Robot Coverage of Terrain with Non-Uniform Traversability

Xiaoming Zheng Sven Koenig
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781, USA
{*xiaominz, skoenig*}*@usc.edu*

***Abstract*— In this paper, we study how multiple robots can cover known terrain quickly. We extend Multi-Robot Forest Coverage, a state-of-the-art multi-robot coverage algorithm, from terrain with uniform traversability to terrain with non-uniform traversability, which is nontrivial. We prove that its cover times are at most about sixteen times larger than minimal and demonstrate experimentally that they are significantly smaller than those of an alternative multi-robot coverage algorithm.**

***Index Terms*— Cell Decomposition, Multi-Robot Coverage, Robot Teams, Spanning Tree Coverage, Terrain Coverage.**

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. In this paper, we study coverage with multiple robots since multiple robots can often cover terrain faster than a single robot. Recently, several researchers have proposed multi-robot coverage algorithms for terrain with uniform traversability, where the traversal time is the same everywhere. Two promising multi-robot coverage algorithms are Multi-Robot Spanning Tree Coverage (MSTC) [4] and Multi-Robot Forest Coverage (MFC) [5], which both extend the single-robot coverage algorithm Spanning Tree Coverage (STC) [3]. In this paper, we generalize these multi-robot coverage algorithms to terrain with non-uniform traversability (= weighted terrain), as shown in Figure 1, to extend their applicability to more realistic situations [1].

We show that STC finds solutions with minimal cover times in polynomial time for a single robot in weighted terrain if the robot has to return to its initial location after it has covered the terrain. Multi-robot coverage with minimal cover times is known to be NP-hard for two robots and conjectured to be NP-hard for an arbitrary number of robots [5]. Thus, one needs to design multi-robot coverage algorithms that determine solutions with suboptimal (but good) cover times in polynomial time. To this end, we generalize MSTC and MFC to weighted terrain. MSTC can be generalized relatively easily but cannot guarantee to find solutions with good cover times. MFC is nontrivial to generalize because it uses a tree cover algorithm [2] as a subroutine that is specific to non-weighted terrain. We thus first generalize the tree cover algorithm and only then MFC. We prove that the new version of MFC is guaranteed to find solutions with cover times that are at most about sixteen times larger than minimal. We then demonstrate experimentally that its cover times are significantly smaller than both those guaranteed by the worst-case bound and those of MSTC. We also demonstrate experimentally that the robots are close to their initial locations after they have covered the terrain, which facilitates their retrieval. Therefore, our generalization of MFC to weighted terrain indeed results in a powerful multi-robot coverage algorithm.

This work is partially supported by NSF Grants 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 National Science Foundation.

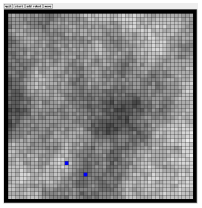

Fig. 1. Example of Weighted Terrain

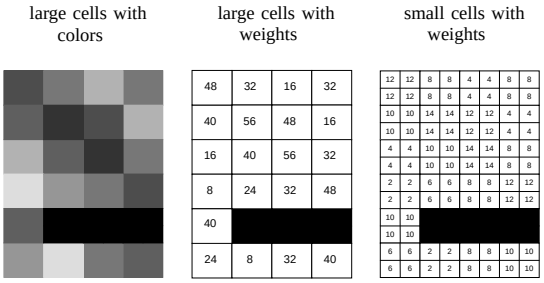

Fig. 2. Model of Weighted Terrain

II. Problem Description

We model weighted terrain as consisting of large square cells. Each large cell is either entirely blocked or entirely unblocked. Each unblocked large cell has a positive integer weight that corresponds to how difficult it is to traverse the large cell and 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 2. Each robot has the same size as the small cells. The robots start in different large cells and can move from any small cell to any adjacent small cell in the four main compass directions

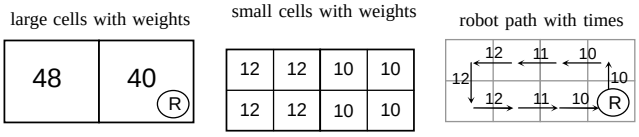

Fig. 3. Simple Single-Robot Coverage Problem

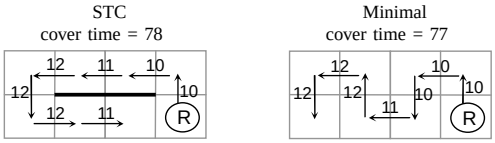

Fig. 4. Suboptimal Cover Time of STC

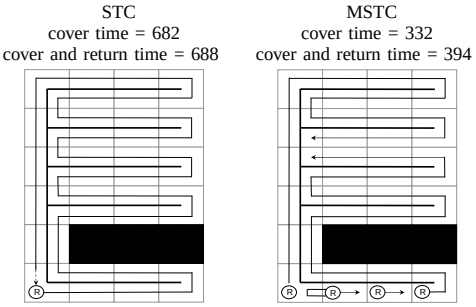

Fig. 5. Example of STC

Fig. 6. Example of MSTC

with a time that is equal to the average of the weight of the two small cells. Each move is atomic, that is, needs to be executed in full by a robot. The travel time along a robot path is the sum of the times of the moves of the robot when it moves along the path.

We study two different team objectives. For the team objective "Cover," the robots need to move so that each small cell is visited by at least one robot. Their cover time is equal to the largest travel time along any robot path. For the team objective "Cover and Return," the robots need to move so that each small cell is visited by at least one robot and then return to their initial small cells. Their cover and return time is again equal to the largest travel time along any robot path.

