Multi-Robot Forest Coverage for Weighted and Unweighted Terrain

Xiaoming Zheng, Sven Koenig, *Senior Member, IEEE,* David Kempe, and Sonal Jain

***Abstract*—One of the main applications of mobile robots is coverage: visiting each location in known terrain. Coverage is crucial for lawn mowing, cleaning, harvesting, search-and-rescue, intrusion detection and mine clearing. Naturally, coverage can be sped up with multiple robots. However, we show that solving several versions of multi-robot coverage problems with minimal cover times is NP-hard, which provides motivation for designing polynomial-time constant-factor approximation algorithms. We then describe Multi-Robot Forest Coverage (MFC), a new polynomial-time multi-robot coverage algorithm based on an algorithm by Even et al. for finding a tree cover with trees of balanced weights. Our theoretical results show that the cover times of MFC in weighted and unweighted terrain are at most about a factor of 16 larger than minimal. Our simulation results show that the cover times of MFC are close to minimal in all tested scenarios and smaller than the cover times of an alternative multi-robot coverage algorithm.**

***Index Terms*—Approximation Algorithm, Cell Decomposition, Complexity, Multi-Robot Coverage, NP-Hardness, Robot Teams, Spanning Tree Coverage, Terrain Coverage, Tree Cover.**

I. Introduction

COVERAGE requires robots to visit each location in known terrain once to perform some task. Examples include lawn mowing, cleaning, harvesting, search-and-rescue, intrusion detection and mine clearing. It is frequently desirable to minimize the time by which coverage is complete, called the cover time. In recent years, robotics researchers have investigated spanning tree-based coverage algorithms in unweighted terrain, where the travel times of robots are the same everywhere in the terrain. Single-robot coverage problems are solved with minimal cover times by Spanning Tree Coverage (STC), a polynomial-time single-robot coverage algorithm that decomposes terrain into cells, finds a spanning tree of the resulting graph, and makes the robot circumnavigate it [13]. Naturally, coverage can be sped up with multiple robots. We show that solving several versions of multi-robot coverage problems with minimal cover times is NP-hard, which provides motivation for designing polynomial-time constant-factor approximation algorithms. Hazon and Kaminka recently generalized STC to Multi-Robot Spanning Tree Coverage (MSTC), a polynomial-time multi-robot coverage algorithm [18]. While MSTC provably improves the cover times compared to STC, it cannot guarantee its cover times to be small. We generalize STC to Multi-Robot Forest Coverage (MFC), a polynomial multi-robot coverage algorithm based on finding tree covers with trees of balanced weights, one tree for each robot. We then generalize MFC from unweighted terrain to weighted terrain, where the travel times of robots are not the same everywhere, for example, because different terrain properties (such as rock, sand and grass) require different speeds [35] [39]. MFC is nontrivial to generalize because it uses a tree cover algorithm as a subroutine that is specific to unweighted terrain. We thus first generalize the tree cover algorithm and only then MFC. We prove that the cover times of MFC in weighted and unweighted terrain are at most about sixteen times larger than minimal.[1] Our simulation results show that the cover times of MFC are close to minimal in all tested scenarios and smaller than the cover times of MSTC. MFC has the additional benefit that it tends to return the robots close to their initial cells, which facilitates their collection and storage.

Xiaoming Zheng, Sven Koenig and David Kempe are with the Computer Science Department, University of Southern California, Los Angeles, CA 90089, USA. Their emails are {xiaominz, skoenig, dkempe}@usc.edu. Sonal Jain is with Microsoft, US-Windows Client Platform, Redmond, WA 98052, USA. His email is sonalja@microsoft.com.

Manuscript received August 31, 2007.

II. Related Work

Coverage has been investigated extensively in the literature [7]. One line of research investigates ant-like robots that plan locally, mostly to cover *unknown* terrain [33]. Ant-like robots require only a limited amount of memory, computation and communication, yet are able to cover terrain robustly. They are simple to design, cheap to build and easy to program but can result in large cover times [37]. There are three main classes of robots, namely

- robots that leave permanent markings in the terrain [9] [10]. Some of these robots use real-time search to exploit the markings, such as [22]. Other robots use algorithms that preserve the connectivity of the uncovered terrain, such as [16] and [40].
- robots that leave evaporating markings in the terrain. Some of these robots use adaptive algorithms to exploit the markings, such as [41]. Other robots use algorithms inspired by alarm pheromones of ants, such as [29].
- robots that do not leave any markings in the terrain. Some of these robots mimic the behavior of gas flow, such as [36]. Other robots divide the terrain into regions that are covered in parallel by different robots, such as [30].

Rekleitis et al. [34] study how to reduce repeat coverage with ant-like robots, Noborio et al. [31] and Qutub et al. [32] study deadlock free coverage with ant-like robots, and Ichikawa and Hara [19] study coverage after which ant-like robots have to return to their initial locations.

[1]The word "about" indicates the omission of an additive constant.

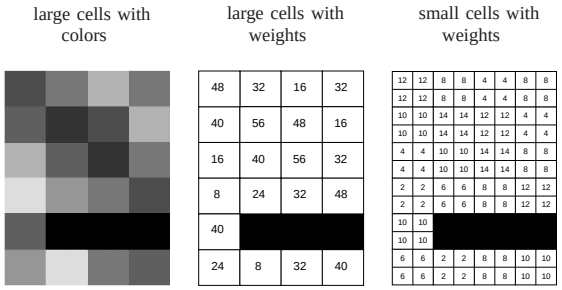


Fig. 1. Model of Weighted Terrain

Another line of research investigates robots that plan globally to cover both *known* and *unknown* terrain. Some robots represent known changing terrain with neural networks so that the uncovered areas attract robots, such as [28]. Some robots coordinate with sensor networks to cover known terrain, such as [15]. However, most robots divide the terrain explicitly into regions that are covered in parallel by different robots [25]. There are two main classes of robots, namely

- robots that use *exact cellular decompositions* to model the unblocked terrain precisely. Some of these robots use trapezoidal decompositions [27] to divide the terrain into regions, such as [2] in known terrain and [6] and [38] in unknown terrain. Other robots use Boustrophedon decompositions [8] (that use Morse functions [1] to determine critical locations indicating changes in the terrain connectivity), such as [24] and [42] in known terrain and [23] and [26] in unknown terrain.
- robots that that use *approximate cellular decompositions* to model the unblocked regions only approximately, by partitioning the terrain into cells of the same shape and size that are either entirely blocked (that is, untraversable) or unblocked (that is, traversable). Some of these robots use the distance wavefront algorithm [20], such as [43] in known terrain.

The above coverage algorithms assume unweighted terrain and often do not provide analytical results on the resulting cover times, which is not surprising for unknown terrain since the cover times can be arbitrarily bad in unknown terrain [26]. The new coverage algorithms introduced in this article use approximate cellular decompositions where the cells are four times the size of the robots. They are (to the best of our knowledge) the first polynomial-time constant-factor approximation algorithms for multi-robot coverage in both weighted and unweighted (known) terrain.

III. Problem Description

We discretize the given known terrain into large square cells. Each large cell is either entirely blocked or entirely unblocked. Robots cannot traverse blocked cells. Each unblocked large cell has a weight that corresponds to how difficult it is to traverse the large cell. Each unblocked large cell is evenly divided into four small square cells. Each small cell has a weight that is equal to one quarter of the weight of the large cell, as shown in Figure 1. Large blocked cells are colored black and large unblocked cells are colored grey, where the levels of grey correspond to the weights. We study two different kinds of terrain.

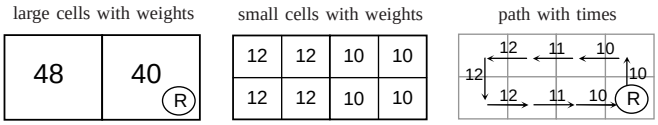


Fig. 2. Single-Robot Coverage Problem with the Team Objective "Cover with Return" in Weighted Terrain

- Terrain is *unweighted* if the weight of each large cell is four, which implies that the weight of each small cell is one.
- Terrain is *weighted* if the weight of each large cell is an arbitrary positive integer.

The robots have the same size as the small cells and start in small cells that belong to different large cells. They always know their current small cell and can move from their current small cell to any adjacent small cell in the four main compass directions without error in a time that is equal to the average of the weights of the two small cells (although our analytical results can easily be adapted to other definitions, such as a time that is equal to the maximum of the weights of the two small cells). Each move is atomic, that is, needs to be executed in full by a robot. The *travel time* of a robot along a path is the sum of the times of its moves and thus equal to the number of its moves in unweighted terrain. We assume that several robots are able to occupy the same small cell simultaneously and thus never block each other. This assumption avoids deadlocks and simplifies our analytical results.

We study two different team objectives.

- The team objective *Cover* requires each small cell to be visited by at least one robot. We want to minimize the *cover time without return*, which is equal to the largest travel time of any robot.
- The team objective *Cover with Return* requires each small cell to be visited by at least one robot and each robot to return to its initial small cell. We want to minimize the *cover time with return*, which again is equal to the largest travel time of any robot.

We simply use "cover time" when we mean "cover time both with and without return." For illustration, Figure 2 shows both a single-robot coverage problem with the team objective "Cover with Return" and one of its solutions, including the large cells with their weights (left), the small cells with their weights (center) and the path with the times of the moves (right). The cover time with return is equal to 88, the sum of the weights of all large cells, which is the minimal cover time with return.

We use several symbols throughout this article.

