.4	1.9
pmed26.txt			5	120	3.3	1.5	1.8
pmed27.txt	600	7200	10	60	3.5	2.6	1.9
pmed28.txt			60	10	2.2	0.5	1.9
pmed31.txt			5	140	3.5	1.2	1.8
pmed32.txt	700	9800	10	70	3.2	1.0	1.9
pmed33.txt			70	10	1.8	0.4	1.9
pmed35.txt			5	160	5.5	12.1	1.8
pmed36.txt	800	12,800	10	80	4.0	3.6	1.9
pmed37.txt			80	10	1.9	0.3	1.9
pmed38.txt			5	180	5.1	9.8	1.8
pmed39.txt	900	16,200	10	90	4.3	4.0	1.9
pmed40.txt			90	10	1.9	0.4	1.9

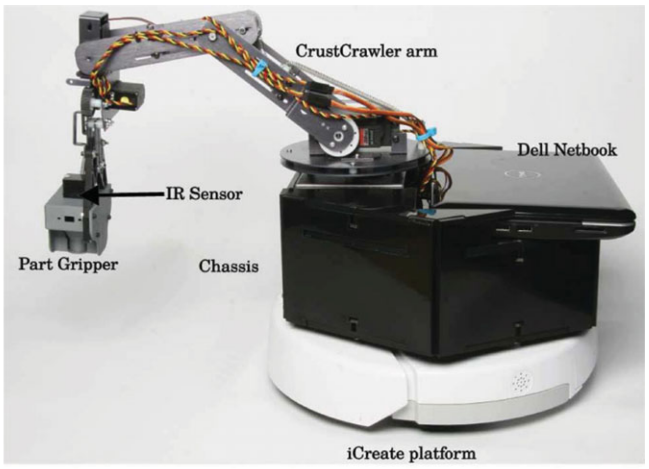


Fig. 13. (Colour online) Side view of robot hardware.

3×3 m. For each graph example, we divided the space into grid cells using a grid size of 0.1 m. Each grid point is a node in the corresponding graph. Nodes are connected when they are adjacent, either by side or by diagonal. Each node has a unit weight. The bridge has two *virtual* empty spaces as shown in the right figures of Fig. 16. For equal-mass partitioning, the robots broadcast their centroid when they update the centroid.

Table III. Specifications of the robot.

Mobile		iRobot iCreate
Arm	Model	CrustCrawler SG5-UT
	DoF	4
	Reach	0.5 m
	Payload	0.6 kg
Communication		IR, UDP, xBee