Figure 3 shows a complete coverage problem for a single robot, including the large cells with their weights, the small cells with their weights, and the robot path with the times of the moves for the team objective "Cover and Return." The cover and return time is equal to the sum of the weights of all large cells, namely 88. (We actually mean the sum of the weights of all unblocked large cells since blocked large cells do not have a weight but sacrifice precision but conciseness.)

III. Spanning Tree Coverage

Spanning Tree Coverage (STC) [3] finds solutions with minimal cover times (and cover and return times) in polynomial time for single robots in non-weighted terrain. STC can be generalized easily to weighted terrain, as follows: First, STC constructs a graph whose vertices correspond to the unblocked large cells and whose edges connect adjacent unblocked large cells. Second, STC finds a spanning tree of this graph. Third, STC lets the robot move along the path that circumnavigates this spanning tree. For the team objective "Cover and Return," the robot completely circumnavigates the spanning tree until it returns to its initial small cell. For the team objective "Cover," the robot stops once all small cells have been visited, that is, one move earlier. Clearly, STC runs in polynomial time.

Theorem 1: STC finds solutions with minimal cover and return times for a single robot in weighted terrain. The minimal cover and return times are equal to the sums of the weights of all large cells.

Proof: The robot needs to enter and exit every small cell at least once for the team objective "Cover and Return." Assume that the robot path is $(s_1, \ldots, s_n)$, where move s_i connects the two adjacent small cells c_i and c_{i+1}. ($c_{n+1} = c_1$ is the initial small cell of the robot.) Let the weight of small cell c_i be $w(c_i)$ and the time of move s_i be $t(s_i) = (w(c_i) + w(c_{i+1}))/2$. Then, the travel time along the robot path is $\sum_{i=1}^{n} t(s_i) = \sum_{i=1}^{n} (w(c_i)+w(c_{i+1}))/2 = \sum_{i=1}^{n} w(c_i)$, which is at least the sum of the weights of all small cells. STC makes the robot enter and exit every small cell exactly once. Its cover and return times are thus equal to the sums of the weights of all small cells and thus minimal. The sums of the weights of all small cells are equal to the sums of the weights of all large cells. ■.

The cover times of STC can be smaller than the sum of the weights of all large cells by at most the largest weight of any small cell because the robot stops one move before returning to its initial cell. STC does not necessarily find solutions with minimal cover times, as shown in Figure 4 for the single-robot coverage problem from Figure 3, but it finds solutions with close-to-minimal cover times. The robot needs to enter every small cell except for its initial small cell at least once and needs to exit every small cell except for its final small cell at least once. STC finds solutions where its final small cell is next to its initial small cell but the best final small cell might have a larger weight. Thus, the cover times of STC can be larger than minimal by at most the largest weight of any small cell (that is, a quarter of the largest weight of any large cell). Overall, STC finds solutions with close-to-minimal cover times and minimal cover and return times in polynomial time for a single robot in weighted terrain.

Figure 5 shows the spanning tree and robot path for the terrain from Figure 2 for one robot with the team objective "Cover." The cover time is 682 for STC. The robot has to make one additional move to return to its initial small cell for the team objective "Cover and Return" (shown with a dashed line in the figure). The cover and return time is 688 for STC.

IV. Multi-Robot Spanning Tree Coverage

Coverage with multiple robots can be faster than coverage with a single robot. (The backtracking version of) Multi-Robot Spanning Tree Coverage (MSTC) [4] finds solutions with suboptimal cover times (and cover and return times) in polynomial time for multiple robots in non-weighted terrain. MSTC can be generalized relatively easily to weighted terrain, as follows. We assume for simplicity here that there

are at least three robots and handle fewer than three robots in the extended version of the paper: First, MSTC constructs a graph whose vertices correspond to the unblocked large cells and whose edges connect adjacent unblocked large cells. Second, MSTC finds a spanning tree of this graph. Third, MSTC splits the path that circumnavigates this spanning tree into segments between the initial small cells of the robots. The number of segments is equal to the number of robots. The travel time along a segment is the sum of the times of the moves of a robot when it moves along the segment. Case 1: If the travel time along each segment is at most half of the travel time along the path, then MSTC lets each robot move counterclockwise along the segment adjacent to it. Otherwise, let $t(r, r')$ be the travel time along the segment from the initial small cell of robot r in the counterclockwise direction to the initial small cell of robot r'. Assume without loss of generality that robot r_i (r_j and r_k, respectively) is adjacent to robot r_h (r_i and r_j, respectively) in the counterclockwise direction and that $t(r_i, r_j)$ is larger than half of the travel time along the path. (Robots r_h and r_k are identical if there are only three robots.) Case 2: If $t(r_j, r_k) \leq t(r_h, r_i)$, then MSTC lets robot r_j first move counterclockwise until it is in an adjacent small cell to robot r_k (= meets robot r_k) and then move clockwise, lets robot r_k first move clockwise until it meets robot r_j and then move counterclockwise, and lets all other robots move counterclockwise. Case 3: If $t(r_j, r_k) > t(r_h, r_i)$, then MSTC lets robot r_h first move counterclockwise until it meets robot r_i and then move clockwise, lets robot r_i first move clockwise until it meets robot r_h and then move counterclockwise, and lets all other robots move clockwise. For the team objective "Cover," the robots move as given above and stop once all small cells have been visited. For the team objective "Cover and Return," the robots move as given above and, once all small cells have been visited, return to their initial small cells by moving either backward along their segments (MSTC) or along paths with minimal times from their current small cells to their initial small cells (optimized MSTC). Clearly, MSTC runs in polynomial time.

Theorem 2: The cover times of MSTC for at least three robots are at least about a factor of $2/(1+\phi)$ smaller than the cover times of STC, where ϕ is the ratio of the largest weight of any large cell and the sum of the weights of all large cells.

