Multi-Robot Coverage of 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 terrain coverage: visiting each location in known terrain. 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. In this article, we describe Multi-Robot Forest Coverage (MFC), a new multi-robot coverage algorithm based on an algorithm by Even et al. for finding a tree cover with trees of balanced weights. We then extend MFC and an alternative existing multi-robot coverage algorithm from unweighted terrain to weighted terrain. Our theoretical results show that the cover time of MFC is at most about sixteen and eight times larger than minimal in weighted and unweighted, respectively, terrain. Our experimental results show that the cover time of MFC is close to minimal in all tested scenarios and smaller than the cover time of the alternative multi-robot coverage algorithm.**

***Index Terms*—Cell Decomposition, Complexity, Multi-Robot Coverage, NP-Hardness, 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. It is frequently desirable to minimize the time by which coverage is completed. In recent years, robotics researchers have investigated spanning tree-based coverage algorithms in unweighted terrain, where the travel time is the same everywhere. The single-robot coverage problem is solved with minimal cover time by Spanning Tree Coverage (STC), a polynomial-time coverage algorithm that decomposes terrain into cells, computes a spanning tree of the resulting graph, and makes the robot circumnavigate it [3]. Naturally, coverage can be sped up with multiple robots. The multi-robot coverage problem is to compute a robot path for each robot so that the cover time (that is, largest travel time of any robot) is minimized. As we show in this article, this problem is NP-complete. Thus, one needs to design polynomial-time multi-robot coverage algorithms with suboptimal (but small) cover time. Hazon and Kaminka recently generalized STC to Multi-Robot Spanning Tree Coverage (MSTC), a polynomial-time multi-robot coverage algorithm [5]. While MSTC provably improves the cover time of STC, it cannot guarantee its cover time to be close to minimal.

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

Manuscript received August 31, 2007.

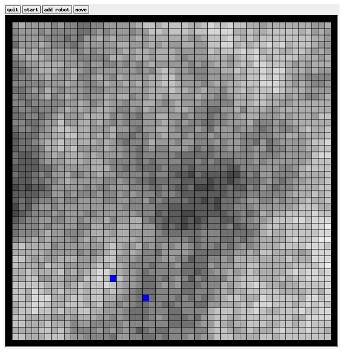

Fig. 1. Example of Weighted Terrain

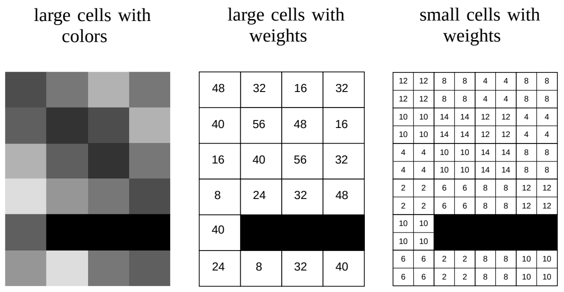

Fig. 2. Model of Weighted Terrain

We generalize STC to Multi-Robot Forest Coverage (MFC), a polynomial multi-robot coverage algorithm based on finding a tree cover with trees of balanced weights, one tree for each robot. We then extend both MSTC and MFC from unweighted terrain to weighted terrain, where the travel time is not the same everywhere, as shown in Figure 1, in order to extend their applicability to more realistic situations. MSTC can be generalized relatively easily but cannot guarantee its cover time to be small. 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 time of MFC is at most about sixteen and eight times larger than minimal in weighted and unweighted terrain respectively. Our experimental results show that the cover time of MFC is close to minimal in all tested scenarios and smaller than the cover time of MSTC. MFC has the additional benefit that it tends to return the robots close to their initial cells, facilitating their collection and storage.

II. Problem Description

Terrain: We model terrain as consisting of large square cells. Each large cell is either entirely blocked or entirely unblocked.

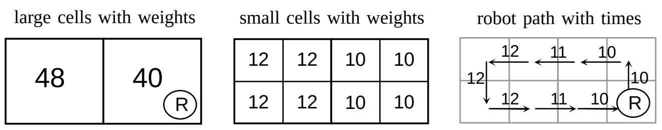

Fig. 3. Simple Single-Robot Coverage Problem

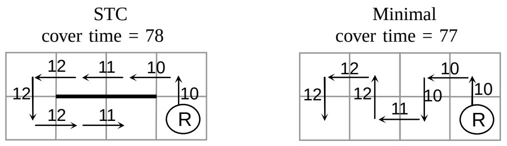

Fig. 4. Suboptimal Cover Time of STC

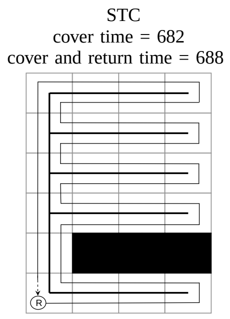

Fig. 5. Example of STC

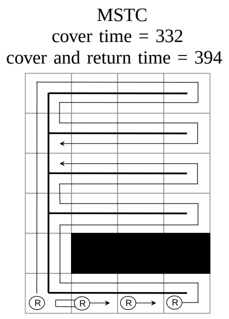

Fig. 6. Example of MSTC

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. We call terrain *unweighted* if the weight of all large cells is four, which implies that the weight of all small cells is one and thus the travel time along any robot path is equal to the number of moves. We call terrain *weighted* if the weights of the large cells can be arbitrary positive integers. Thus, unweighted terrain is a special case of weighted terrain.

Robots: The robots have the same size as the small 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. (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, it 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. The robots start in different large cells but are able to occupy the same small cell simultaneously without blocking each other.