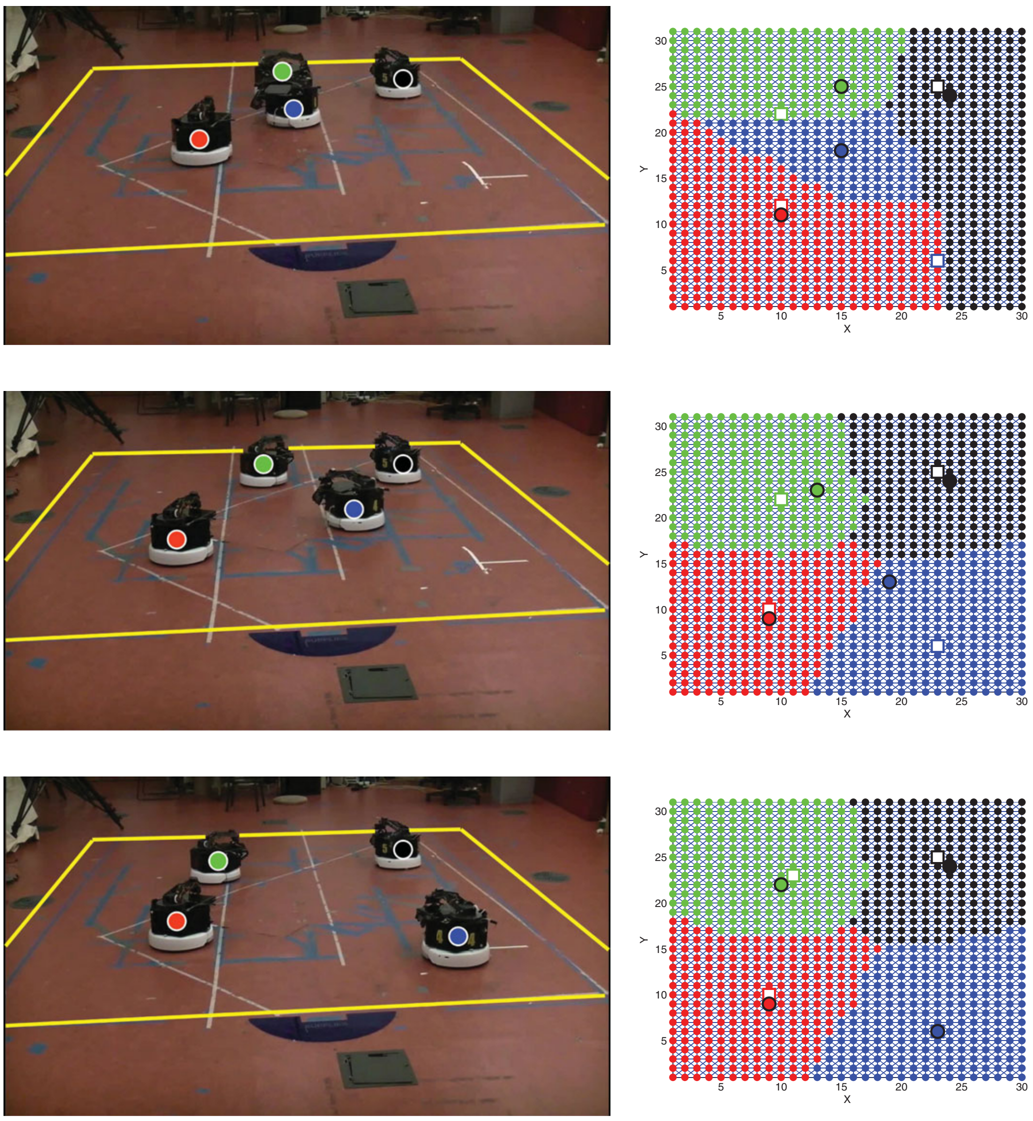


Fig. 14. (Colour online) Snapshots of equal-mass partitioning on a graph representing the square region enclosed by yellow lines. The right figures show the partitions, including nodes (circles), edges (lines), robot locations (bold circles), and target locations (squares). Each color represents a partition which belongs to the robot with the same color. This experiment took about 50 sec.

Figures 15 and 17 show convergence of the masses during the experiments.

6. Conclusions

In this paper we describe two algorithms for distributed coverage on graph. The algorithms enable mobile robots to cover a target graph with minimizing the cost functions. The target graph represents

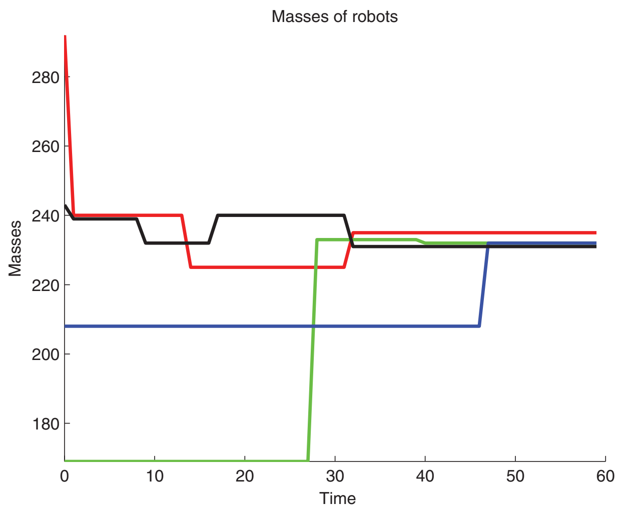


Fig. 15. (Colour online) Masses of four robots during the experiment. All of them converge to a single value.