Proof: Let w_{max} be the largest weight of any large cell and w_{sum} be the sum of the weights of all large cells (which equals the time of the path that circumnavigates the spanning tree). Then $\phi = w_{max}/w_{sum}$. Case 1: If the travel time along each segment is at most half of the travel time along the path that circumnavigates the spanning tree, then MSTC lets each robot move along the segment adjacent to it in the counterclockwise direction. The travel time of each robot and the cover time of MSTC thus is at most $w_{sum}/2 \leq (1+\phi)w_{sum}/2$. Case 2: MSTC lets robot r_j first move counterclockwise until it meets robot r_k. The sum of the times of the paths of robots r_j and r_k until they meet is at most $t(r_j, r_k)$. Thus, robots r_j and r_k meet after a travel time of at most $t(r_j, r_k)/2 + w_{max}/4$. The term $w_{max}/4$ takes into account that each move is atomic, and the robots might thus not be able to split the travel time evenly between them. MSTC lets robot r_j then move clockwise until it meets robot r_i. The sum of the times of the paths of robots r_j and r_i until they meet is at most $t(r_j, r_k)/2 + w_{max}/4 + t(r_j, r_k)/2 + w_{max}/4 + t(r_i, r_j)$. Thus, robots r_j and r_i meet after a travel time of at most $(t(r_j, r_k)/2 + w_{max}/4 + t(r_j, r_k)/2 + w_{max}/4 + t(r_i, r_j))/2 + w_{max}/4 = (t(r_j, r_k) + t(r_i, r_j))/2 + w_{max}/2 \leq w_{sum}/2 + w_{max}/2 = (1+\phi)w_{sum}/2$ and their travel times are thus at most $(1+\phi)w_{sum}/2$. MSTC lets robot r_k first move clockwise until it meets robot r_j and then move counterclockwise. Assume without loss of generality that robot r_l is adjacent to robot r_k in the counterclockwise direction. (Robots r_l and r_h are identical if there are only four robots, and robots r_l and r_i are identical if there are only three robots.) A similar argument as for robot r_j then shows that the travel time of robot r_k is at most $t(r_j, r_k)/2 + w_{max}/4 + t(r_j, r_k)/2 + w_{max}/4 + t(r_k, r_l) \leq w_{sum}/2 + w_{max}/2 = (1+\phi)w_{sum}/2$ since $t(r_i, r_j) > w_{sum}/2$ and thus $t(r_j, r_k) + t(r_k, r_l) < w_{sum}/2$. MSTC lets every other robot move counterclockwise and their travel time thus is at most the time of the segment in their counterclockwise direction which is at most $w_{sum}/2 \leq (1+\phi)w_{sum}/2$. The travel time of each robot and the cover time of MSTC thus is at most $(1+\phi)w_{sum}/2$. Case 3: Case 3 is just a mirror image of Case 2. - Let t_{stc} be the cover time of STC and t_{mstc} be the cover time of MSTC. Then, we have shown that $t_{mstc} \leq (1+\phi)w_{sum}/2$ in all three cases. Thus, it holds that $t_{mstc} \leq (1+\phi)w_{sum}/2 \leq (1+\phi)(t_{stc} + w_{max}/4)/2 = (1+\phi)t_{stc}/2 + (1+\phi)w_{max}/8$ since $t_{stc} \geq w_{sum} - w_{max}/4$. ■

Figure 6 shows the spanning tree and robot paths for the terrain from Figure 2 for four robots with the team objective "Cover." The cover time is 332 for MSTC. The cover and return time is 664 for MSTC and only 394 for optimized MSTC. Unfortunately, this example also demonstrates that MSTC finds solutions whose cover times (and cover and return times) 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 times of the paths of these two robots. The cover times (and cover and return times) of MSTC become arbitrarily bad compared to the minimal ones 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 have to exit the bottom-most row of large cells to minimize the cover times (and cover and return times). Thus, MSTC cannot guarantee to find solutions with good cover times (and cover and return times).

V. Multi-Robot Forest Coverage

Multi-Robot Forest Coverage (MFC) [5] finds solutions with suboptimal cover times (and cover and return times) in polynomial time for multiple robots in non-weighted terrain. MSTC determines one tree, splits the path that circumnavigates it into one path for each robot and lets each robot move along its path. MFC, on the other hand, determines one tree for each robot and lets each robot move along the path

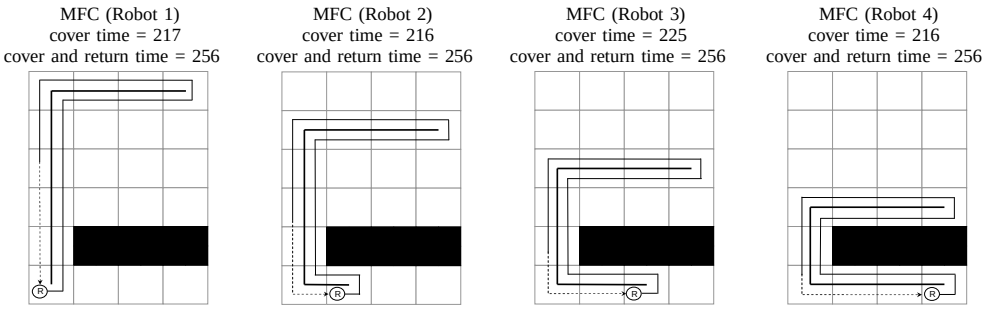

Fig. 7. Example of MFC

that circumnavigates its tree. MFC is nontrivial to generalize to weighted terrain. It uses a tree cover algorithm [2] as a subroutine that is specific to non-weighted terrain because it operates on graphs with weighted edges. We therefore build on the existing algorithm and design a tree-cover algorithm TREE COVER that is specific to weighted terrain because it operates on graphs with weighted vertices. We describe it and prove its properties in Section VI.