Team Objective: 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 single-robot coverage problem, 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.

III. Spanning Tree Coverage (STC)

Spanning Tree Coverage (STC) [3] solves single-robot coverage problems. It was originally proposed for unweighted terrain but also applies unchanged to weighted terrain: 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. The cover times (and cover and return times) of STC are minimal for single-robot coverage problems in unweighted terrain since the robot never enters any small cell twice [3].

Proposition 1: The cover times of STC in weighted terrain are larger than minimal by at most the half of the largest weight of any small cell. The cover and return times of STC in weighted (and thus also unweighted) terrain are minimal for single-robot coverage problems terrain. The minimal cover and return times are equal to the sums of the weights of all large cells.

Proof: For the team objective "Cover," 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. The final small cell of a robot that uses STC 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 half of the largest weight of any small cell. For the team objective "Cover and Return," the robot needs to enter and exit every small cell at least once. 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 most 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. ∎

Thus, the cover times of STC are not necessarily minimal in weighted terrain, as shown in Figure 4 for the single-robot coverage problem from Figure 3 (the thick line shows the spanning tree), but it finds close-to-minimal cover times. The cover and return times of STC are minimal in weighted terrain.

For illustration, 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. Complexity of Multi-Robot Coverage

Coverage with multiple robots has received much less attention than coverage with single robots, even though it often results in smaller cover (and return) times. Unfortunately, it is computationally complex to minimize the cover (and return) times for multi-robot coverage problems, as we show in the following for two natural versions of multi-robot coverage problems. We thus do not expect to be able to solve them with minimal cover (and return) times in polynomial time, and it becomes necessary to solve them with suboptimal cover (and return) times.

Theorem 2: It is NP-complete to determine whether the following two versions of multi-robot coverage problems can be solved with cover and return times (for the first version) or cover times (for the second version) that are smaller than a given value:

1) multi-robot coverage problems with n robots for the team objective "Cover and Return," where n is part of the input and the times of moving from one small cell to an adjacent one are uniform; and
2) multi-robot coverage problems with two robots for the team objective "Cover," where the times of moving from one small cell to an adjacent one can be non-uniform (and large).

Proof: Both versions of the multi-robot coverage problem are in NP since one can easily guess the robot paths and then verify the travel times along them in polynomial time. To prove their NP-hardness, we reduce from partitioning problems.

1) The 3-Partition problem is defined as follows: Given a positive integer B and positive integers $a_1, \ldots, a_{3n}$ strictly between $B/4$ and $B/2$ with $\sum_{i=1}^{3n} a_i = B \cdot n$, can they be partitioned evenly into n sets? The 3-Partition problem is known to be strongly NP-complete [4], that is, NP-hard even if the integers have sizes that are only polynomial in n.
   Given an instance of it, we construct a multi-robot coverage problem with n robots as follows: We start with a "corridor" consisting of $4n$ vertically adjacent large cells, numbered from $-n+1$ (bottom) to $3n$ (top). For $i = 1, \ldots, 3n$, there is a "tunnel" of $a_i \cdot 6n$ horizontally adjacent large cells. The tunnel is connected to the i^{th} corridor cell. The i^{th} tunnel is to the left of the corridor for odd i and to the right of the corridor for even i. The n robots start in corridor cells $0, -1, \ldots, -n+1$, one robot in each of the cells. This completes the construction, which can be done in polynomial time. We claim that the minimal cover time is at most $B \cdot 24n + 16n$ if and only if the given integers can be partitioned evenly into n sets.
   If 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 $i \in S_j$, ending in its initial cell. It thus traverses the tunnels for a travel time of at most $\sum_{i \in S_j} 4 \cdot a_i \cdot 6n = B \cdot 24n$, and the corridor for a travel time of at most $12n + 4n$ (where the $4n$ is an upper bound on the time to get to cell 1 of the corridor and back at the end). The total travel time is thus at most $B \cdot 24n + 16n$, meeting the requirement.
   Conversely, if the robots cover all small cells with the desired cover time, then let S_j be the set of indices i such that the j^{th} robot is the first robot to cover the small upper cell of the i^{th} tunnel cell that is farthest away from the corridor. These sets partition the given integers. The total travel time of the j^{th} robot is at least $24n \sum_{i \in S_j} a_i$, since it needs to traverse its tunnels in both directions to return to its initial small cell, and needs two moves to traverse each large tunnel cell. By assumption, the total travel time of any robot is at most $B \cdot 24n + 16n$, which implies that $\sum_{i \in S_j} a_i \leq B + \frac{2}{3}$. Since both $\sum_{i \in S_j} a_i$ and B are integers, we have that $\sum_{i \in S_j} a_i \leq B$ for all sets S_j. Since $\sum_{j=1}^{n} \sum_{i \in S_j} a_i = \sum_{i=1}^{3n} a_i = B \cdot n$, all inequalities are equalities, and the sets S_j partition the given integers evenly.
2) This reduction has to be slightly adapted to prove the NP-hardness of the second version of the multi-robot coverage problem. The Partition problem is defined as follows: Given a set of k positive integers, 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 [4]. But then, building tunnels of length $a_i \cdot 6n$ cannot necessarily be done in polynomial time. Instead, we collapse each tunnel to a single large cell, with weight $a_i \cdot 6n$. Once we use non-uniform cell weights, we can also avoid the requirement that the robots return to their initial small cells, by adding two more large "destination cells", with weights $8n^2 \cdot A$, where $A := \sum_i a_i$. Since there are only two robots, we only add two corridor cells $0, -1$ at the bottom, where these two robots start. We claim that there is a valid partition if and only if the robots can cover all cells in time at most $28n^2A + 12nA + 4n$.
   If there is a partition $(S, \overline{S})$, then we can assign one robot all tunnel cells $i \in S$, and the other all $i \notin S$. Each robot gets one destination cell. The total time in tunnel cell i is $a_i \cdot 24n$ (three transitions of cost $a_i \cdot 6n$, and two of cost $(a_i \cdot 6n)/2$, each half of a move to enter or leave the large tunnel cell), so the total of all tunnel and corridor cells for each robot is at most $4n + A/2 \cdot 24n$. Travel in the destination cell takes time $3 \cdot 8n^2A + (8n^2A)/2 \leq 28n^2A$, for a total of at most $28n^2A + 12nA + 4n$, meeting the desired bounds.
   Conversely, if the time is at most $28n^2A + 12nA + 4n$, then each robot can only visit one destination cell, and spends a total of $28n^2A$ there. Thus, each robot spends at most $12nA + 4n$ in tunnel and corridor cells. Define $S, \overline{S}$ as in the previous proof. We next claim that if $i \in S$,