$w_{\max}$:	the largest weight of any large cell
w_{sum} :	the sum of the weights of all large cells
K :	the set of robots for multi-robot coverage problems and the set of roots for rooted tree cover problems
ϕ :	$w_{\max}/w_{\text{sum}}$
ϵ :	$\phi\lvert K\rvert$

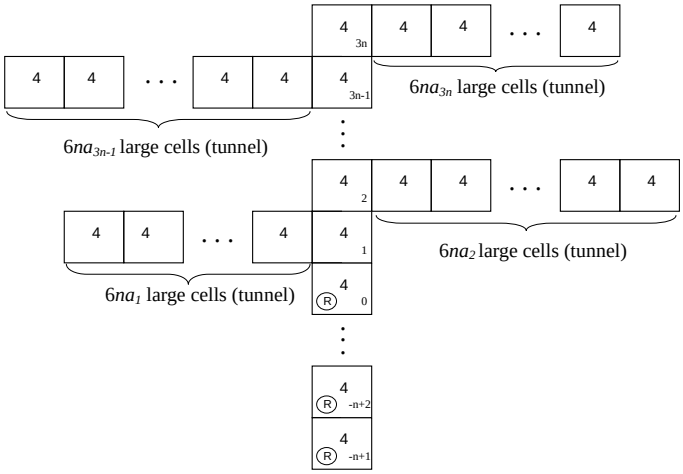


Fig. 3. Multi-Robot Coverage Problem with the Team Objective "Cover with Return" in Unweighted Terrain

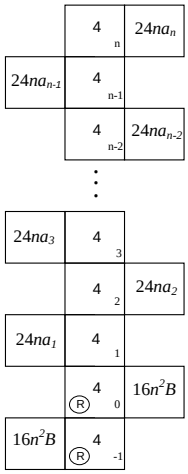


Fig. 4. Multi-Robot Coverage Problem with the Team Objective "Cover without Return" in Weighted Terrain

IV. Complexity of Multi-Robot Coverage

Multi-robot coverage typically results in smaller cover times than single-robot coverage. Unfortunately, we show that solving several versions of multi-robot coverage problems with minimal cover times is NP-hard, which provides motivation for designing polynomial-time constant-factor approximation algorithms.

Theorem 1: It is NP-hard to determine whether the following two versions of multi-robot coverage problems can be solved with cover times with return (for Version 1) or cover times without return (for Version 2) that are smaller than a given value:

- *Version 1:* multi-robot coverage problems with the team objective "Cover with Return" for a number of robots specified in the problem description in unweighted terrain; and
- *Version 2:* multi-robot coverage problems with the team objective "Cover without Return" for two robots in weighted terrain.

Proof: We reduce from partitioning problems to prove the NP-hardness of both versions of multi-robot coverage problems.

- *Version 1:* We reduce from the 3-Partition problem: Given a positive integer B and positive integers $a_1, \ldots, a_{3n}$ strictly between $B/4$ and $B/2$ with $\sum_{i=1}^{3n} a_i = nB$, can they be partitioned evenly into n sets? The 3-Partition problem is strongly NP-hard [14], that is, NP-hard even if the sizes of its integers are only polynomial in n.
  For a given instance of the 3-Partition problem, we construct an instance of the multi-robot coverage problem with n robots, as shown in Figure 3. We start with a "corridor" consisting of $4n$ vertically adjacent unblocked large cells, numbered from $-n+1$ (bottom) to $3n$ (top) as indicated in the lower right corners of the corridor cells in Figure 3. There is a "tunnel" of $6na_i$ horizontally adjacent unblocked large cells for each $i = 1, \ldots, 3n$. The i^{th} tunnel is connected to the i^{th} corridor cell. It is to the left of the corridor for odd i and to the right of the corridor for even i. One robot starts in each of the corridor cells $-n+1, -n+2, \ldots, 0$. This completes the construction, which can be done in polynomial time. We claim that the minimal cover time is at most $24nB+16n$ iff the given integers can be partitioned evenly into n sets.
  "If" direction: Assume that the given integers can be partitioned evenly into n sets $S_1, \ldots, S_n$. Then, we let the j^{th} robot cover the i^{th} tunnel for each $i \in S_j$ and then return to its initial cell. It traverses its tunnels for a travel time of at most $\sum_{i \in S_j} 4 \cdot 6na_i = 24nB$ and the corridor for a travel time of at most $4 \cdot 4n$. Its total travel time is thus at most $24nB + 16n$, which meets the requirement.
  "Only if" direction: Assume that the travel time of each robot is at most $24nB + 16n$. Then, we define S_j to be the set of indices i such that the j^{th} robot is the first robot to cover the upper small cell of the i^{th} tunnel cell that is farthest away from the corridor. These sets partition the given integers. The j^{th} robot needs to traverse its tunnels in both directions. Moving from one tunnel cell to an adjacent tunnel cell requires at least entering two small cells. The total travel time of the j^{th} robot thus is at least $2 \cdot 2 \cdot \sum_{i \in S_j} 6na_i = 24n \sum_{i \in S_j} a_i$. We assumed that the total travel time of each robot is at most $24nB + 16n$, which implies that $24n \sum_{i \in S_j} a_i \leq 24nB + 16n$ or, equivalently, $\sum_{i \in S_j} a_i \leq B + \frac{2}{3}$. But then $\sum_{i \in S_j} a_i \leq B$ since both $\sum_{i \in S_j} a_i$ and B are integers. Thus, the sets S_j partition the given integers evenly since $\sum_{j=1}^{n} \sum_{i \in S_j} a_i = \sum_{i=1}^{3n} a_i = nB$ and thus $\sum_{i \in S_j} a_i = B$.
- *Version 2:* The above construction has to be adapted slightly to prove the NP-hardness of Version 2 of the multi-robot coverage problem. We reduce from the Partition problem: Given at least five positive integers $a_1, \ldots, a_n$ with $\sum_{i=1}^{n} a_i = 2B$, can they be partitioned evenly into two sets? The Partition problem is known to be NP-hard if the integers can be exponential in

n [14]. But then the tunnels of length $6na_i$ from the above construction cannot necessarily be constructed in polynomial time. Instead, weighted terrain allows us to collapse each tunnel to a single unblocked large cell with weight $24na_i$, as shown in Figure 4. Weighted terrain also allows us to prove the NP-hardness of multi-robot coverage problems with the team objective "Cover without Return" by adding one unblocked large "destination" cell of weight $16n^2B$ for each robot to the left of its initial corridor cell. This weight is so large that exiting the destination cell results in large cover times. The corridor consists of $n+2$ unblocked large cells with weight four each, numbered from -1 (bottom) to n (top). One robot starts in each of the corridor cells -1 and 0. This completes the construction, which can be done in polynomial time. We claim that there is a schedule with cover time of at most $56n^2B + 24nB + 4n + 8$ iff the given integers can be partitioned evenly into two sets.

"If" direction: Assume that the given integers can be partitioned evenly into two sets S_1 and S_2. Then, we let the j^{th} robot cover the i^{th} tunnel for each $i \in S_j$ and then move to its destination cell. It thus traverses its tunnels with a travel time of at most $\sum_{i\in S_j} 4 \cdot 6na_i = 4{\cdot}6nB = 24nB$, the corridor with a travel time of at most $4n+8$ and its destination cell with a travel time of at most $3.5 \cdot 16n^2B = 56n^2B$. (The factor of 3.5 results from the fact that the destination cell is not exited.) Its total travel time is thus at most $56n^2B+24nB+4n+8$, which meets the requirement.

"Only if" direction: Assume that the travel time of each robot is at most $56n^2B+24nB+4n+8$. Then, we define S_j to be the set of indices i such that the j^{th} robot is the first robot to cover the upper small cell of the i^{th} tunnel that is farthest away from the corridor. These sets partition the given integers. The j^{th} robot can cover only one destination cell and needs to traverse it with a travel time of at least $56n^2B$. Thus, it has to traverse the tunnels and corridor with a travel time of at most $24nB+4n+8$ and also needs to cover the lower small cell of the i^{th} tunnel that is farthest away from the corridor if $i \in S_j$. If it did not, then each robot would traverse the i^{th} tunnel with a travel time of at least $3 \cdot 6na_i = 18na_i$ each. They would traverse the other tunnels with a combined travel time of at least $\sum_{i'\neq i} 24na_{i'}$. Their combined total travel time would thus be at least $\sum_{i'\neq i} 24na_{i'} + 2 \cdot 18na_i = 24n\sum_{i=1}^{n} a_i + 12na_i = 48nB+12na_i > 48nB+8n+16$ since $n \geq 5$. Thus, at least one of them would have to traverse the tunnels and corridor with a travel time of more than $24nB + 4n + 8$, which is a contradiction. Thus, the j^{th} robot covers both the upper and lower small cell of the i^{th} tunnel that is farthest away from the corridor if $i \in S_j$. It traverses the i^{th} tunnel with a travel time of at least $24na_i$ and all tunnels with a travel time of at least $\sum_{i\in S_j} 24na_i$. But then $\sum_{i\in S_j} a_i \leq B$ since $\sum_{i\in S_j} 24na_i \leq 24nB + 4n + 8$ or, equivalently, $\sum_{i\in S_j} a_i \leq B + 1/6 + 1/(3n)$ and, furthermore, the a_i and B are integers and $n \geq 5$. Thus, the sets S_j partition the given integers evenly since $\sum_{i\in S_j} a_i = B$. ■

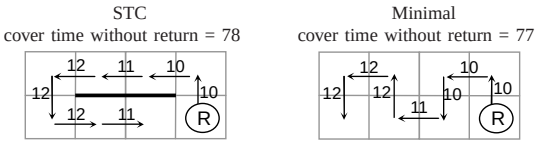


Fig. 5. Suboptimal Cover Time without Return of STC

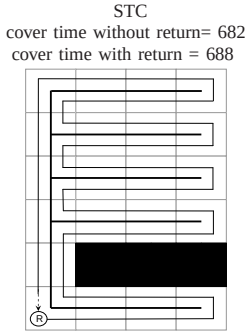


Fig. 6. Example of STC

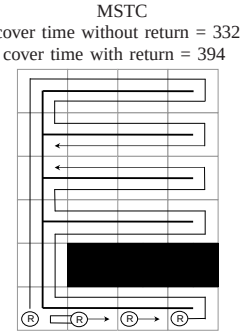


Fig. 7. Example of MSTC

It is currently open whether solving Version 1 of the multi-robot coverage problem is NP-hard for a fixed number of robots (that is, a number of robots that is always the same and thus does not need to be specified in the problem description). We conjecture that this problem is indeed NP-hard. It is currently also on open whether the problem remains NP-hard for the team objective "Cover without Return."