MFC for weighted terrain then uses TREE COVER as follows: First, MFC constructs a graph whose vertices correspond to the unblocked large cells and whose edges connect adjacent unblocked large cells. Each vertex has a weight that is equal to the weight of its large cell. Second, MFC uses TREE COVER to find a rooted tree cover of this graph, where the roots are the vertices that correspond to the large cells that contain the initial small cells of the robots. The roots thus correspond to the robots. A rooted tree cover of this graph is a forest of trees with exactly one tree for each root. Every vertex is in at least one tree. The weight of a tree is the sum of the weights of its vertices. The weight of the rooted tree cover is the largest weight of any of its trees. MFC performs a binary search (described later) that runs in polynomial time and uses TREE COVER to find a rooted tree cover with a weight that is at most a factor of $4(1 + \phi|K| + \epsilon)$ larger than minimal, where $\epsilon > 0$ is an arbitrary precision parameter that affects how often TREE COVER is called, $|K|$ is the number of robots and ϕ is the ratio of the largest weight of any large cell and the sum of the weights of all large cells. Third, MFC lets each robot move along the path that circumnavigates its tree. For the team objective "Cover and Return," each robot completely circumnavigates its tree until it returns to its initial small cell. For the team objective "Cover," the robots stop once all small cells have been visited. Clearly, MFC runs in polynomial time.

Remember that the cover times of MSTC for at least three robots are at least about a factor of $2/(1+\phi)$ smaller than the cover times of STC according to Theorem 2. MFC cannot make such a strong guarantee with respect to STC:

Theorem 3: The cover times of MFC can be larger than the cover times of STC by at most the largest weight of any small cell (that is, a quarter of the largest weight of any large cell). The cover and return times of MFC cannot be larger than the cover and return times of STC.

Proof: The cover times of STC can be smaller than the sum of the weights of all large cells by at most the largest weight of any small cell, while the cover times of MFC are at most the weights of the largest trees and thus at most the sum of the weights of all large cells. Consequently, the cover times of MFC can be larger than the cover times of STC by at most the largest weight of any small cell (that is, a quarter of the largest weight of any large cell). The cover and return times of STC are equal to the sum of the weights of all large cells, while the cover and return times of MFC are equal to the weights of the largest trees and thus at most the sum of the weights of all large cells. Consequently, the cover and return times of MFC cannot be larger than the cover and return times of STC. ■

However, MFC can make the following much more powerful guarantee with respect to the minimal cover times (and cover and return times), which MSTC cannot make:

Theorem 4: The cover times (and cover and return times) of MFC are at most about a factor of $16(1 + \phi|K| + \epsilon)$ larger than minimal, where $\epsilon > 0$ is an arbitrary precision parameter that affects how often TREE COVER is called, $|K|$ is the number of robots and ϕ is the ratio of the largest weight of any large cell and the sum of the weights of all large cells.

Proof: Let M be the weight of the rooted tree cover found by TREE COVER, N be the weight of a weight-minimal rooted tree cover, O be the cover time of MFC, P be the minimal cover time, and Q be the minimal cover time if the robots need to visit only the upper left small cells of all large cells. Furthermore, let w_{max} be the largest weight of any large cell. First, it holds that $O \leq M$ since the robots visit all small cells and return to their initial small cells when they circumnavigate their trees. The resulting cover time thus cannot be larger than the weight of the rooted tree cover. Second, it holds that $M \leq 4(1+\phi|K|+\epsilon)N$ since we use TREE COVER to find rooted tree covers with a weight that is at most a factor of $4(1+\phi|K|+\epsilon)$ larger than minimal. Third, it holds that $N/4 \leq Q + w_{max}/4$. Consider a solution where the robots need to visit only the upper left small cells of all large cells. Construct a rooted tree cover where the tree of a robot contains exactly the vertices that correspond to the large cells that contain the upper left small cells visited by the robot. The weight of each tree divided by four is equal to the sum of the weights of all upper left small cells visited by the robot. The sum of the weights of all upper left small cells visited by the robot is at most the travel time of the robot plus the largest weight of any small cell (that is, a quarter of the

largest weight of any large cell) since the robot has to enter and exit all upper left small cells except possibly for its initial small cell (if it starts in one), which it does not need to enter, and its final small cell (if it ends in one), which it does not need to exit. Thus, the weight of this rooted tree cover (and thus also the weight of a weight-minimal rooted tree cover) divided by four is at most the minimal cover time if the robots need to visit only the upper left small cells of all large cells plus a quarter of the largest weight of any large cell. Fourth, $Q \leq P$ trivially. Using these results, it holds that $O \leq M \leq 4(1+\phi|K|+\epsilon)N \leq 16(1+\phi|K|+\epsilon)Q + 4(1+\phi|K|+\epsilon)w_{max} \leq 16(1+\phi|K|+\epsilon)P + 4(1+\phi|K|+\epsilon)w_{max}$. The proof continues to hold if each occurrence of cover time is replaced with cover and return time. ■

$\phi \approx 0$ for terrain with many large cells of about the same weight. For example, $\phi = 0.0814$ for the terrain from Figure 2. Then, $16(1+\phi|K|+\epsilon) \approx 16$ for a small number of robots $|K|$ and ϵ close to zero. Thus, the cover times (and cover and return times) of MFC are at most about sixteen times larger than minimal.

Figure 7 shows the trees and robot paths for the terrain from Figure 2 for four robots, together with the cover time and cover and return time for each robot. The cover time is 225 and the cover and return time is 256 for MFC.

VI. Weight-Minimal Rooted Tree Covers

In Section V, we stated that we modified an existing tree cover algorithm [2] to work on graphs with weighted vertices rather than weighted edges. We now state the resulting algorithm (called TREE COVER), prove its properties and describe how MFC uses it.