i.e., the first robot visits tunnel cell i's upper corner first, then it also visits the corresponding lower corner. For if not, then each robot spends at least time $18na_i$ in tunnel cell i, and the total time spent between the two robots in all other tunnel cells is at least $\sum_{j \neq i} a_j \cdot 24n$. But then, the total time they spend is at least $24nA + 12na_i > 24nA+8n$, so at least one of them must spend more than $12nA + 4n$ in corridor and tunnel cells, a contradiction. Given that each robot visits both of the far two cells of each tunnel cell, it must spend at least $24na_i$ in tunnel cell i. Thus, the first robot spends at least $\sum_{i \in S} 24na_i$ in its tunnel cells, and the second one at least $\sum_{i \in \overline{S}} 24na_i$. Because $\sum_{i \in S} 24na_i \leq 12nA+4n$, and all numbers are integers, we get that $\sum_{i \in S} a_i \leq A/2$, and similarly for $\sum_{i \in \overline{S}}$. This implies that $(S, \overline{S})$ is a partition.

■

The first version of the multi-robot coverage problem is for unweighted terrain but the second version is for a more general version of weighted terrain than we consider in the following. Currently, it is an open problem whether the first version of the multi-robot coverage problem is NP-hard for the team objective "Cover." It is also an open problem whether the first version is NP-hard for a fixed number of robots, although we conjecture it to be.

V. Multi-Robot Spanning Tree Coverage (MSTC)

An overview of multi-robot coverage algorithms is given in [1]. Many multi-robot coverage algorithms are for robots that interact and plan only locally [7], often called ant robots [6], even though global planning can lead to significantly smaller cover (and return) times since it allows the robots to coordinate much better. Recently, STC was generalized to Multi-Robot Spanning Tree Coverage (MSTC) [5]. (The backtracking version of) MSTC computes suboptimal cover (and return) times in polynomial time for multi-robot coverage problems in unweighted terrain, as follows: MSTC first 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 [5], then ensure that MSTC reduces the cover times of STC in unweighted terrain by a factor of at least two for three or more robots.

We now generalize MSTC to weighted terrain, assuming for simplicity that there are three or more robots: First, MSTC 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. 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. 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_1 (r_2 and r_3, respectively) is adjacent to robot r_4 (r_1 and r_2, respectively) in the counterclockwise direction and that $t(r_1, r_2)$ is the largest travel time along the segments. (Robots r_3 and r_4 are identical if there are only three robots.) We distinguish several cases:

1) 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.
2) Case 2: If $t(r_2, r_3) \leq t(r_4, r_1)$, then MSTC lets robot r_2 first move counterclockwise until it is in an adjacent small cell to robot r_3 (i.e. it meets robot r_3) and then move clockwise, lets robot r_3 first move clockwise until it meets robot r_2 and then move counterclockwise, and lets all other robots move counterclockwise.
3) Case 3: If $t(r_2, r_3) > t(r_4, r_1)$, then MSTC lets robot r_4 first move counterclockwise until it meets robot r_1 and then move clockwise, lets robot r_1 first move clockwise until it meets robot r_4 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 travel times from their current small cells to their initial small cells (optimized MSTC). Each small cell is visited by only one robot, so there are never any collisions or blockages. Clearly, MSTC runs in polynomial time.

Theorem 3: The cover times of MSTC in weighted terrain for three or more 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 is equal to the travel time along the path that circumnavigates the spanning tree). Then $\phi = w_{max}/w_{sum}$.

1) 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 $w_{sum}/2$.
2) Case 2: MSTC lets robot r_2 first move counterclockwise until it meets robot r_3. The sum of the travel times of robots r_2 and r_3 until they meet is at most $t(r_2, r_3)$. Thus, robots r_2 and r_3 meet after a travel time of at most $t(r_2, r_3)/2 + w_{max}/4$. 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_2 then move clockwise until it meets robot r_1. The sum of the travel times of robots r_2 and r_1 until they meet is at most $t(r_2, r_3)/2+w_{max}/4+t(r_2, r_3)/2+w_{max}/4+t(r_1, r_2)$. Thus, robots r_2 and r_1 meet after a travel time of at