V. Existing Coverage Algorithms

We build on insights provided by the following single-robot and multi-robot coverage algorithms from the literature.

A. Spanning Tree Coverage

Spanning Tree Coverage (STC) [13] is a single-robot coverage algorithm that was originally proposed for unweighted terrain but also applies unchanged to weighted terrain [45]. First, STC constructs a graph whose vertices correspond to the unblocked large cells and whose edges connect adjacent unblocked large cells. This graph needs to be connected. Then, STC finds a spanning tree of this graph in polynomial time. Finally, STC lets the robot move along the path that circumnavigates this spanning tree. For the team objective "Cover with Return," the robot completely circumnavigates the spanning tree until it returns to its initial small cell. For the team objective "Cover without Return," the robot stops once all small cells have been covered, which is one move earlier. Clearly, STC runs in polynomial time. The cover times of STC in unweighted terrain are minimal [13]. The cover times with return of STC in weighted terrain are still minimal. The cover times without return of STC in weighted terrain are not necessarily minimal but are larger than minimal by at most the largest weight of any small cell [45], as shown in Figure 5 for the coverage problem from Figure 2. (The thick line shows the spanning tree.)

For illustration, Figure 6 shows the spanning tree and path for the coverage problem from Figure 1. The cover time without return of STC is 682, The robot has to make one additional move to return to its initial small cell for the team objective "Cover with Return," which is shown with a dashed line. The cover time with return of STC is 688.

B. Multi-Robot Spanning Tree Coverage

STC has been generalized to Multi-Robot Spanning Tree Coverage (MSTC) [18] [45]. MSTC is a multi-robot coverage algorithm for both unweighted terrain [18] and weighted terrain [45] that computes suboptimal cover times in polynomial time, as follows: MSTC computes the same spanning tree as STC and considers the path that circumnavigates the spanning tree. Each robot follows the segment of the path counterclockwise ahead of it, with one exception: To improve the cover times, the longest segment is divided evenly between the two adjacent robots. A few small adjustments, detailed in [18] for unweighted terrain and in [45] for weighted terrain, then ensure that MSTC reduces the cover times without return of STC in unweighted terrain by at least a factor of 2 for three or more robots. For the team objective "Cover with Return," MSTC returns the robots to their initial small cells on paths with minimal travel times once all small cells have been covered.

For illustration, Figure 7 shows the spanning tree and paths for the coverage problem from Figure 1 for four robots. The cover time without return of MSTC is 332. The robots have to return to their initial small cells for the team objective "Cover with Return," which is not shown in the figure. The cover time with return of MSTC is 394. This example demonstrates that the cover times of MSTC do not necessarily improve with an increasing number of robots since MSTC makes only two robots exit the bottom-most row of large cells through the narrow passage. Additional robots in the center of the bottom-most row do not shorten the travel times of these two robots. The cover times of MSTC thus become larger than minimal by an arbitrary factor if one expands the terrain above the narrow passage and adds robots in the center of the bottom-most row since then all of the robots would have to exit the bottom-most row to minimize the cover times. Thus, MSTC cannot guarantee that its cover times are close to minimal or even small.

VI. Multi-Robot Forest Coverage

We now introduce Multi-Robot Forest Coverage (MFC), a new multi-robot coverage algorithm for both unweighted and weighted terrain that is a polynomial-time constant-factor approximation algorithm for computing suboptimal cover times in polynomial time. Remember that MSTC determines one spanning tree, splits the path that circumnavigates it into one path for each robot and lets each robot move along its path. MSTC constructs the tree without taking into account that it will be split afterwards, which results in unbalanced travel times of the robots. MFC, on the other hand, determines one tree for each robot and lets each robot move along the path that circumnavigates its tree, as follows: First, MFC constructs a graph whose vertices correspond to the unblocked large cells and whose edges connect adjacent unblocked large cells. This graph does not need to be connected, as long as each of its components contains at least one vertex that corresponds to the initial large cell of a robot. Then, MFC finds a rooted tree cover of this graph, where the roots are the vertices that correspond to the initial large cells of the robots. A rooted tree cover of this graph is a forest of trees with exactly one tree for each root. The trees can share vertices and edges. Every vertex of the graph has to be contained in at least one tree. Finally, MFC lets each robot move along the path that circumnavigates its tree. For the team objective "Cover with Return," the robots completely circumnavigate their trees until they return to their initial small cells. For the team objective "Cover without Return," the robots stop once all small cells have been covered. Clearly, MFC runs in polynomial time if it can determine a suitable rooted tree cover in polynomial time.

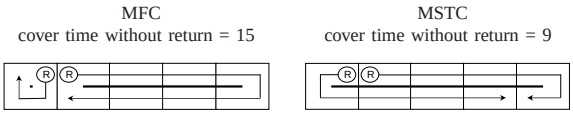


Fig. 9. MFC versus MSTC in Unweighted Terrain

For illustration, Figure 8 shows the trees and paths for the coverage problem from Figure 1 for four robots. The cover time without return of MFC is 225. The robots have to return to their initial small cells for the team objective "Cover with Return," which is shown with dashed lines. The cover time with return of MFC is 256.

A. Unweighted Terrain

In unweighted terrain, we define the weight of a tree to be the number of its edges and the weight of a rooted tree cover to be the largest weight of any of its trees. Finding a weight-minimal rooted tree cover is NP-hard [11]. MFC therefore uses the tree-cover algorithm by Even et al. [11] to find in polynomial time a rooted tree cover with a weight that is at most a factor of 4 larger than minimal. For a single robot, MFC reduces to STC and thus minimizes the cover times. For multiple robots, remember that the cover times without return of MSTC are at least a factor of 2 smaller than those of STC and thus at least a factor of 2 smaller than the minimal ones for a single robot. MFC cannot provide such a strong worst-case guarantee about how small its cover times without return are with respect to the minimal ones without return of a single robot. Figure 9 shows an example of unweighted terrain where the cover time without return of MFC is almost equal to that of STC if the corridor is sufficiently long, even though the cover time without return of MSTC is only half that of STC. However, we now prove that MFC provides a more powerful guarantee than MSTC, namely a worst-case guarantee about how small its cover times are with respect to the minimal ones for the number of available robots.

Theorem 2: The cover times (with and without return) of MFC in unweighted terrain are at most about a factor of 16 larger than minimal.

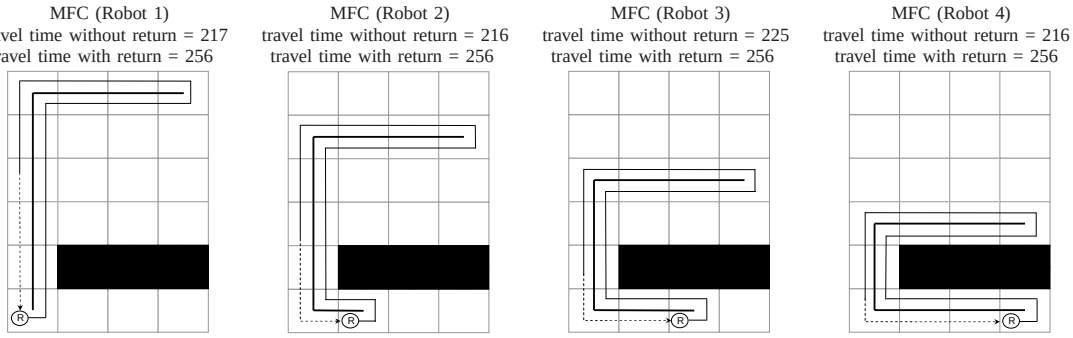


Fig. 8. Example of MFC

Proof: Consider the team objective "Cover without Return." We define M to be the weight of the rooted tree cover found by the tree-cover algorithm by Even et al. [11], N to be the weight of the weight-minimal rooted tree cover, O to be the cover time without return of MFC, P to be the minimal cover time without return and Q to be the minimal cover time without return if the robots only need to cover the upper left small cells of all unblocked large cells. Because circumnavigating a tree of weight M requires entering at most $4M+4$ small cells, we get that $O \leq 4M+4$. The approximation guarantee proved in [11] provides that $M \leq 4N$. Because the weight-minimal rooted tree cover (shifted slightly to the upper left) contains all upper left small cells and thus provides a lower bound on the minimal cover time without return if the robots only need to cover the upper left small cells, we get that $N \leq Q$. Finally, the minimal cover time without return if the robots need to cover only the upper left small cells is at most the minimal cover time without return if the robots need to cover all small cells. Because of this fact we get that $Q \leq P$. Combining all these results yields $O \leq 4M+4 \leq 16N+4 \leq 16Q+4 \leq 16P+4$. The proof continues to hold for the team objective "Cover with Return" if each occurrence of "cover time without return" is replaced with "cover time with return." ■

Thus, MFC provides a more powerful guarantee than MSTC. MFC also has disadvantages. For example, it makes several robots occupy the same small cell at the same time. Then, some robots have to wait for other robots to leave their small cells (or move around them) if our assumption that several robots are able to occupy the same small cell simultaneously is unjustified.