A. The Problem

We solve the following problem: Let $G = (V, E)$ be a graph with weighted vertices, where $w(v)$ is the integer weight of vertex $v \in V$. Let $K \subseteq V$ 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. The problem is to find a weight-minimal K-rooted tree cover of graph G.

B. Definitions

We use the shorthands $w_{sum} := \sum_{v \in V} w(v)$, $w_{max} := \max_{v \in V} w(v)$ and $\phi := w_{max}/w_{sum}$ (as used earlier). Furthermore, we define the weight of a path in the graph to be the sum of the weights of its vertices, except for its end vertices. We define the distance between 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.

C. NP-Hardness

We show that finding weight-minimal K-rooted tree covers is NP-hard, which provides our motivation for designing approximation algorithms that run in polynomial time.

Theorem 5: Finding a weight-minimal K-rooted tree cover for graphs G is NP-hard.

Proof: We reduce BINPACKING to our problem. BINPACKING consists of a set of elements with given integer sizes and a fixed number of bins, each with the same given integer capacity. The problem is to determine whether 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. Given an instance of BINPACKING, we transform it in polynomial time to an instance of the problem of determining whether the weight of a weight-minimal K-rooted tree cover for graph G is at most a given constant, as follows: We create 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. If the weight of a weight-minimal K-rooted tree cover is at most the given capacity plus one, then 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, by placing each element in one of the bins whose vertex is the root of a tree that contains the vertex of the element. Similarly, if 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, then one can construct a K-rooted tree cover whose weight is at most the given capacity plus one, by making the tree rooted in the vertex of a bin contain the vertices of the elements that the bin contains. Thus, the weight of a weight-minimal K-rooted tree cover is at most the given capacity plus one as well. ■

D. Our Algorithm

We now describe TREE COVER, a tree-cover algorithm inspired by [2] that takes as input a graph G, a set of roots K and a bound $B \geq w_{max}$. It 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 $B/(1+\phi|K|)$. TREE COVER operates as follows:

1) Contract all roots into a single vertex, find any spanning tree of the resulting graph, and then uncontract the single vertex again, splitting the spanning tree into $|K|$ trees.
2) Decompose each tree recursively into zero or more non-leftover subtrees and one leftover subtree. We call the following decomposition procedure once for each tree from the previous step. The decomposition procedure removes vertices from the given tree as it generates the non-leftover subtrees. When it terminates, we declare the leftover subtree to be the root of the given tree if all vertices have been deleted. Otherwise, we declare the leftover subtree to be the remaining tree (formed by the non-deleted vertices).
   The decomposition procedure applies to a tree rooted in r. We distinguish three cases:
   Case 1: The weight of the tree rooted in r is less than B. Then, we simply return.
   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. We remove the subtree from the tree rooted in r (leaving the empty tree) and return.

Case 3: The weight of the tree rooted in r is $2B$ or larger. We distinguish three subcases:

Case 3a: The weights of all trees rooted in children of r are less than B. Then, we pick 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 $[B, 2B)$. One non-leftover subtree consists of r and these trees. We remove the subtree except for r from the tree rooted in r and recursively apply the decomposition procedure to the remaining tree rooted in r in order to find the other non-leftover subtrees. It is possible to pick 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 $[B, 2B)$ since the weight of r is at most B (since $B \geq w_{max}$) 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 $2B$ or larger.

Case 3b: The weight of at least one tree rooted in a child of r is in the interval $[B, 2B)$. Then, we pick such a tree. One non-leftover subtree consists of this tree. We remove the subtree from the tree rooted in r and recursively apply the decomposition procedure to the remaining tree rooted in r in order to find the other non-leftover subtrees.

Case 3c: Otherwise, the weight of at least one tree rooted in a child of r is $2B$ or larger. Then, we recursively apply the decomposition procedure to that tree and then to the remaining tree rooted in r in order to find the non-leftover subtrees.

3) Find a maximum matching 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 of at most B.
4) If any non-leftover subtree 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 2 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 the main properties of TREE COVER.

Theorem 6: 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 of weight at most B, the single non-leftover subtree (if any) matched to the root of weight at most $2B$, and a weight-minimal path (if any) of weight at most B from the non-leftover subtree to the leftover subtree. The weight of each tree is thus at most $4B$, resulting in a K-rooted tree cover of weight at most $4B$. ■

Theorem 7: If TREE COVER reports FAILURE, then there does not exist a K-rooted tree cover of graph G with weight at most $B/(1 + \phi|K|)$.

Proof: Assume that a weight-minimal K-rooted tree cover of graph G has weight B' with $B' \leq B/(1 + \phi|K|)$. Let L be the set of non-leftover subtrees created in Step 2 of TREE COVER and $K(l) \subseteq K$ be the set of roots that can be matched to non-leftover subtree $l \in L$ because the non-leftover subtree and the leftover subtree of the root are at distance of at most B. We show that $|\cup_{l \in L'} K(l)| \geq |L'|$ for every set of non-leftover subtrees $L' \subseteq L$. Step 3 of TREE COVER can then match all non-leftover subtrees according to Hall's Marriage Theorem. Therefore, TREE COVER reports SUCCESS and not FAILURE, which proves the contrapositive of the theorem and thus also the theorem itself.