most $(t(r_2, r_3)/2 + w_{max}/4 + t(r_2, r_3)/2 + w_{max}/4 + t(r_1, r_2))/2 + w_{max}/4 = (t(r_2, r_3) + t(r_1, r_2))/2 + w_{max}/2 \leq w_{sum}/2 + w_{max}/2 = (1 + \phi)w_{sum}/2$ and their travel times are thus at most $(1+\phi)w_{sum}/2$. MSTC lets robot r_3 first move clockwise until it meets robot r_2 and then move counterclockwise. Assume without loss of generality that robot r_5 is adjacent to robot r_3 in the counterclockwise direction. (Robots r_5 and r_4 are identical if there are only four robots, and robots r_5 and r_1 are identical if there are only three robots.) A similar argument as for robot r_2 then shows that the travel time of robot r_3 is at most $t(r_2, r_3)/2+w_{max}/4+t(r_2, r_3)/2+w_{max}/4+t(r_3, r_5) \leq w_{sum}/2+w_{max}/2 = (1 + \phi)w_{sum}/2$ since $t(r_1, r_2) > w_{sum}/2$ and thus $t(r_2, r_3) + t(r_3, r_5) < w_{sum}/2$. MSTC lets every other robot move counterclockwise and their travel time is thus at most the travel time along the segment in their counterclockwise direction which is at most $w_{sum}/2$. Thus the travel time of each robot and the cover time of MSTC is at most $(1 + \phi)w_{sum}/2$.

3) 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, $t_{mstc} \leq (1 + \phi)w_{sum}/2 \leq (1 + \phi)(t_{stc}+w_{max}/4)/2 = (1+\phi)t_{stc}/2+(1+\phi)w_{max}/8$ since $t_{stc} \geq w_{sum} - w_{max}/4$. ■

For illustration, 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. This example demonstrates that the cover (and return) 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 (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. For then all of the robots would have to exit the bottom-most row of large cells to minimize the cover (and return) times. Thus, MSTC cannot guarantee small cover (and return) times, which is due to the fact that the construction of the spanning tree does not take into account that it will be split up afterwards, resulting in unbalanced travel times of the robots. This observation motivates our idea of constructing a tree cover with one tree for each robot right away, where we ensure during the construction that the weights of the trees and thus the travel times of the robots are balanced.

VI. Multi-Robot Forest Coverage (MFC)

We now generalize STC to Multi-Robot Forest Coverage (MFC) [8]. MFC computes suboptimal cover times (and cover and return times) in polynomial time for multi-robot coverage problems. 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 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 is allowed to be disconnected, so long as each of its components contains at least one robot. Second, MFC finds a rooted tree cover of this graph in polynomial time, 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. The trees can share vertices and edges. Every vertex is in at least one tree. 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, MSTC runs in polynomial time. The main question is to how determine a suitable rooted tree cover in polynomial time.

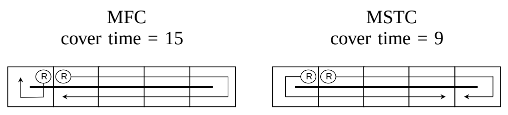

Fig. 8. MFC and MSTC versus STC in Unweighted Terrain

A. Unweighted Terrain

In unweighted terrain, we use the tree-cover algorithm by Even et al. [2]. We define the *weight* of a tree to be the number of its edges. The weight of the rooted tree cover is the largest weight of any of its trees. The problem of finding a weight-minimal rooted tree cover is NP-hard [2]. MFC therefore uses a polynomial-time tree-cover algorithm to find a rooted tree cover with a weight that is at most a factor of four larger than minimal. If there is only one robot, then MFC reduces to STC and thus minimizes the cover (and return) times. If there is more than one robot, then recall that MSTC reduces the cover times of STC by a factor of at least two for three or more robots. MFC cannot make such a strong worst-case guarantee about how small its cover times are with respect to the minimal cover times of a single robot.

Proposition 4: The cover times and cover and return times of MFC in unweighted terrain cannot be larger than the ones of STC.

Proof: The cover times (and cover and return times) of MFC in unweighted terrain cannot be larger than the ones of STC because MFC makes every robot circumnavigate a tree that can be extended to a spanning tree. ■

Figure 8 shows an example of unweighted terrain where the cover time of MFC is almost equal to the cover time of STC given the corridor is sufficiently long, even though the cover time of MSTC is only half the cover time of STC. However, MFC can make a much more powerful guarantee than MSTC, namely a worst-case guarantee about how small its cover times are with respect to the minimal cover times for the number of available robots: Its cover times are only a constant factor larger than minimal.

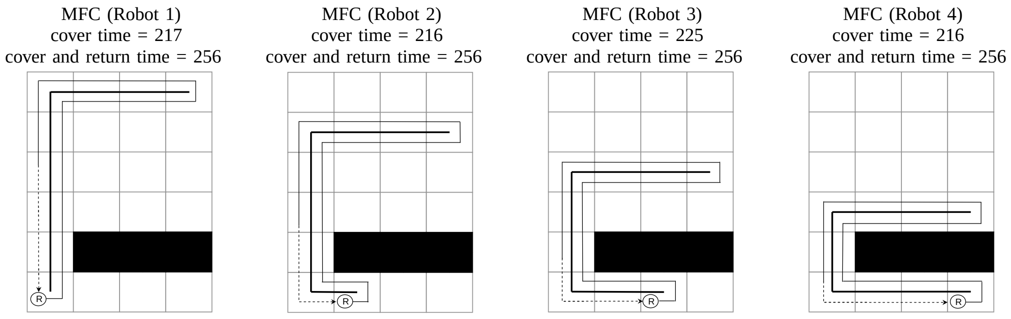

Fig. 7. Example of MFC

Theorem 5: The cover times (and cover and return times) of MFC in unweighted terrain are at most about a factor of eight larger than minimal.