B. Weighted Terrain

In weighted terrain, we define the weight of each vertex to be the weight of the corresponding large cell, the weight of a tree to be the sum of the weights of its vertices and the weight of a rooted tree cover to be the largest weight of any of its trees. Finding a weight-minimal rooted tree cover remains NP-hard as proved in the appendix but the tree-cover algorithm by Even et al. [11] no longer applies. MFC therefore uses a new tree-cover algorithm, TREE COVER, to find in polynomial time a rooted tree cover with a weight that is at most a factor of 4 larger than minimal. We describe this polynomial-time constant-factor approximation algorithm in the appendix.

Theorem 3 (= Theorem 9): TREE COVER can be used to find a K-rooted tree cover of graph G whose weight is at most a factor of $4(1+\epsilon)$ larger than minimal, where $\epsilon = \phi|K|$, $\phi = w_{\max}/w_{\text{sum}}$ and $|K|$ is the number of roots.

We now prove that MFC provides a worst-case guarantee about how small its cover times are with respect to the minimal ones for the number of available robots that is similar to the one in unweighted terrain. We first relate the weight of a weight-minimal rooted tree cover to the minimal cover time.

Lemma 4: The weight of a weight-minimal rooted tree cover divided by four is at most the largest weight of any small cell plus the minimal cover time (with and without return).

Proof: The proof follows the same structure as the proof of Theorem 2. Consider the team objective "Cover without Return." We define N to be the weight of a weight-minimal rooted tree cover, P to be the minimal cover time without return and Q to be the minimal cover time without return if the robots need to cover only the upper left small cells of all large cells. Assume that the robots have to cover only the upper left small cells of all unblocked large cells and that they cover these small cells with minimal cover time without return. Construct a rooted tree cover where the tree of a robot contains exactly the vertices that correspond to the large cells that contain any small cell visited by the robot. The robot has to enter and exit all small cells it visits except possibly for its initial small cell (which it does not need to enter) and its final small cell (which it does not need to exit). Thus, the weight of the tree of a robot divided by four is at most the largest weight of any small cell plus the travel time of the robot, implying that the weight of the given rooted tree cover divided by four is at most the largest weight of any small cell plus the minimal cover time without return if the robots need to cover only the upper left small cells of all large cells, Since the weight of a weight-minimal rooted tree cover is at most the weight of the given rooted tree cover, we get $N/4 \leq Q + w_{\max}/4$. Finally, the minimal cover time without return if the robots need to cover only the upper left small cells is at most the minimal cover time without return if the robots need to cover all small cells. Because of this fact we get that $Q \leq P$. Combining all these results yields $N/4 \leq Q + w_{\max}/4 \leq P + w_{\max}/4$. The proof continues to hold for the team objective "Cover with Return" if each occurrence of "cover time without return" is replaced with "cover time with return." ■

Theorem 5: The cover times (with and without return) of MFC in weighted terrain are at most about a factor of $16(1+\epsilon)$ larger than minimal, where $\epsilon = \phi|K|$, $\phi = w_{\max}/w_{\text{sum}}$ and $|K|$ is the number of robots.

Proof: Consider either the team objective "Cover without Return" or "Cover with Return." We define M to be the weight of the rooted tree cover found by TREE COVER, N to be the weight of a weight-minimal rooted tree cover, O to be the cover time of MFC and P to be the minimal cover time. Because circumnavigating a tree of weight M requires a travel time of M, we get that $O \leq M$. The approximation guarantee of Theorem 3 provides that $M \leq 4(1+\epsilon)N$. Lemma 4 provides that $N/4 \leq P + w_{\max}/4$. Combining all these results yields $O \leq M \leq 4(1+\epsilon)N \leq 16(1+\epsilon)P + 4(1+\epsilon)w_{\max}$. ■

The ratio $\phi = w_{\max}/w_{\text{sum}}$ is close to zero for terrain with many large cells of about the same weight. For example, $\phi = 0.0814$ for the terrain from Figure 1. Then, $16(1+\epsilon) = 16(1+\phi K)$ is close to sixteen for a small number of robots K. Thus, the cover times of MFC are at most about sixteen times larger than minimal.

VII. Simulation Results

We evaluate MFC and MSTC experimentally for both team objectives, namely "Cover without Return" and "Cover with Return," in different scenarios, namely different kinds of terrain, different numbers of robots and different clustering of the initial large cells of the robots. The terrain always consists of 49×49 large cells. Their weight in weighted terrain is chosen uniformly at random from the weights 8, 16, 24, ..., 80. Figure 10 shows the three different kinds of terrain. The first kind of terrain is empty. The second kind is an outdoor-like terrain where walls are randomly removed from a random depth-first search maze until the wall density drops to 10 percent, resulting in terrain with random obstacles. The third kind is an indoor-like terrain with walls and doors. The positions of the walls and doors are fixed, but doors are closed with 20 percent probability. We vary the number of robots from 2, 8, 14 to 20. We ensure that no two robots are placed in the same large cell by randomly choosing different large cells for each robot and placing the robots in their upper right small cells. A clustering percentage parameter x determines how strongly the initial large cells of the robots are clustered. The initial large cell of the first robot is chosen uniformly at random from all unblocked large cells. The initial large cells of the other robots are then chosen uniformly at random from all unblocked large cells in an area centered at the first robot, whose height and width are (approximately) $x\%$ of the height and width of the terrain. Thus, a small value of x results in a large clustering of initial large cells, while $x = 200$ is equivalent to no clustering at all (called *none* clustering).

For each scenario, we average the cover times of MSTC and MFC over 50 runs with randomly generated terrain (if applicable), randomly chosen initial large cells, randomly generated spanning trees (for MSTC) and randomly generated rooted tree covers (for MFC).[2] All runs terminate within 30 second. A lower bound that represents idealized cover times for each scenario can be calculated by dividing the sum of the weights of all large cells by the number of robots and subtract one in unweighted terrain and $w_{\max}/4$ in weighted terrain. These ideal cover times would result if no robot needed to pass through already covered small cells. We are interested in the ratios of the actual cover times of MSTC or MFC and the ideal cover times since these ratios are upper bounds on how much the actual cover times are larger than minimal. They are only upper bounds since the ideal might not be achievable. For instance, several robots must visit the same small cells in the example from Figure 8. Figures 11-14 show the relationships of the number of robots and the ratios for each combination of multi-robot coverage algorithm, scenario and team objective.

We make the following observations: The ratios of MFC are smaller than those of MSTC for both team objectives. MFC takes the team objective already into account when finding a tree for each robot to circumnavigate, whereas MSTC takes the team objective into account when it decides how the robots should circumnavigate the single tree. Thus, the cover times of MFC are smaller than those of MSTC. The ratios increase with the number of robots for both MFC and MSTC since the robots then need to pass more and more through already covered cells. The ratios increase very slowly with the number of robots for MFC, but much faster for MSTC, which distributes most of the travel time among two robots only. The ratios change insignificantly with the amount of clustering for MFC (as can be clearly seen in Figures 11-14), but a lot for MSTC. In particular, the cover times of MFC are small in the common situation where robots are deployed together. Finally, the ratios change insignificantly for MFC if the team objective is changed from "Cover without Return" to "Cover with Return," but noticeably for MSTC. Thus, all robots are close to their initial small cells when coverage is complete for MFC, which facilitates their retrieval.

Theorem 2 guarantees that the cover times of MFC in unweighted terrain are at most a factor of about 16 larger than minimal. Similarly, Theorem 5 guarantees that the cover times of MFC in weighted terrain are at most a factor of about 16 larger than minimal since the values of ϕ are indeed very small. For example, $\phi = 8.9 \times 10^{-4}$ for empty terrain, $\phi = 9.9 \times 10^{-4}$ for outdoor-like terrain and $\phi = 10.5 \times 10^{-4}$ for indoor-like terrain. Empirically, the ratios are significantly smaller for all tested scenarios and both team objectives, namely at most 1.91.

Figures 15 and 16 present examples of MSTC and MFC, respectively, solving multi-robot coverage problems for four robots in indoor-like terrain. The areas covered by different robots are shown in different colors. The covered areas of MFC are much more balanced in size than those of MSTC, explaining both the smaller cover time with return and the smaller ratio of MFC.

[2]For MSTC, we first choose the root of the spanning tree randomly among the initial large cells of the robots and then use the Prim algorithm to generate the spanning tree, breaking ties randomly. For MFC, we use the tree cover algorithms described in this article, breaking ties randomly.