Consider any $L' \subseteq L$. Let T be the set of trees of a weight-minimal K-rooted tree cover of graph G and $T' \subseteq T$ be the set of trees which have at least one vertex in common with at least one of the non-leftover subtrees in L'. Let $w(L')$ be the sum of the weights of all non-leftover subtrees in L' and $w(T')$ be the sum of the weights of all trees in T'. First, it holds that $B' \geq w_{sum}/|K|$ since the sum of the weights of all vertices can be split evenly among the trees in the best case. Second, it holds that $w(L') \geq B|L'|$ since the weights of all non-leftover subtrees in L' are in the interval $[B, 2B)$ and thus B or larger. Third, it holds that $w(T') \leq B'|T'|$ since the weights of all trees in T' are at most B' since any weight-minimal K-rooted tree cover of graph G has weight B'. Fourth, it holds that $|\cup_{l \in L'} K(l)| \geq |T'|$. For every tree in T', there exists at least one non-leftover subtree in L' that has at least one vertex in common with the tree in T'. Then, the non-leftover subtree in L' and the root of the tree in T' are at distance of at most $B' \leq B/(1 + \phi|K|) \leq B$. The non-leftover subtree in L' can thus be matched to the root of the tree in T'. Overall, the set $\cup_{l \in L'} K(l)$ contains the roots of all trees in T'. Its cardinality thus is at least the cardinality of T'. Fifth, it holds that $w(L') \leq w(T') + w_{max}|L'|$ since every vertex in at least one non-leftover subtree in L' is also in at least one tree in T'. The non-leftover subtrees in L' can contain at most $|L'|$ duplicate vertices, each with weight at most w_{max}: Every non-leftover subtree that Step 2 of TREE COVER creates contains at most one vertex that has not yet been removed from all trees created in Step 1 and thus could be a duplicate vertex. This statement holds because the trees created in Step 1 share at most their roots and Step 2 removes all vertices of a non-leftover subtree from its tree, except possibly for the root of the non-leftover subtree in Case 3a, when it creates the non-leftover subtree. Using these results, it holds that $w_{max} = w_{sum}\phi \leq |K|B'\phi$. This inequality implies that $B'|T'| \geq w(T') \geq w(L') - w_{max}|L'| \geq B|L'| - |K|B'\phi|L'| = (B - |K|B'\phi)|L'| \geq (B'(1 + \phi|K|) - |K|B'\phi)|L'| = B'|L'|$. This inequality in turn implies that $|\cup_{l \in L'} K(l)| \geq |T'| \geq |L'|$, which is what we wanted to prove. ■

F. Application

We perform binary search on the interval $[w_{max}, w_{sum}]$ 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 average 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 the given value of the arbitrary precision parameter $\epsilon > 0$. We then return the K-rooted tree cover of graph G returned by TREE COVER with B set to the upper bound.[1]

Let b be the weight of a weight-minimal K-rooted tree cover of graph G. We assume in the following that $b \geq 1$ since this property holds for MFC due to the weight of each unblocked large cell being a positive integer. Let B_l be the lower bound and B_u be the upper bound of the binary search after termination. First, it holds that $B_u - B_l \leq \epsilon$ according to the termination criterion. Second, it holds that $b \geq B_l/(1 + \phi|K|)$ according Theorem 7 since TREE COVER with B set to B_l reports FAILURE. (This statement also holds for $B_l = w_{max}$, the initial lower bound, since $b \geq w_{max}$.) Third, the weight of the K-rooted tree cover of graph G returned by the binary search is at most $4B_u$ according to Theorem 6 since TREE COVER with B set to B_u reports SUCCESS. (This statement also holds for $B_u = w_{sum}$, the initial upper bound, since TREE COVER then generates at most one non-leftover subtree for each root, which contains the root.) Using these results, it holds that the weight of the K-rooted tree cover of graph G returned by the binary search is at most $4B_u \leq 4(B_l + \epsilon) \leq 4(1 + \phi|K|)b + 4\epsilon \leq 4(1+\phi|K|+\epsilon)b$, which is at most a factor of $4(1+\phi|K|+\epsilon)$ larger than minimal. The binary search runs in polynomial time because TREE COVER runs in polynomial time and is run $\lceil \log_2((w_{sum} - w_{max})/\epsilon) \rceil \leq \log_2 w_{sum} - \log_2 \epsilon + 1$ times, which is polynomial in the size of the input for a constant value of ϵ.

VII. Experimental Results

We now compare MFC (with a small value for the precision parameter) and MSTC experimentally. We evaluate them on both team objectives, namely "Cover" and "Cover and Return", and in different scenarios, namely different kinds of terrain [terrain], different numbers of robots [robots], and different clustering of the robots [clustering]. The size of the terrain is always 49×49 large cells. The weight of each large cell is always chosen uniformly at random from the weights 8, 16, 24, ..., 80. Figure 8 shows the three different kinds of terrain used in the experiments. The first kind of terrain is empty [empty]. The second kind is an outdoor-like terrain where walls are randomly removed from a random depth-first maze until the wall density drops to 10 percent, resulting in terrain with random obstacles [outdoor]. The third kind is an indoor-like terrain with walls and doors [indoor]. The position 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 robots. 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 lower left small cells. A clustering percentage parameter x determines how strongly the initial large cells of the robots are clustered. The first robot is placed uniformly at random. Subsequent robots are then placed within 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 high clustering of initial large cells, while $x = 200$ is equivalent to no clustering at all [none]. For each scenario, we report data that has been averaged over 50 runs with randomly generated terrain (if applicable) and randomly generated initial small cells. All cover times and cover and return times have been rounded to the nearest integer.

Table 9 reports for each scenario a lower bound that represents an idealized cover time (and cover and return time) [ideal max]: It simply divides the sum of the weights of all large cells by the number of robots. The ideal cover time (and cover and return time) would result if no robot needed to pass through already visited small cells. The table also reports the smallest [min] and largest [max] travel time of any robot for each combination of a multi-robot coverage algorithm, scenario and team objective. The largest travel time is equal to the cover time (or cover and return time), and the difference between the smallest and largest travel times gives an indication of how balanced the travel times of the robots are. In addition, the table also reports the ratio of the actual travel time and the ideal cover time (and cover and return time) [ratio], giving an upper bound on how far the actual cover time (or cover and return time) is larger than minimal. The ratio is indeed only an upper bound, since the ideal may not be achievable. For instance, several robots must visit the same small cells in the example from Figure 7.