Proof: Let M be the weight of the rooted tree cover found by the tree-cover algorithm of Even et al., N the weight of the weight-minimal rooted tree cover, O the cover time of MFC, P the minimal cover time, and Q the minimal cover time 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 $4M + 4$ small cells, we get that $O \leq 4M + 4$, which is bounded by $16N + 4$ by the approximation guarantee proved in [2]. In turn, $2N \leq Q$ since the weight-minimal rooted tree cover (shifted slightly up and to the left) connects exactly all of the upper left small cells. The factor of two results from the fact that traversing each edge between large cells requires entering two small cells. As $Q \leq P$ trivially, we can combine these results to get $O \leq 4M + 4 \leq 16N + 4 \leq 8Q + 4 \leq 8P + 4$. The proof continues to hold if each occurrence of cover time is replaced with cover and return time. ■

B. Weighted Terrain

The tree-cover algorithm by Even et al. [2] does not apply to weighted terrain. In weighted terrain, we therefore use a new tree-cover algorithm TREE COVER that we describe in Section VII. Each vertex now has a weight equal to the weight of its large cell. We define the weight of a tree to be the sum of the weights of its vertices. The weight of the rooted tree cover is the largest weight of any of its trees. The problem of finding a weight-minimal rooted tree cover remains NP-hard, as we prove in Section VII. MFC therefore uses this polynomial-time tree-cover algorithm to find a rooted tree cover with a weight that is at most a factor of $4(1 + \phi|K|)$ larger than minimal, where $|K|$ 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. We now prove theorems of MFC with TREE COVER in weighted terrain that are similar to those that we proved already for MFC with the tree-cover algorithm by Even et al. [2] in unweighted terrain.

Proposition 6: The cover times of MFC in weighted terrain can be larger than the ones of STC by at most the largest weight of any small cell. The cover and return times of MFC in weighted terrain cannot be larger than the ones of STC.

Proof: The cover times of STC in weighted terrain 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. Thus, the cover times of MFC can be larger than the cover times of STC by at most the largest weight of any small 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. ■

Theorem 7: The cover times (and cover and return times) of MFC in weighted terrain are at most about a factor of $16(1 + \phi|K|)$ larger than minimal, where $|K|$ 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 the weight of a weight-minimal rooted tree cover, O the cover time of MFC, P the minimal cover time, and Q 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.

Because the robots visit all small cells and return to their initial small cells when they circumnavigate their trees, the resulting cover time cannot be larger than the weight of the rooted tree cover. Therefore, $O \leq M$. By Theorem 11 below, TREE COVER finds rooted tree covers with a weight that is at most a factor of $4(1 + \phi|K|)$ larger than minimal, so $M \leq 4(1 + \phi|K|)N$.

The key part of the proof is to bound N. Consider the paths of the robots if they 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. For 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 the largest weight of any small cell. This lets us bound $N/4 \leq Q + w_{max}/4$.

Finally, $Q \leq P$ holds trivially, and we can combine these bounds to get $O \leq M \leq 4(1+\phi|K|)N \leq 16(1+\phi|K|)Q + 4(1+\phi|K|)w_{max} \leq 16(1+\phi|K|)P + 4(1+\phi|K|)w_{max}$. The proof continues to hold if each occurrence of "cover time" is replaced with "cover and return time". ■

The ratio ϕ 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 2. Then, $16(1+\phi|K|) \approx 16$ for a small number of robots $|K|$. Thus, the cover times (and cover and return times) of MFC are at most about sixteen times larger than minimal.

For illustration, 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.

VII. Weight-Minimal Rooted Tree Covers

We now modify the tree-cover algorithm by Even et al. [2] (and the proofs in that paper) to work on graphs with weighted vertices rather than weighted edges. We state the tree-cover algorithm (called TREE COVER), prove its properties and describe how MFC uses it.

A. Tree Cover Problem

We define the Weight-Minimal Rooted Tree Cover problem as follows: 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. 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 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. Complexity

We show that the Weight-Minimal Rooted Tree Cover problem is NP-hard, which provides our motivation for designing polynomial-time approximation algorithms.

Theorem 8: It is NP-hard to find weight-minimal K-rooted tree covers for graphs G.

Proof: To prove the NP-hardness, we reduce from the Bin-packing problem, which is defined as follows: Given a set of elements with given 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?

Given an instance of it, we construct a yes/no version of the Weight-Minimal Rooted Tree Cover problem, namely determining whether a given graph G has a K-rooted tree cover whose weight is at most a given constant. We give our reduction 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. This completes the construction, which can be done in polynomial time.

If the weight of a K-rooted tree cover is at most the given capacity plus one, then placing each element in one of its "root bins" will meet the capacity constraints. 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 make 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. Tree Cover Algorithm

TREE COVER 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, the 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. 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 at least $2B$. 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 $w_r \leq w_{max} \leq B$ 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 $2B$ 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 first to such a tree rooted in a child of r 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. For notational convenience, we write $\epsilon := \phi|K|$.

Theorem 9: 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 $4B$. ■

Theorem 10: If TREE COVER reports FAILURE, then there is no K-rooted tree cover of graph G with weight at most $B/(1+\epsilon)$.

Proof: Assume that a weight-minimal K-rooted tree cover T of graph G has weight $B' \leq B/(1+\epsilon)$. 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 at distance at most B from subtree $l \in L$. We show that $|\bigcup_{l \in L'} K(l)| \geq |L'|$ 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.

Consider any $L' \subseteq L$. Let $T' \subseteq T$ be the set of trees in the optimum solution 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'.