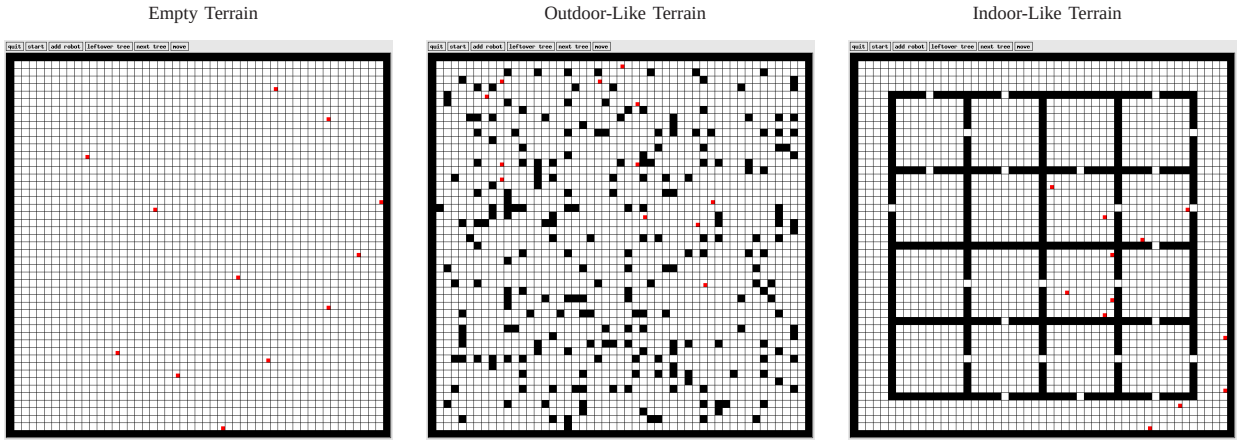


Fig. 10. Screenshots of Different Kinds of Terrain

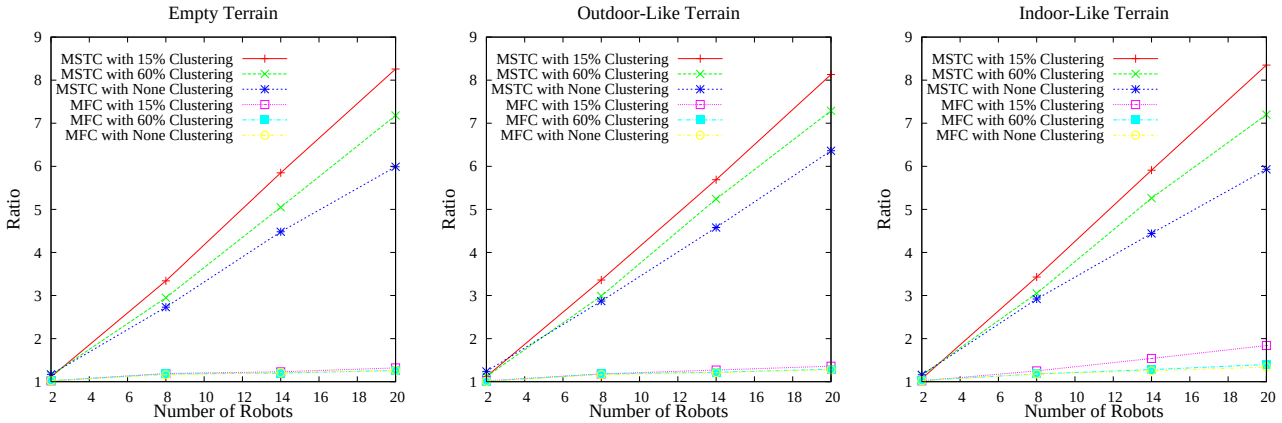


Fig. 11. Simulation Results for the Team Objective "Cover with Return" in Unweighted Terrain

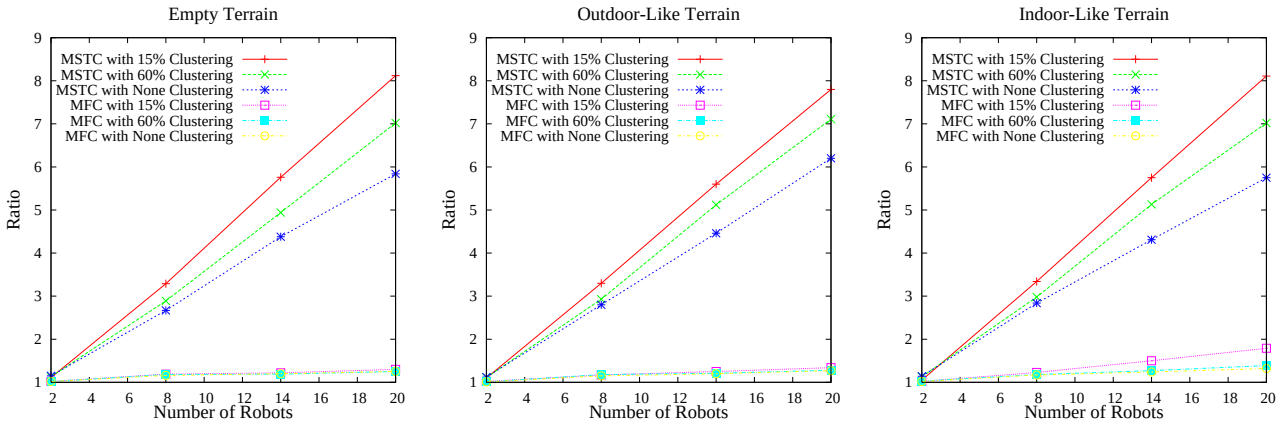


Fig. 12. Simulation Results for the Team Objective "Cover without Return" in Unweighted Terrain

VIII. Discussion and Future Work

Our main contribution is a theoretical one since we demonstrated how difficult multi-robot coverage is under idealized conditions. We showed that solving several versions of multi-robot coverage problems with minimal cover times is NP-hard. We then introduced a new multi-robot coverage algorithm, called Multi-Robot Forest Coverage (MFC), which is (to the best of our knowledge) the first polynomial-time constant-factor approximation algorithm for multi-robot coverage in both weighted and unweighted terrain. Our simulation results showed that the cover times of MFC are smaller than the ones of Multi-Robot Spanning-Tree Coverage (MSTC), an alternative multi-robot coverage algorithm, and close to minimal in all tested scenarios. Our simulation results used randomly generated spanning trees (for MSTC) and randomly generated rooted tree covers (for MFC), to compare MSTC and MFC on equal grounds. However, the cover times of MSTC can be reduced by carefully constructing its spanning trees [3] [4]. It is future work to determine whether the cover times of MFC can be reduced as well by carefully constructing its rooted tree covers and then compare MSTC and MFC using these

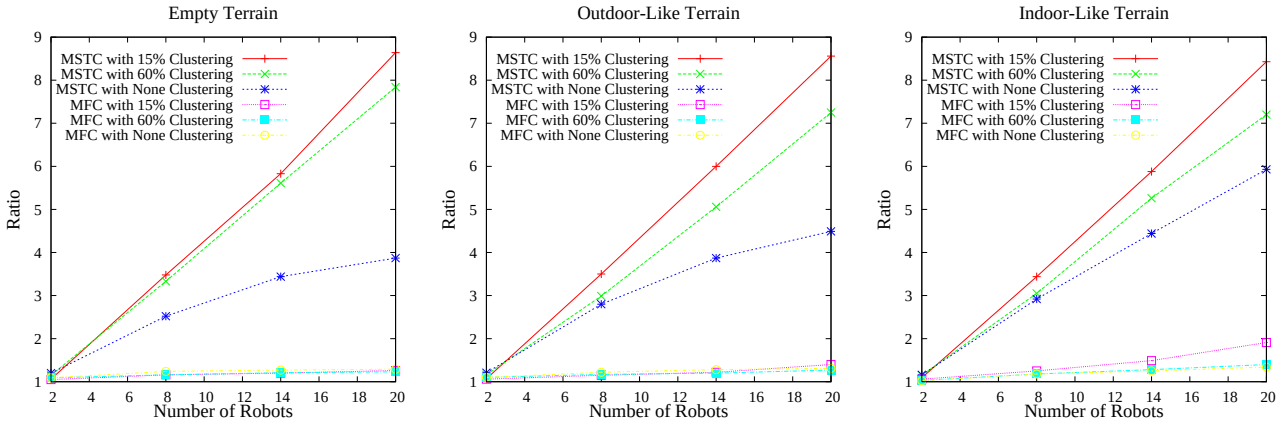


Fig. 13. Simulation Results for the Team Objective "Cover with Return" in Weighted Terrain

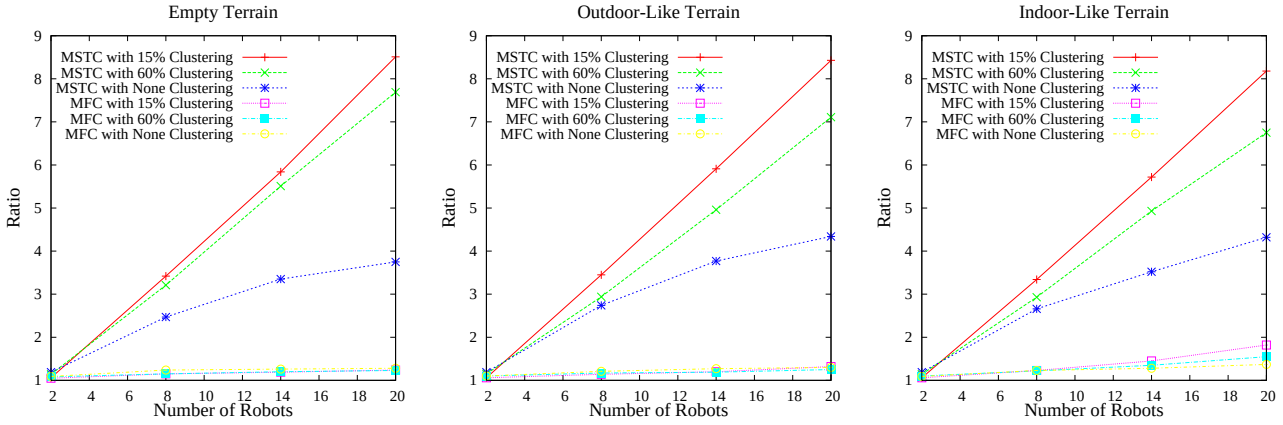


Fig. 14. Simulation Results for the Team Objective "Cover without Return" in Weighted Terrain