We make the following observations: The ratio of the cover time (or cover and return time) and the ideal cover time (and cover and return time) increases with the number of robots for both MFC and MSTC since the overhead (defined as the number of already visited small cells that a robot passes through) increases with the number of robots. The ratio increases very slowly with the number of robots for MFC, but much faster for MSTC, implying that the cover times (and cover and return times) of MFC remain close to minimal for large numbers of robots. The ratio changes insignificantly with the amount of clustering for MFC, but a lot for MSTC, implying that the cover times (and cover and return times) of MFC remain small if robots start in nearby small cells – a common situation since robots are often deployed or stored together. The ratio changes insignificantly for MFC if the team objective is changed from "Cover" to "Cover and Return", but increases by about a factor of two for non-optimized MSTC (because the robot with the largest

[1]Our description generalizes easily since it does not take into account that the weights of the vertices are integers. A weight-minimal K-rooted tree cover can be inferred after a binary search for $\epsilon < 1$ if the weights of the vertices are integers.

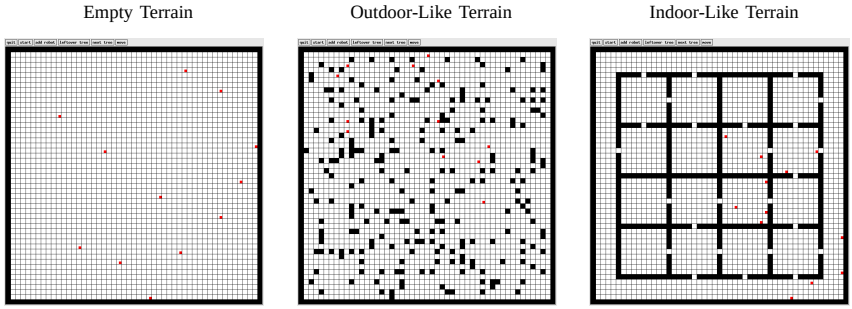

Fig. 8. Screenshots of Different Kinds of Terrain

Terrain	Robots	Clustering	Ideal Max	MFC						MSTC						Optimized MSTC					
				"Cover and Return"			"Cover"			"Cover and Return"			"Cover"			"Cover and Return"			"Cover"		
				Max	(Min)	Ratio	Max	(Min)	Ratio	Max	(Min)	Ratio	Max	(Min)	Ratio	Max	(Min)	Ratio	Max	(Min)	Ratio
Empty	2	30	45094	47369	(43018)	1.07	47353	(10612)	1.07	96669	(83672)	2.19	48340	(46595)	1.10	48865	(42446)	1.11	48334	(41325)	1.10
	2	60	45094	47840	(42516)	1.09	47825	(10621)	1.08	100558	(79782)	2.27	50284	(48821)	1.14	50940	(40688)	1.15	50279	(40101)	1.14
	2	none	45094	48061	(42334)	1.09	48028	(10613)	1.09	104811	(75532)	2.37	52409	(48862)	1.19	53082	(38625)	1.20	52406	(38078)	1.19
	8	30	11273	12698	(9676)	1.15	12645	(9208)	1.14	73870	(411)	6.67	36967	(206)	3.34	37506	(261)	3.38	36938	(207)	3.33
	8	60	11273	12765	(10058)	1.16	12749	(9549)	1.15	72479	(1106)	5.54	36240	(559)	2.27	36883	(668)	3.33	36240	(559)	2.27
	8	none	11273	13726	(8983)	1.24	13699	(8729)	1.24	54885	(2519)	4.94	27453	(1259)	2.47	28026	(1511)	2.52	27445	(1260)	2.47
	14	30	6442	7620	(5396)	1.21	7586	(5359)	1.20	72107	(37)	11.41	36054	(19)	5.71	36639	(37)	5.80	36054	(19)	5.71
	14	60	6442	7620	(5208)	1.21	7581	(5166)	1.20	69594	(177)	11.01	34797	(89)	5.51	35441	(149)	5.61	34797	(89)	5.51
	14	none	6442	8004	(4768)	1.27	7977	(4719)	1.26	43131	(616)	6.71	21566	(308)	3.35	22099	(438)	3.44	21566	(308)	3.35
	20	30	4509	5575	(3487)	1.26	5505	(3466)	1.24	70424	(19)	15.93	35214	(9)	7.97	35810	(19)	8.10	35214	(9)	7.97
	20	60	4509	5460	(3666)	1.23	5428	(3628)	1.23	67842	(93)	15.39	33922	(48)	7.69	34553	(93)	7.84	33921	(48)	7.69
	20	none	4509	5736	(3093)	1.29	5704	(3054)	1.28	33042	(280)	7.50	16521	(140)	3.75	17028	(254)	3.87	16251	(140)	3.75
Outdoor	2	30	40586	43430	(37877)	1.09	43418	(10612)	1.09	86654	(75655)	2.18	43330	(42868)	1.09	43927	(38497)	1.10	43327	(37933)	1.09
	2	60	40586	43677	(37652)	1.10	43664	(10600)	1.10	91671	(70637)	2.29	45841	(42694)	1.15	46410	(36050)	1.16	45836	(35512)	1.15
	2	none	40586	43910	(37472)	1.10	43884	(10652)	1.10	94781	(67529)	2.38	47396	(42937)	1.19	48083	(34655)	1.21	47390	(34071)	1.19
	8	30	10146	11679	(8657)	1.17	11622	(8484)	1.17	66563	(303)	6.72	33287	(153)	3.36	33847	(209)	3.42	33283	(153)	3.36
	8	60	10146	11677	(8526)	1.17	11633	(8436)	1.17	58422	(1131)	5.88	29270	(573)	2.94	29834	(691)	2.99	29223	(570)	2.94
	8	none	10146	12124	(8248)	1.22	12078	(8164)	1.21	54687	(1988)	5.47	27355	(1004)	2.74	27999	(1229)	2.80	27347	(1000)	2.74
	14	30	5798	6919	(4876)	1.22	6838	(4835)	1.20	63965	(41)	11.29	31983	(21)	5.65	32580	(40)	5.75	31983	(21)	5.65
	14	60	5798	6803	(4877)	1.20	6752	(4842)	1.19	56196	(245)	9.92	28098	(123)	4.96	28645	(198)	5.06	28098	(124)	4.96