We first lower bound $|\bigcup_{l \in L'} K(l)| \geq |T'|$. For every tree $t \in T'$, there exists at least one non-leftover subtree l in L' that has at least one vertex in common with t. l and the root of t are at distance at most $B' \leq B/(1+\epsilon) \leq B$. Therefore, l can be matched to the root of t. Overall, the set $\bigcup_{l \in L'} K(l)$ contains the roots of all trees in T', implying that $|\bigcup_{l \in L'} K(l)| \geq |T'|$.

Next, we want to show that $|T'| \geq |L'|$, which we do by relating the weights of the corresponding sets. Because the weights of all non-leftover subtrees are in the interval $[B, 2B)$, and the weights of all trees in T' are at most B', we obtain that $w(L') \geq B|L'|$ and $w(T') \leq B'|T'|$.

To relate $w(L')$ and $w(T')$, we observe that every vertex in non-leftover subtrees in L' is also in at least one tree in T'. The non-leftover subtrees in L' can contain at most $|L'|$ duplicate vertices. The reason is that every non-leftover subtree created by Step 2 of TREE COVER 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. 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. Since each of these duplicate vertices has weight at most w_{max}, we obtain the bound $w(L') \leq w(T') + w_{max}|L'|$.

Finally, to bound w_{max}, we observe that at best, the sum of the weights of all vertices could be split evenly among the trees; hence, $B' \geq w_{sum}/|K|$, which implies that $w_{max} = w_{sum}\phi \leq |K|B'\phi$. Combining all these bounds, we get

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

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

F. Application

Theorem 11: There is an algorithm finding a K-rooted tree cover of graph G with weight at most a factor of $4(1+\epsilon)$ larger than minimal.

Proof: 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 less than $(1+\epsilon)$. Let b be the weight of a weight-minimal K-rooted tree cover T of graph G. Because the weight of each unblocked large cell is a positive integer, $b \geq 1$. Let B_l be the lower bound and B_u be the upper bound of the binary search after termination. Let $\hat{B} = \lceil B_l/(1+\epsilon) \rceil$. We consider the following two cases.

1) TREE COVER reports SUCCESS with B set to $(1+\epsilon)\hat{B}$. If $B_l = w_{max}$ (the initial lower bound), then the fact that $b > w_{max}/(1+\epsilon)$ implies that $b > B_l/(1+\epsilon)$. Otherwise, the definition of binary search implies that TREE COVER with B set to B_l reports FAILURE, and Theorem 10 implies that $b > B_l/(1+\epsilon)$.
   Because b is an integer, we in fact get the stronger bound $b \geq \lceil B_l/(1+\epsilon) \rceil = \hat{B}$. The weight of the K-rooted tree cover returned by TREE COVER with B set to $(1+\epsilon)\hat{B}$ is at most $4(1+\epsilon)\hat{B}$ (according to Theorem 9), which in turn is bounded by $4(1+\epsilon)b$. That is, the weight is at most a factor of $4(1+\epsilon)$ larger than minimal.
2) TREE COVER reports FAILURE with B set to $(1+\epsilon)\hat{B}$. Theorem 10 thus implies that $b > (1+\epsilon)B_1/(1+\epsilon) = \hat{B}$, and since b is an integer, $b \geq \hat{B} + 1 =: B'$. Since $(B_u - B_l)/(1+\epsilon) < 1$ according to the termination condition of the binary search, $B' \geq B_l/(1+\epsilon) + 1 > B_u/(1+\epsilon)$. Thus, $(1+\epsilon)B' > B_u$, and TREE COVER reports SUCCESS with B set to $(1+\epsilon)B'$. The weight of the K-rooted tree cover we have with B set to $(1+\epsilon)B'$ is at most $4(1+\epsilon)B' \leq 4(1+\epsilon)b$, according to Theorem 9. This 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 $\lceil \log_2((w_{sum} - w_{max})/(1+\epsilon)) \rceil \leq \log_2 w_{sum} + 1$ times, which is polynomial in the size of the input.

VIII. Experimental Results

We compare MFC and (the backtracking version of) 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 unblocked large cell in weighted terrain is chosen from the weights 8, 16, 24, $\ldots$, 80. Figure 9 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 are averaged over at least 50 runs with randomly generated terrain (if applicable) and randomly generated initial large cells. All cover (and return) times are rounded to the nearest integer.

We report our experimental results for unweighted terrain in Figure 10 and for weighted terrain in Figure 11. The tables report for each scenario a lower bound that represents the idealized cover (and return) time [ideal max]: It simply divides the sum of the weights of all large cells by the number of robots (and, for unweighted terrain, subtracts one). The ideal cover (and return) times 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 times are equal to the cover (and return) times, 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 ratios of the actual travel times and the ideal cover (and return) times [ratio], giving the upper bounds on how much the actual cover (and return) times are 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 ratios of the cover (and return) times and the ideal cover (and return) times increase 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 ratios increase very slowly with the number of robots for MFC, but much faster for MSTC, implying that the cover (and return) times of MFC remain close to minimal for large numbers of robots. The ratios change insignificantly with the amount of clustering for MFC, but a lot for MSTC, implying that the 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 ratios change insignificantly for MFC if the team objective is changed from “Cover” to “Cover and Return”, but increase by about a factor of two for non-optimized MSTC because the robot with the largest travel time has to backtrack along most of its robot path. This implies that all robots are close to their initial small cells when coverage is complete for MFC, which facilitates their retrieval. The cover times of