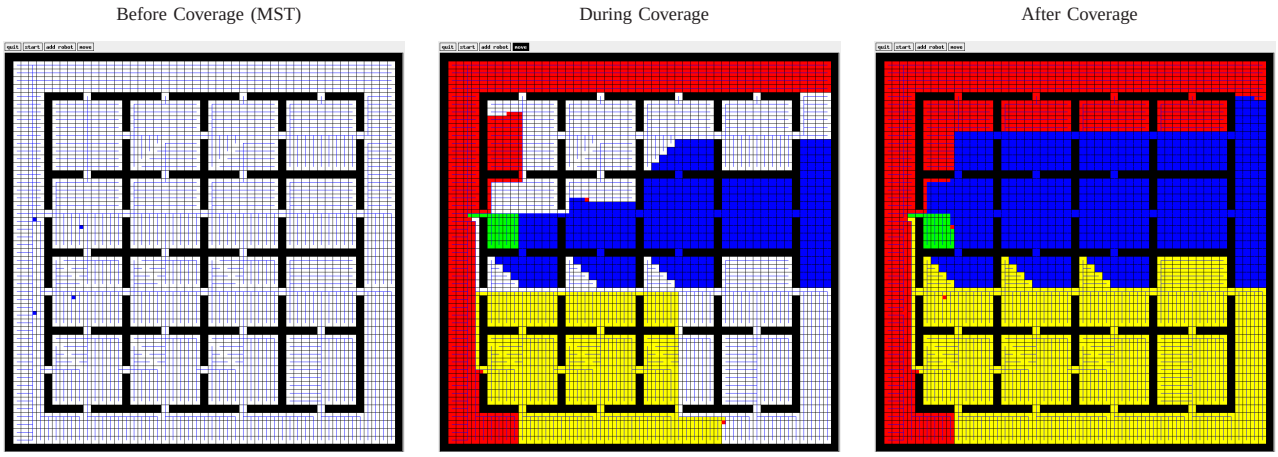


Fig. 15. Example of MSTC in Indoor-Like Terrain (Cover Time with Return = 3,338 and Ratio = 1.63)

spanning trees and rooted tree covers, respectively. Also, the current version of MFC assumes ideal robots. However, robots embedded in the real world are subject to sensor and actuator noise and may not always be reliable [5]. It is future work to generalize our algorithm to robots with such uncertainty and other typical imperfections, which also includes making it robust in the presence of failing robots, a property that MSTC already has. It is also promising to combine the ideas behind MSTC and MFC.

APPENDIX

We modify the tree-cover algorithm by Even et al. [11] (and the proofs in that paper) to work on graphs with weighted vertices rather than weighted edges. We present the resulting tree-cover algorithm (called TREE COVER), prove its properties and describe how MFC uses it.

A. Weight-Minimal Rooted Tree Cover Problem

We define the WEIGHT-MINIMAL ROOTED TREE COVER problem as follows: Let $G = (V, E)$ be a graph with weighted

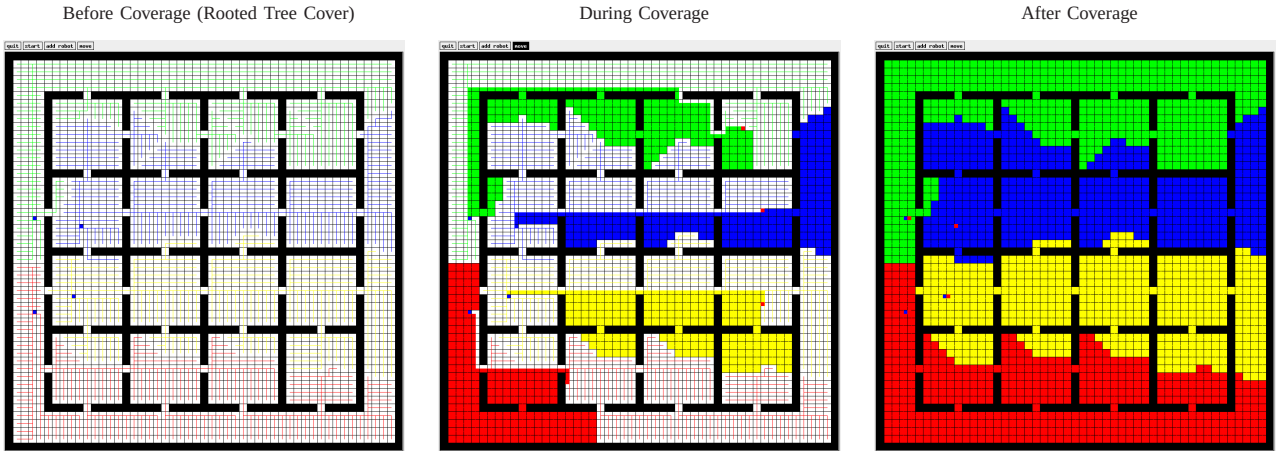


Fig. 16. Example of MFC in Indoor-Like Terrain (Cover Time with Return = 2,064 and Ratio = 1.01)

vertices, where $w(v)$ is the positive integer weight of vertex $v \in V$. Define $K \subseteq V$ to be a set of distinguished vertices, called roots. A K-rooted tree cover of G is a forest of $|K|$ trees, which can share vertices and edges. The set of their roots must be equal to K, and every vertex in V has to be in at least one tree. The weight of a tree is the sum of the weights of its vertices. The weight of a K-rooted tree cover is the largest weight of any of its trees. Given a graph $G = (V, E)$ with weighted vertices and a set $K \subseteq V$ of roots, find a weight-minimal K-rooted tree cover of graph G.

B. Definitions

We define the weight of a path in the graph to be the sum of the weights of the vertices in the path, except for its end vertices. We define the distance of a vertex and a tree in the graph to be the minimal weight of any path that connects the vertex to some vertex in the tree. We define the distance of two trees in the graph to be the minimal weight of any path that connects some vertex in one of the trees to some vertex in the other tree. We also use the notation from Section III.

C. Complexity of Finding Weight-Minimal Rooted Tree Covers

We show that finding weight-minimal rooted tree covers is NP-hard, which provides our motivation for designing the polynomial-time constant-factor approximation algorithm TREE COVER.

Theorem 6: It is NP-hard to determine whether there exist K-rooted tree covers whose weights are smaller than given values for specified graphs and sets of roots.

Proof: To prove the NP-hardness of solving the WEIGHT-MINIMAL ROOTED TREE COVER problem, we reduce from the BIN-PACKING problem: Given a set of elements with given positive integer sizes and a fixed number of bins, each with the same given integer capacity, can each element be placed in exactly one of the bins so that the sum of the sizes of the elements in each bin does not exceed its capacity?

For a given instance of the BIN-PACKING problem, we construct a completely connected graph G with one vertex for each element (whose weight is equal to the size of the element) and one vertex for each bin (whose weight is one). The set of roots K contains exactly the vertices for the bins. This completes the construction, which can be done in polynomial time. We claim that the weight of a K-rooted tree cover of graph G is at most the given capacity plus one iff the bin-packing problem is solvable.

"If" direction: Assume that each element can be placed in exactly one of the bins so that the sum of the sizes of the elements in each bin does not exceed its capacity. We make the tree rooted in the vertex of a bin contain the vertices of the elements that the bin contains. Then, the weight of the resulting K-rooted tree cover is at most the given capacity plus one, which meets the requirement.

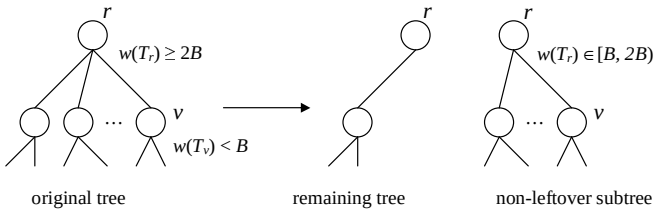
"Only if" direction: Assume that the weight of a K-rooted tree cover of graph G is at most the given capacity plus one. We place each element in one of the bins whose vertex is the root of a tree that contains the vertex of the element. Then, the bin-packing solution meets the capacity constraints. ■

D. Tree Cover Algorithm

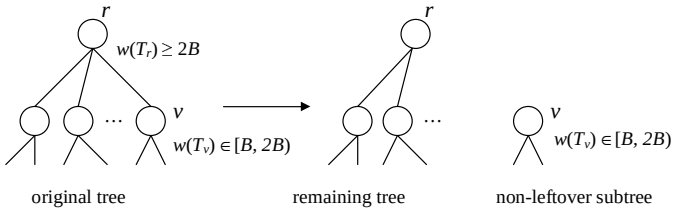
We now describe TREE COVER, that takes as input a graph $G = (V, E)$, a set of roots K and a bound $B \geq w_{\max}$. Given any bound $B \in [w_{\max}, w_{\text{sum}}]$, TREE COVER either reports SUCCESS and returns a K-rooted tree cover of graph G with weight at most $4B$ or reports FAILURE, in which case there does not exist a K-rooted tree cover of graph G with weight at most $B/(1+\epsilon)$. In later sections, we prove these properties and show how to find a small value of B such that TREE COVER reports SUCCESS and the weight of the resulting K-rooted tree cover is at most a factor of $4(1+\epsilon)$ larger than minimal.

TREE COVER operates as follows:

1) Contract all roots into a new vertex k^* by adding vertex k^* to the graph, removing all roots and the edges connecting them from the graph and, for each edge $(v, k) \in E$ with $v \notin K$ and $k \in K$, adding edge (v, k^*) to the graph and removing the edge (v, k) from the graph.
2) Find any spanning tree of the resulting graph.