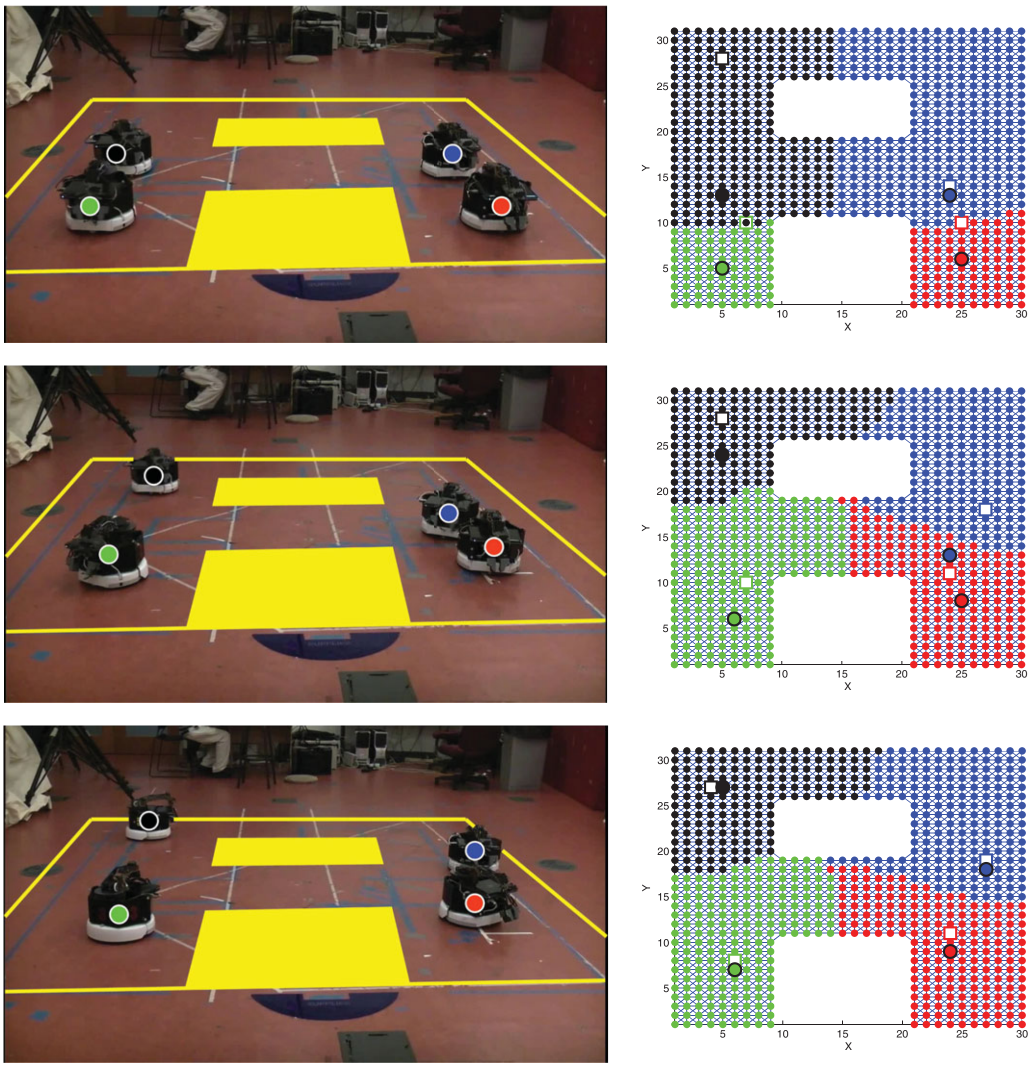


Fig. 16. (Colour online) Snapshots from a hardware execution of equal-mass partitioning with a planar bridge-like structure provided as input. The desired boundary and the free space are drawn in yellow on the left; the corresponding partitions are drawn on the right. This experiment took about 35 sec.

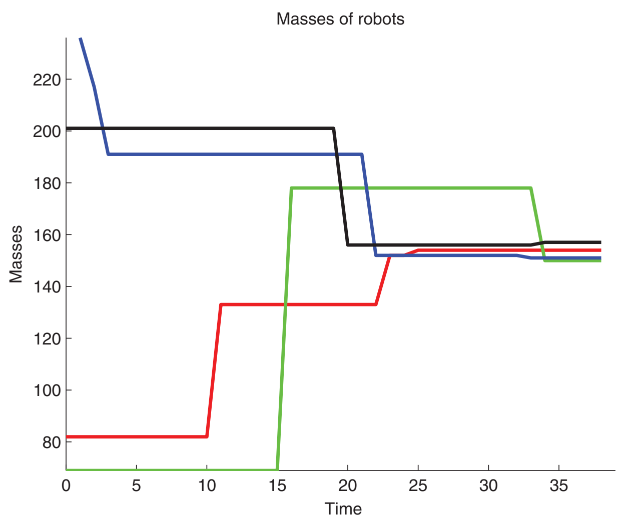


Fig. 17. (Colour online) Masses of four robots during the experiment of partitioning the A-shaped bridge.

environments that can be inherently modeled by a set of nodes and edges or non-convex region. Two cost functions for locational optimization and equal-mass partitioning are used and the corresponding algorithms are designed based on vertex substitution. The algorithms construct Voronoi tessellation based on positions of the robots, and find the optimal next positions. We show that two-hop communication is necessary for the convergence of the algorithms. We have implemented the algorithms and evaluated them in both simulation and hardware and tested their performance for a variety of graphs encoding several types of environments. Our implementations show that the graph-coverage algorithms work correctly and achieve the partitioning of the environment in an effective and efficient way. This graph partitioning algorithm was used for mobile robots tasked with distributed assembly.[21]