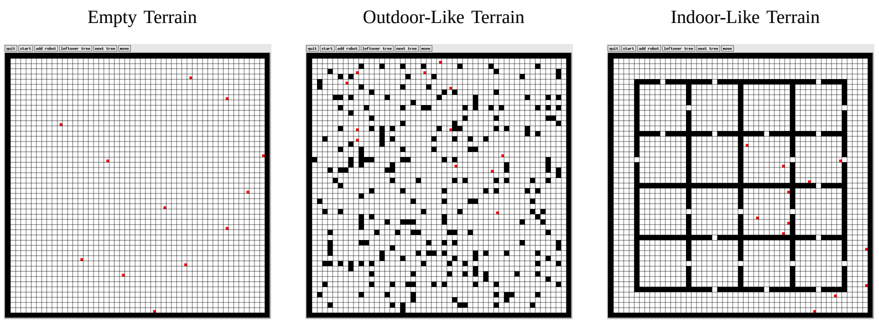

Fig. 9. Screenshots of Different Kinds of Terrain

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	4801	4878	(4731)	1.02	4877	(4730)	1.02	10538	(8666)	2.19	5269	(5048)	1.10	5337	(4410)	1.11	5269	(4346)	1.10
	2	60	4801	4886	(4720)	1.02	4885	(4719)	1.02	10889	(8315)	2.27	5445	(5095)	1.13	5513	(4241)	1.15	5445	(4180)	1.13
	2	none	4801	4888	(4725)	1.02	4886	(4723)	1.02	11057	(8147)	2.30	5529	(5161)	1.15	5602	(4168)	1.17	5529	(4107)	1.15
	8	30	1200	1399	(838)	1.17	1396	(837)	1.16	7499	(73)	6.25	3752	(38)	3.13	3817	(45)	3.18	3751	(38)	3.13
	8	60	1200	1415	(904)	1.18	1414	(902)	1.18	6923	(154)	5.77	3462	(77)	2.89	3539	(93)	2.95	3462	(77)	2.89
	8	none	1200	1394	(956)	1.16	1391	(953)	1.16	6411	(248)	5.34	3210	(127)	2.68	3281	(146)	2.73	3206	(124)	2.67
	14	30	685	841	(431)	1.23	836	(431)	1.22	7369	(5)	10.76	3685	(2)	5.38	3756	(5)	5.48	3685	(2)	5.38
	14	60	685	819	(522)	1.20	815	(522)	1.19	6774	(17)	9.89	3387	(8)	4.94	3461	(16)	5.05	3387	(8)	4.94
	14	none	685	830	(513)	1.21	824	(511)	1.20	6005	(49)	8.77	3002	(25)	4.38	3072	(40)	4.48	3002	(25)	4.38
	20	30	479	615	(307)	1.28	609	(307)	1.27	7224	(3)	15.08	3612	(1)	7.54	3685	(3)	7.69	3612	(1)	7.54
	20	60	479	604	(332)	1.26	599	(332)	1.25	6728	(9)	14.05	3364	(4)	7.02	3439	(9)	7.18	3364	(4)	7.02
	20	none	479	604	(321)	1.26	599	(319)	1.25	5591	(18)	11.67	2796	(9)	5.84	2867	(18)	5.99	2796	(9)	5.84
Outdoor	2	30	4321	4380	(4269)	1.01	4379	(4268)	1.01	9391	(7893)	2.17	4695	(4574)	1.09	4772	(4031)	1.10	4695	(3960)	1.09
	2	60	4321	4382	(4266)	1.01	4381	(4265)	1.01	9556	(7728)	2.21	4778	(4627)	1.11	4854	(3957)	1.12	4778	(3890)	1.11
	2	none	4321	4377	(4269)	1.01	4376	(4268)	1.01	9683	(7601)	2.24	4842	(4525)	1.12	4923	(3903)	1.14	4842	(3931)	1.12
	8	30	1079	1263	(789)	1.17	1260	(788)	1.17	6985	(36)	6.47	3500	(18)	3.24	3561	(26)	3.30	3494	(18)	3.24
	8	60	1079	1278	(790)	1.18	1274	(789)	1.18	6314	(113)	5.85	3158	(59)	2.93	3229	(70)	2.99	3157	(58)	2.93
	8	none	1079	1247	(873)	1.16	1243	(871)	1.15	6032	(151)	5.59	3016	(76)	2.80	3099	(94)	2.87	3016	(76)	2.80
	14	30	616	764	(450)	1.24	760	(451)	1.23	6759	(6)	10.97	3392	(3)	5.51	3452	(6)	5.60	3380	(3)	5.49
	14	60	616	750	(482)	1.22	745	(481)	1.21	6311	(27)	10.25	3156	(13)	5.12	3228	(20)	5.24	3156	(13)	5.12
	14	none	616	746	(464)	1.21	741	(463)	1.20	5497	(52)	8.92	2748	(26)	4.46	2819	(37)	4.58	2748	(26)	4.46
	20	30	431	572	(280)	1.33	567	(281)	1.32	6723	(3)	15.60	3362	(2)	7.80	3437	(3)	7.97	3362	(2)	7.80
	20	60	431	557	(285)	1.29	552	(285)	1.28	6131	(10)	14.23	3066	(5)	7.11	3140	(9)	7.29	3065	(5)	7.11
	20	none	431	551	(296)	1.28	547	(294)	1.27	5348	(23)	12.40	2674	(12)	6.20	2740	(18)	6.36	2674	(12)	6.20
Indoor	2	30	4090	4172	(4017)	1.02	4171	(4015)	1.02	8937	(7422)	2.19	4468	(4230)	1.09	4539	(3797)	1.11	4468	(3729)	1.09
	2	60	4090	4196	(3995)	1.03	4194	(3994)	1.03	9243	(7116)	2.26	4621	(4290)	1.13	4690	(3648)	1.15	4621	(3585)	1.13
	2	none	4090	4172	(4015)	1.02	4171	(4014)	1.02	9326	(7033)	2.28	4663	(4166)	1.14	4739	(3615)	1.16	4663	(3549)	1.14
	8	30	1022	1232	(849)	1.21	1225	(849)	1.20	6501	(24)	6.36	3262	(12)	3.19	3319	(17)	3.25	3253	(12)	3.18
	8	60	1022	1209	(846)	1.18	1202	(846)	1.18	6081	(86)	5.95	3042	(44)	2.98	3114	(55)	3.05	3041	(43)	2.98
	8	none	1022	1209	(842)	1.18	1199	(839)	1.17	5815	(180)	5.69	2905	(90)	2.84	2981	(108)	2.92	2907	(90)	2.84
	14	30	584	775	(438)	1.33	768	(439)	1.32	6348	(4)	10.86	3192	(2)	5.47	3254	(4)	5.57	3190	(2)	5.46
	14	60	584	748	(452)	1.28	741	(452)	1.27	5995	(22)	10.27	2999	(11)	5.14	3071	(16)	5.26	2998	(11)	5.13
	14	none	584	732	(448)	1.25	725	(445)	1.24	5033	(46)	8.62	2517	(23)	4.31	2594	(31)	4.44	2517	(23)	4.31
	20	30	408	617	(241)	1.51	608	(242)	1.49	6370	(3)	15.61	3188	(1)	7.81	3248	(3)	7.96	3186	(1)	7.81
	20	60	408	570	(270)	1.40	566	(271)	1.39	5732	(10)	14.05	2866	(5)	7.02	2939	(8)	7.20	2866	(5)	7.02
	20	none	408	547	(279)	1.34	540	(277)	1.32	4696	(22)	11.51	2348	(11)	5.75	2420	(17)	5.93	2348	(11)	5.75