(a) *Case 3a*



(b) *Case 3b*

Fig. 17. Cases 3a and 3b of the Tree Decomposition

3) Uncontract the single vertex k^* back to the roots by adding all roots to the spanning tree, removing vertex k^* from the spanning tree, and, for each edge (v, k^*) with $v \in V$ in the spanning tree, adding one edge $(v, k) \in E$ with $k \in K$ to the spanning tree and removing the edge (v, k^*) from the spanning tree. This splits the spanning tree into a forest of $|K|$ trees.
4) Decompose each tree recursively into zero or more non-leftover subtrees whose weights are in the interval $[B, 2B)$ and one leftover subtree whose weight is in the interval $(0, B)$. Assume that a tree is rooted in r. The decomposition procedure repeatedly removes vertices from the tree as it generates the non-leftover subtrees. The leftover subtree consists of all vertices that have not been removed after the decomposition procedure terminates. If all vertices have been removed, then the leftover subtree consists of only r. We distinguish three cases:
   - *Case 1:* The weight of the tree rooted in r is less than B. Then, the decomposition procedure simply returns.
   - *Case 2:* The weight of the tree rooted in r is in the interval $[B, 2B)$. Then, one non-leftover subtree consists of the tree rooted in r. The decomposition procedure removes all vertices of this non-leftover subtree from the tree rooted in r (leaving the empty tree) and returns.
   - *Case 3:* The weight of the tree rooted in r is at least $2B$. We distinguish three subcases:
     – *Case 3a:* The weights of all trees rooted in children of r are less than B. Then, the decomposition procedure picks a number of trees rooted in children of r so that the weight of the tree consisting of r and these trees is in the interval $[B, 2B)$. (This is possible since $w(r) \leq w_{\max} \leq B$ and the weights of all trees rooted in children of r are less than B but the weight of the tree rooted in r is at least $2B$.) One non-leftover subtree consists of r and these trees. The decomposition procedure removes all vertices of this non-leftover subtree except for r from the tree rooted in r and calls itself recursively on the remaining tree rooted in r in order to find the other non-leftover subtrees. This case is illustrated in Figure 17a, where T_v denotes the tree rooted in v and $w(T_v)$ denotes its weight.
     – *Case 3b:* The weight of at least one tree rooted in a child of r is in the interval $[B, 2B)$. Then, the decomposition procedure picks such a tree. One non-leftover subtree consists of this tree. The decomposition procedure removes all vertices of the non-leftover subtree from the tree rooted in r and calls itself recursively on the remaining tree rooted in r in order to find the other non-leftover subtrees. This case is illustrated in Figure 17b.
     – *Case 3c:* Otherwise, the weight of at least one tree rooted in a child of r is at least $2B$. Then, the decomposition procedure calls itself recursively on such a tree rooted in a child of r and then on the remaining tree rooted in r in order to find the non-leftover subtrees.
5) Find a maximum matching [21] of all non-leftover subtrees to the roots, subject to the constraint that a non-leftover subtree can only be matched to a root if the non-leftover subtree and the leftover tree of the root are at distance at most B. The maximum matching problem can be modeled as a maximal flow problem and be solved in polynomial time, for example, with the Ford-Fulkerson algorithm [12].
6) If one or more non-leftover subtrees cannot be matched, report FAILURE. Otherwise, report SUCCESS and, for each root, return the tree consisting of the leftover subtree of the root, the single non-leftover subtree (if any) matched to the root and a weight-minimal path (if any) from the non-leftover subtree to the leftover subtree.

E. Properties

Clearly, TREE COVER runs in polynomial time and either reports SUCCESS or FAILURE. It is also easy to see that the weights of all non-leftover subtrees (if any) returned by the decomposition procedure in Step 4 of TREE COVER for a given tree are in the interval $[B, 2B)$. The weight of the leftover subtree is in the interval $(0, B)$. Also, the root of the tree is in the leftover subtree. We now prove two main properties of TREE COVER.

Theorem 7: If TREE COVER reports SUCCESS, then it returns a K-rooted tree cover of graph G with weight at most $4B$.

Proof: If TREE COVER reports SUCCESS, then it returns, for each root, the tree consisting of the leftover subtree of the root (whose weight is at most B), the single non-leftover

subtree (if any) matched to the root (whose weight is at most $2B$) and a weight-minimal path (if any) from the non-leftover subtree to the leftover subtree (whose weight is at most B). The weight of each tree is thus at most $4B$. ■

Theorem 8: If TREE COVER reports FAILURE, then there does not exist a K-rooted tree cover of graph G with weight at most $B/(1+\epsilon)$, where $\epsilon = \phi|K|$, $\phi = w_{\max}/w_{\text{sum}}$ and $|K|$ is the number of roots.

Proof: Define B' to be the weight of a weight-minimal K-rooted tree cover T of graph G and assume that $B' \leq B/(1+\epsilon)$. Define L to be the set of non-leftover subtrees created in Step 4 of TREE COVER and $K(l) \subseteq K$ to be the set of roots at a distance of at most B from $l \in L$. We show that $|\bigcup_{l\in L'} K(l)| \geq |L'|$ for every $L' \subseteq L$. Step 5 of TREE COVER can then match all non-leftover subtrees according to Hall's Marriage Theorem [17]. Therefore, TREE COVER reports SUCCESS, which is a contradiction.

We now show that $|\bigcup_{l\in L'} K(l)| \geq |L'|$. Consider any $L' \subseteq L$. Define $T' \subseteq T$ to be the set of trees in a weight-minimal K-rooted tree cover of graph G which each have at least one vertex in common with at least one of the trees in L'. Define $w(L') = \sum_{l\in L'} w(l)$ and $w(T') = \sum_{t\in T'} w(t)$.

First, for every $t \in T'$, there exists an l in L' that has at least one vertex in common with t, per definition of T'. The root of t is thus at a distance of at most $w(t) \leq B' \leq B/(1+\epsilon) \leq B$ from l, which implies that the root of t is contained in $K(l)$. $\bigcup_{l\in L'} K(l)$ thus contains the roots of all trees in T', which implies that $|\bigcup_{l\in L'} K(l)| \geq |T'|$.

Second, $w(L') \geq B|L'|$ and $w(T') \leq B'|T'|$ since $w(l) \in [B, 2B)$ for all $l \in L'$ and $w(t) \leq B'$ for all $t \in T'$.

Third, every vertex in a tree in L' is also in at least one tree in T' since T is a tree cover. The trees in L' can contain at most $|L'|$ duplicate vertices because every non-leftover subtree created in Step 4 of TREE COVER contains at most one vertex that has not yet been removed from all trees created in Step 3 of TREE COVER and thus could be a duplicate vertex. (The trees created in Step 3 of TREE COVER share at most their roots, and Step 4 of TREE COVER removes all vertices of a non-leftover subtree from its tree when it creates the non-leftover subtree, except possibly for the root of the non-leftover subtree in Case 3a.) Each duplicate vertex has a weight of at most $w_{\max}$, which implies that $w(L') \leq w(T') + w_{\max}|L'|$.

Finally, the sum of the weights of all vertices can be split evenly among the trees at best. Thus, $B' \geq w_{\text{sum}}/|K|$, which implies that $w_{\max} = w_{\text{sum}}\phi \leq |K|B'\phi$.

Combining all these observations yields

$$\begin{aligned} B'|T'| &\geq w(T') \\ &\geq w(L') - w_{\max}|L'| \\ &\geq B|L'| - |K|B'\phi|L'| \\ &= (B - |K|B'\phi)|L'| \\ &\geq (B'(1+\epsilon) - B'\epsilon)|L'| \\ &= B'|L'|, \end{aligned}$$

and thus $|\bigcup_{l\in L'} K(l)| \geq |T'| \geq |L'|$. ■

F. Application

Theorem 9: TREE COVER can be used to find a K-rooted tree cover of graph G whose weight is at most a factor of $4(1+\epsilon)$ larger than minimal, where $\epsilon = \phi|K|$, $\phi = w_{\max}/w_{\text{sum}}$ and $|K|$ is the number of roots.

Proof: Define B' to be the weight of a weight-minimal K-rooted tree cover of G. B' is an integer since the weights of all vertices are integers, and $B' \in [w_{\max}, w_{\text{sum}}]$ since at least one tree contains a vertex with weight $w_{\max}$ and every tree contains at most all vertices. We perform binary search on the interval $[w_{\max}, w_{\text{sum}}]$ to find a small value of B for which TREE COVER reports SUCCESS. We start with the lower bound $w_{\max}$ and the upper bound w_{sum}. We then repeatedly run TREE COVER with B set to the mean of the lower and upper bound. If TREE COVER reports FAILURE, then we set the lower bound to B. Otherwise, we set the upper bound to B. We stop once the difference of the upper and lower bound is at most $1+\epsilon$. Define B_l to be the lower bound and B_u to be the upper bound of the binary search after termination. Define $\hat{B} = \lceil B_l/(1+\epsilon)\rceil$. We distinguish two cases:

- *Case 1:* If TREE COVER reports SUCCESS with B set to $(1+\epsilon)\hat{B}$, then we output the tree cover returned by TREE COVER for this value of B.
  If $B_l = w_{\max}$ (the initial lower bound), then $B' > B_l/(1+\epsilon)$ since $B' \geq w_{\max} > w_{\max}/(1+\epsilon)$. Otherwise, $B' > B_l/(1+\epsilon)$ according to Theorem 8 since TREE COVER reports FAILURE with B set to B_l according to the binary search used. In both cases, $B' > B_l/(1+\epsilon)$. But then, $B' \geq \lceil B_l/(1+\epsilon)\rceil = \hat{B}$ since B' is an integer. TREE COVER reports SUCCESS with B set to $(1+\epsilon)\hat{B}$ according to the assumption. It returns a tree cover whose weight is at most $4(1+\epsilon)\hat{B}$ according to Theorem 7, which is at most a factor of $4(1+\epsilon)$ larger than minimal since $4(1+\epsilon)B' \geq 4(1+\epsilon)\hat{B}$.
- *Case 2:* If TREE COVER reports FAILURE with B set to $(1+\epsilon)\hat{B}$, then we output the tree cover returned by TREE COVER with B set to B_u.
  $B' > (1+\epsilon)\hat{B}/(1+\epsilon) = \hat{B}$ according to Theorem 8 since TREE COVER reports FAILURE with B set to $(1+\epsilon)\hat{B}$. Then $B' \geq \hat{B}+1$ since B' and $\hat{B}$ are integers. $\hat{B}+1 \geq B_l/(1+\epsilon)+1 \geq B_u/(1+\epsilon)$ since $(B_u - B_l)/(1+\epsilon) \leq 1$ according to the termination condition of the binary search. Thus, $B_u \leq (1+\epsilon)(\hat{B}+1) \leq (1+\epsilon)B'$.
  If $B_u = w_{\text{sum}}$ (the initial upper bound), then TREE COVER reports SUCCESS with B set to B_u, as explained above already. Otherwise, TREE COVER reports SUCCESS with B set to B_u according to the binary search used. It returns a tree cover of weight at most $4B_u \leq 4(1+\epsilon)B'$ according to Theorem 7, which is at most a factor of $4(1+\epsilon)$ larger than minimal. ■

The binary search runs in polynomial time because TREE COVER runs in polynomial time and is run at most $\lceil \log_2((w_{\text{sum}} - w_{\max})/(1+\epsilon))\rceil + 2 \leq \log_2 w_{\text{sum}} + 3$ times, which is polynomial in the size of the input.