6.1. Lessons learned

We have found out that the proposed algorithms performed considerably better for the regularly shaped graphs than the generic graphs. This suggests that the graph-coverage algorithm can be a good fit for coordinated construction, where the target structure likely comprises many regularly shaped components such as trusses and connectors.[21] The performance data that we collected empirically for different graphs with different number of robots allocated to the coverage task on these graphs suggest a critical ratio of robots to graph size. As we increase the number of robots allocated to the task of covering a given graph, we observed that the performance increases up to this critical ratio and then begins to decrease. We believe this is caused by a limited choice of the next node for each robot, but further investigation is necessary. As such, there are environments for which a more centralized approach to give more choices for the robots might improve performance. For example, we can extend the number of hops for communication and let the robot choose the next node not only in its own partition but also in the partitions of its multi-hopped neighbors. The challenge is to identify how many hops to consider while guaranteeing convergence. We may also use such an extended communication range occasionally to rescue the robots from local minima at the cost of additional communication, since undesirable local minima was inevitable time to time. Guaranteeing the global optimum has been the ultimate goal for distributed coverage, and our algorithms are also in the same boat.

6.2. Future direction and extension

The proposed algorithms can be applied to any graph. Since our target application is coordinated construction, graphs are our primary targets, and the target graphs are most likely planar or 3D graphs converted from a continuous non-convex region or a certain target structure. A well-modeled graph that captures the essence of the model and is as sparse as possible will result in efficient performance. However, while the regularly spaced graphs were very good fit for our algorithms, we conjecture that biasing the node allocation such that we have sparse node assignments for large open regions and dense assignments on a bottleneck-like regions may provide faster calculation as well as lower chance of being stuck in (unbalanced) local minima. We will investigate the finite element methods (FEM) literature[33] to better understand this idea.

We prove the algorithms converge to local minima; however, we have not discussed the quality of the converged configuration. In an extreme case, even the global minimum may not exist when we use Voronoi tessellation.

Another thing we should consider is a range of communication if we do not have enough assembly robots. Partitioning while maintaining connectivity may be challenging.

Development of a completely asynchronous controller is an imminent future work. Local handshake among neighboring robots can be an easy fix; however, it will still involve certain amount of synchronization with heavy communication load. Efficient and stochastically provable asynchronous algorithm will be a good solution.

Acknowledgments

This research was done at the Distributed Robotics Laboratory, at CSAIL, MIT. This project has been supported in part by The Boeing Company, the US National Science Foundation, NSF grant number IIS-0426838, Emerging Frontiers in Research and Innovation (EFRI) grant #0735953, MURI SMARTS grant #N0014-09-1051, MURI ANTIDOTE grant #138802, and MURI SWARMS grant #544252. Seung-kook Yun is supported in part by Samsung Scholarship. We are grateful for this support.

References

1. J. Cortes, S. Martinez, T. Karatas and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.* **20**(2), 243–255 (2004).
2. M. Schwager, D. Rus and J.-J. E. Slotine, "Decentralized, adaptive control for coverage with networked robots," *Int. J. Robot. Res.* **28**(3), 357–375 (Mar. 2009).
3. M. Pavone, E. Frazzoli and F. Bullo, "Distributed Algorithms for Equitable Partitioning Policies: Theory and Applications," *IEEE Conference on Decision and Control*, Cancun, Mexico (Dec. 2008), pp. 4191–4197.
4. S.-kook Yun, M. Schwager and D. Rus, "Coordinating Construction of Truss Structures Using Distributed Equal-Mass Partitioning," *Proceedings of the 14th International Symposium on Robotics Research*, Lucern, Switzerland (Aug. 2009).
5. H. Choset, "Coverage for robotics – a survey of recent results," *Ann. Math. Arti. Intell.* **31** , 113–126 (2001).
6. H. Choset and P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," *Proceedings of the International Conference on Field and Service Robotics* (1997).
7. S. Hert and V. Lumelsky, "Polygon area decomposition for multiple-robot workspace division," *Int. J. Comput. Geom. Appl.* **8**, 437–466 (1998).
8. L. C. A. Pimenta, V. Kumar, R. C. Mesquita and G. A. S. Pereira, "Sensing and Coverage for a Network of Heterogeneous Robots," *IEEE Conference on Decision and Control*, Cancun, Mexico (Dec. 2008), pp. 3947–3952.
9. A. Kwok and S. Martínez, "Energy-Balancing Cooperative Strategies for Sensor Deployment," *IEEE International Conference on Decision and Control*, New Orleans, LA (Dec. 2007) pp. 6136–6141.
10. A. Ganguli, J. Cortes and F. Bullo, "Distributed Deployment of Asynchronous Guards in Art Galleries," *American Control Conference*, Minneapolis, MN (Jun. 2006), pp. 1416–1421.
11. C. H. Caicedo-Nunez and M. Zefran, "A Coverage Algorithm for a Class of Non-Convex Regions," *IEEE International Conference on Decision and Control*, Cancun, Mexico (Dec. 2008) pp. 4244–4249.
12. N. Ayanian and V. Kumar, "Decentralized Feedback Controllers for Multi-Agent Teams in Environments with Obstacles," *IEEE International Conference on Robotics and Automation*, Pasadena, CA (May 2008) pp. 1936–1941.
13. S. Bhattacharya, N. Michael and V. Kumar, "Distributed Coverage and Exploration in Unknown Non-Convex Environments," *10th International Symposium on Distributed Autonomous Robots*, Nov. 1–3 (Springer, New York, NY, 2010).
14. M. B. Teitz and P. Bart, "Heuristic methods for estimating the generalized vertex median of a weighted graph," *Oper. Res.* **16**, 955–961 (1968).
15. J. Reese, "Solution methods for the p-median problem: An annotated bibliography," *Networks* **48**, 125–142 (2006).
16. P.-O. Fjallstrom, "Algorithms for graph partitioning: A survey," *Linkoping Electronic Articles in Computer and Information Science*, Vol. 3, article no. 10, available at http://www.ep.liu.se/ea/cis/1998/010/ (1998), online.

17. Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Ann. Math. Arti. Intell.* **31**(1–4), 77–98 (2001).
18. J. W. Durham, R. Carli, P. Frasca and F. Bullo, "Discrete Partitioning and Coverage Control with Gossip Communication," *ASME Dynamic Systems and Control Conference*, Hollywood, CA (Oct. 2009) pp. 225–232.
19. J. Durham, R. Carli, P. Frasca and F. Bullo, "Discrete partitioning and coverage control for gossiping robots," *IEEE Trans. Robot.* **28**(2), 364–378 (2012).
20. J. E. Beasley, "OR-library: Distributing test problems by electronic mail," *J. Oper. Res. Soc.* **41**(11), 1069–1072 (1990).
21. A. Bolger, M. Faulkner, D. Stein, L. White, S.-kook Yun and D. Rus, "Experiments in Decentralized Robot Construction with Tool Delivery and Assembly Robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, China (Oct. 2010) pp. 5085–5092.
22. M. Erwig and F. Hagen, "The graph voronoi diagram with applications," *Networks* **36**, 156–163 (2000).
23. R. W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM* **5**(6), 345 (1962).
24. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Series of Books in the Mathematical Sciences). (W. H. Freeman, New York, NY, Jan. 1979).
25. B. J. Julian, M. Schwager, M. Angermann and D. Rus, "A Location-Based Algorithm for Multi-Hopping State Estimates within a Distributed Robot Team," *Proceedings of the International Conference on Field and Service Robotics (FSR 09)*, Cambridge, MA (Jul. 2009).
26. J. Beasley, "Lagrangean heuristics for location problems," *Eur. J. Oper. Res.* **65**(3), 383–399 (1993).
27. O. Baron, O. Berman, D. Krass and Q. Wang, "The equitable location problem on the plane," *Eur. J. Oper. Res.* **183**(2), 578–590 (Dec. 2007).
28. M. R. Garey, D. S. Johnson and L. Stockmeyer, "Some Simplified Np-Complete Problems," *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing (STOC '74)* (ACM, New York, NY, 1974) pp. 47–63.
29. M. J. Berger and S. H. Bokhari, "A partitioning strategy for nonuniform problems on multiprocessors," *IEEE Trans. Comput.* **36**(5), 570–580 (1987).
30. R. Leland and B. Hendrickson, "An Empirical Study of Static Load Balancing Algorithms," *Proceedings of the Scalable High-Performance Computing Conference* (1994) pp. 682–685.
31. A. Vidwans, Y. Kallinderis and V. Venkatakrishnan, "A parallel dynamic load balancing algorithm for 3D adaptive unstructured grids," *AIAA J.* **32**, 497–505 (1993).
32. C. Ozturan, H. L. deCougny, M. S. Shephard and J. E. Flaherty, "Parallel adaptive mesh refinement and redistribution on distributed memory computers," *Comput. Methods Appl. Mech. Engrg, Tech. Rep.* **119**, 123–137 (1993).
33. I. Babuška, U. Banerjee and J. E. Osborn, "Survey of meshless and generalized finite element methods: A unified approach," *Acta Numerica* **12**, 1–125 (2003).