	14	none	5798	7253	(4446)	1.28	7208	(4386)	1.27	43183	(671)	7.53	21592	(335)	3.77	22177	(453)	3.87	21592	(335)	3.77
	20	30	4059	5240	(2945)	1.32	5170	(2918)	1.30	63018	(26)	18.95	31509	(13)	7.97	32056	(26)	8.11	31509	(13)	7.97
	20	60	4059	5041	(3341)	1.27	4995	(3275)	1.25	56366	(97)	14.22	28183	(48)	7.11	28743	(82)	7.25	28183	(48)	7.11
	20	none	4059	5203	(2811)	1.31	5179	(2778)	1.30	34814	(285)	8.68	17407	(142)	4.34	17998	(214)	4.49	17407	(142)	4.34
Indoor	2	30	38212	41237	(35599)	1.10	41225	(10612)	1.10	81616	(71193)	2.18	40815	(39557)	1.09	41609	(36585)	1.11	40808	(35898)	1.09
	2	60	38212	41091	(35923)	1.10	41028	(10612)	1.10	85686	(67123)	2.28	42849	(41000)	1.14	43726	(34840)	1.17	42843	(33955)	1.14
	2	none	38212	40784	(36339)	1.09	40678	(10625)	1.09	88988	(63823)	2.38	44500	(39984)	1.19	45528	(33535)	1.22	44494	(32470)	1.19
	8	30	9553	11703	(8323)	1.25	11556	(8197)	1.23	60767	(195)	6.50	30421	(103)	3.26	31336	(140)	3.35	30390	(101)	3.25
	8	60	9553	11522	(8464)	1.23	11440	(8346)	1.22	55229	(815)	5.85	27620	(408)	2.93	28670	(502)	3.04	27616	(408)	2.93
	8	none	9533	11602	(8049)	1.24	11516	(7903)	1.23	49818	(1925)	5.31	24909	(962)	2.66	25926	(1114)	2.77	24909	(962)	2.66
	14	30	5459	7815	(4044)	1.46	7686	(3988)	1.43	58513	(35)	10.93	29256	(17)	5.46	30242	(33)	5.65	29256	(17)	5.46
	14	60	5459	7353	(4024)	1.37	7227	(3983)	1.35	52785	(219)	9.85	26393	(111)	4.93	27358	(156)	5.11	26392	(111)	4.93
	14	none	5459	6937	(4128)	1.30	6871	(4047)	1.28	37708	(646)	7.04	18854	(323)	3.52	19782	(410)	3.70	18854	(323)	3.52
	20	30	3821	6669	(1175)	1.77	6536	(1146)	1.74	56833	(20)	15.14	28446	(10)	7.57	29434	(19)	7.84	28421	(10)	7.57
	20	60	3821	5936	(1824)	1.57	5824	(1791)	1.55	50182	(88)	13.50	25091	(44)	6.75	25985	(74)	6.99	25091	(44)	6.75
	20	none	3821	5198	(2288)	1.39	5133	(2238)	1.37	32374	(382)	8.63	16187	(191)	4.32	17040	(264)	4.55	16187	(191)	4.32

Fig. 9. Experimental Results for MFC and MSTC ("Max" = Cover Time or Cover and Return Time)

travel time has to backtrack along most of its trajectory), implying that all robots are close to their initial small cells when coverage is complete for MFC, which facilitates their retrieval. The cover and return times of optimized MSTC are significantly smaller than the ones of non-optimized MSTC for each scenario. Overall, MFC results in much smaller cover times (and cover and return times) than MSTC for more than two robots and in comparable cover times (and cover and return times) than optimized MSTC for two robots. (These results might depend on the initial spanning trees.)

Theorem 4 guarantees that the cover times (and cover and return times) of MFC are at most a factor of about sixteen 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 terrain and $\phi = 10.5 \times 10^{-4}$ for indoor terrain. Empirically, however, the cover times (and cover and return times) of MFC are at most a factor of 1.77 larger than minimal and thus significantly smaller than the cover times (and cover and return times) guaranteed by Theorem 4.

VIII. Conclusion

We extended Multi-Robot Forest Coverage, a state-of-the-art multi-robot coverage algorithm, from terrain with uniform traversability to terrain with non-uniform traversability. Currently, Multi-Robot Forest Coverage assumes ideal robots. It is future work to generalize it to robots with actuator and sensor uncertainty and other typical imperfections.

References

[1] H. Choset. Coverage for robotics – a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

[2] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Min-max tree covers of graphs. *Operations Research Letters*, 32:309–315, 2004.

[3] 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.

[4] N. Hazon and G. Kaminka. Redundancy, efficiency, and robustness in multi-robot coverage (in print). In *Proceedings of the International Conference on Robotics and Automation*, pages 735–741, 2005.

[5] X. Zheng, S. Jain, S. Koenig, and D. Kempe. Multi-robot forest coverage. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2318–2323, 2005.