REFERENCES

[1] E. Acar, H. Choset, A. Rizzi, P. Atkar, and D. Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4):331–344, 2002.

[2] A. Agarwal, L. Hiot, E. Joo, and N. Nghia. Rectilinear workspace partitioning for parallel coverage using multiple unmanned aerial vehicles. *Advanced Robotics*, 21(1-2), 2007.
[3] N. Agmon, N. Hazon, and G. Kaminka. Constructing spanning trees for efficient multi-robot coverage. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1698–1703, 2006.
[4] N. Agmon, N. Hazon, and G. Kaminka. The giving tree: Constructing trees for efficient offline and online nulti-robot coverage. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):143–168, 2008.
[5] P. Amstutz, N. Correll, and A. Martinoli. Distributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):307–333, 2008.
[6] Z. Butler. *Distributed Coverage of Rectilinear Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh (Pennsylvania), 2000.
[7] H. Choset. Coverage for robotics – a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.
[8] H. Choset and P. Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Proceedings of the International Conference on Field and Service Robotics*, 1997.
[9] X. Deng and C. Papadimitriou. Competitive distributed decision-making. *Algorithmica*, 16(2):350–356, 1992.
[10] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.
[11] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Min-max tree covers of graphs. *Operations Research Letters*, 32:309–315, 2004.
[12] L. Ford and D. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
[13] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence*, 31:77–98, 2001.
[14] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.
[15] A. Gasparri, B. Krishnamachari, and G. S. Sukhatme. A framework for multi-robot node coverage in sensor networks. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):281–305, 2008.
[16] S. Ge and C. Fua. Complete multi-robot coverage of unknown environments with minimum repeated coverage. In *Proceedings of the International Conference on Robotics and Automation*, pages 715–720, 2005.
[17] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 10:26–30, 1935.
[18] N. Hazon and G. Kaminka. Redundancy, efficiency, and robustness in multi-robot coverage. In *Proceedings of the International Conference on Robotics and Automation*, pages 735–741, 2005.
[19] S. Ichikawa and F. Hara. Characteristics of object-searching and object-fetching behaviors of multi-robot system using local communication. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pages 775–781, 1999.
[20] R. Jarvis and J. Byrne. Robot navigation: Touching, seeing and knowing. In *Proceedings of the Australian Conference on Artificial Intelligence*, pages 18–20, 1986.
[21] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison Wesley, 2005.
[22] S. Koenig and Y. Liu. Terrain coverage with ant robots: A simulation study. In *Proceedings of the International Conference on Autonomous Agents*, pages 600–607, 2001.
[23] C. Kong, N. Peng, and I. Rekleitis. Distributed coverage with multi-robot system. In *Proceedings of the International Conference on Robotics and Automation*, pages 2423–2429, 2006.
[24] D. Kurabayashi, J. Ota, T. Araiand, and E. Yoshida. Cooperative sweeping by multiple mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 1744 –1749, 1996.
[25] D. Kurabayashi, J. Ota, and E. Yoshida. An algorithm of dividing a work area to multiple mobile robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 286–291, 1995.
[26] D. Latimer, S. Srinivasa, V. Lee-Shue, S. Sonne, H. Choset, and A. Hurst. Towards sensor based coverage with robot teams. In *Proceedings of the International Conference on Robotics and Automation*, pages 961–967, 2002.
[27] J. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
[28] C. Luo and S. Yang. A real-time cooperative sweeping strategy for multiple cleaning robots. In *Proceedings of the International Symposium on Intelligent Control*, pages 660–665, 2002.
[29] R. Menezes, F. Martins, F. Vieira, R. Silva, and M. Braga. A model for terrain coverage inspired by ant's alarm pheromones. In *Proceedings of the ACM Symposium on Applied Computing*, pages 728–732, 2007.
[30] T. Min and H. Yin. A decentralized approach for cooperative sweeping by multiple mobile robots. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 380–385, 1998.
[31] H. Noborio, T. Yoshioka, and T. Hamaguchi. On-line deadlock-free path-planning algorithms by means of a sensor-feedback tracing. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, pages 1291–1296, 1995.
[32] S. Qutub, R. Alami, and F. Ingrand. How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1610–1615, 1997.
[33] I. Rekleitis, G. Dudeck, and E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1340–1345, 1997.
[34] I. Rekleitis, V. Lee-Shue, A. New, and H. Choset. Limited communication, multi-robot team based coverage. In *Proceedings of the International Conference on Robotics and Automation*, pages 3462–3468, 2004.
[35] N. Rowe and R. Alexander. Finding optimal-path maps for path planning across weighted regions. *The International Journal of Robotics Research*, 19(2):83–95, 2000.
[36] D. Spears, W. Kerr, and W. Spears. Physics-based robot swarms for coverage problems. *International Journal on Intelligent Control and Systems*, 11(3):124–140, 2006.
[37] J. Svennebring and S. Koenig. Building terrain-covering ant robots. *Autonomous Robots*, 16(3):313–332, 2004.
[38] J. VanderHeide and N. Rao. Terrain coverage of an unknown room by an autonomous mobile robot. Technical Report ORNL/TM-13117, Oak Ridge National Laboratory, Oak Ridge (Tennessee), 1995.
[39] J. Vörös. Mobile robot path planning among weighted regions using quadtree representations. In *Proceedings of the International Conference on Computer Aided Systems Theory*, pages 239–249, 2000.
[40] I. Wagner, Y. Altshuler, V. Yanovski, and A. Bruckstein. Cooperative cleaners: A study in ant robotics. *The International Journal of Robotics Research*, 27(1):127–151, 2008.
[41] I. Wagner, M. Lindenbaum, and A. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.
[42] G. Winward and N. Flann. Coordination of multiple vehicles for area coverage tasks. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 1351–1356, 2007.
[43] A. Zelinsky, R. Jarvis, J. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of the International Conference on Advanced Robotics*, pages 533–538, 1993.
[44] X. Zheng, S. Jain, S. Koenig, and D. Kempe. Multi-robot forest coverage. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2318–2323, 2005.
[45] X. Zheng and S. Koenig. Robot coverage of terrain with non-uniform traversability. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 2300–2309, 2007.

ACKNOWLEDGMENTS

Parts of this article have been presented at the International Conference on Intelligent Robots and Systems in 2005 [44] and 2007 [45]. This article corrects one mistake in [44], where the proof of Theorem 3 stated that $2\hat{C} \leq \hat{T}_{ul}$ but should have stated that $\hat{C} \leq \hat{T}_{ul}$. We thank Gal Kaminka for interesting discussions on Multi-Robot Spanning Tree Coverage and multi-robot coverage in general. Our research was supported by, or in part by, the U.S. Army Research Laboratory and the U.S. Army Research Office under contract W911NF-08-1-0468 and the National Science Foundation under contracts IIS-0350584 and IIS-0413196 to Sven Koenig. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring organizations and agencies.

Xiaoming Zheng is a Ph.D. student in computer science at the University of Southern California, where he has worked on multi-robot coverage algorithms since 2004. His research interests are multi-robot coordination and task-allocation algorithms. He received his B.S. degree in computer science from the University of Science and Technology of China in 2004 and is a member of IEEE and AAAI.

Sven Koenig is an associate professor of computer science at the University of Southern California. His research centers around techniques for decision making that enable single robots and teams of robots to act intelligently in their environments. He is the recipient of an ACM Recognition of Service Award, an NSF CAREER award, an IBM Faculty Partnership Award, a Charles Lee Powell Foundation Award, a Raytheon Faculty Fellowship Award, a Mellon Mentoring Award, a Fulbright Fellowship and the Tong Leong Lim Pre-Doctoral Prize from the University of California at Berkeley. He co-founded Robotics: Science and Systems, a general robotics conference.

David Kempe is an associate professor of computer science at the University of Southern California. His research is in the areas of computer science theory and the design and analysis of algorithms, with an emphasis on social networks and algorithmic game theory as well as distributed network algorithms. He is the recipient of an NSF CAREER award, the 2007 Junior Research Award of the USC Viterbi School of Engineering, a Mellon Mentoring Award, an ONR Young Investigator Award and a Sloan Fellowship.

Sonal Jain is a software design engineer at Microsoft Corporation in Redmond, Washington. His research interests are resource-allocation strategies for workflows in computational grids and multi-robot task-allocation algorithms. He received his M.S. degree in computer science from the University of Southern California and his B.S. degree in electrical engineering from Banaras Hindu University in India.