Fig. 10. Experimental Results for MFC and MSTC in Unweighted Terrain (“Max” = Cover Time or Cover and Return Time)

optimized MSTC are similar to the ones of non-optimized MSTC but its cover and return times are significantly smaller. Consequently, the ratios are reduced by a factor of two and then no longer differs significantly from the ratios for “cover.” However, even without such optimizations, MFC continues to have much smaller cover times than optimized MSTC, for both team objectives in all scenarios. MFC has much smaller cover (and return) times than MSTC for more than two robots and comparable cover (and return) times than optimized MSTC for two robots because MFC takes the team objective already into account when finding a tree for each robot to circumnavigate, whereas MSTC takes the team objective only into account when it decides how the robots should circumnavigate the single tree. Consequently, MSTC does not balance the travel times of the robots as well, as evidenced by a large difference between the minimal and largest travel times of the robots. Theorem 5 guarantees that the cover (and return) times of MFC in unweighted terrain are at most a factor of eight larger than minimal but empirically the ratios are significantly smaller (at most 1.51) in all tested scenarios. Similarly, Theorem 7 guarantees that the cover (and return) times of MFC in weighted terrain 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 ratios are significantly smaller (at most 1.77) in all tested scenarios.

IX. Conclusions

In this article, we introduced a new multi-robot coverage algorithm, called Multi-Robot Forest Coverage (MFC). We extended MFC and an alternative existing multi-robot coverage algorithm, called Multi-Robot Spanning Tree Coverage (MSTC), from unweighted terrain to weighted terrain. Our experimental results showed that the cover time of MFC is smaller than the one of Multi-Robot Spanning-Tree Coverage (MSTC) and close to minimal in all tested scenarios. Currently, MFC assumes ideal robots. It is future work to generalize it to robots with actuator and sensor uncertainty and other typical imperfections, which includes making it robust in the presence of failing robots. It is also future work to combine the ideas behind MSTC and MFC, especially if several robots start in nearby small cells.

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. 11. Experimental Results for MFC and MSTC in Weighted Terrain (“Max” = Cover Time or Cover and Return Time)

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] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[5] 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.

[6] J. Svennebring and S. Koenig. Building terrain-covering ant robots. *Autonomous Robots*, 16(3):313–332, 2003.

[7] 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.

[8] 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.

[9] X. Zheng and S. koenig. Robot coverage of terrain with non-uniform traversability. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2300–2309, 2007.

ACKNOWLEDGMENT

Parts of this article have been presented at the IEEE International Conference on Intelligent Robots and Systems in 2005 [8] and 2007 [9]. We thank Gal Kaminka for interesting discussions about Multi-Robot Spanning Tree Coverage and multi-robot coverage in general. The research was 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.

PLACE PHOTO HERE

Xiaoming Zheng

PLACE PHOTO HERE

Sven Koenig Sven Koenig is an associate professor in computer science at the University of Southern California. Most of his research centers around techniques for decision making (planning and learning) that enable single situated agents (such as robots or decision-support systems) and teams of agents to act intelligently in their environments and exhibit goal-directed behavior in real-time, even if they have only incomplete knowledge of their environment, imperfect abilities to manipulate it, limited or noisy perception or insufficient reasoning speed. He is the recipient of an NSF CAREER award, an IBM Faculty Partnership Award, a Charles Lee Powell Foundation Award, a Raytheon Faculty Fellowship Award, an ACM Recognition of Service Award and a Fulbright Fellowship, among others. He co-founded Robotics: Science and Systems, a general robotics conference.

PLACE PHOTO HERE

David Kempe

PLACE
PHOTO
HERE

Sonal Jain