Multi-Robot Forest Coverage*

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

***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 paper, we describe Multi-Robot Forest Coverage, a new multi-robot coverage algorithm based on an algorithm by Even et al. for finding a tree cover with trees of balanced weights. The cover time of Multi-Robot Forest Coverage is at most eight times larger than optimal, and our experiments show it to perform significantly better than existing multi-robot coverage algorithms.**

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

I. Introduction

One of the main applications of mobile robots is terrain coverage: visiting each location in known terrain to perform a task. Terrain coverage is crucial for tasks ranging from mundane lawn mowing, cleaning or harvesting to search-and-rescue missions, intrusion detection or mine clearing. It is frequently desirable to minimize the time by which coverage is completed.

The single-robot coverage problem is solved essentially optimally 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 trajectory for each robot so that the cover time (that is, largest travel cost of any robot) is minimized. As we show in this paper, this problem is NP-complete. It thus becomes necessary to consider heuristics for solving it. Recently, STC was generalized to Multi-Robot Spanning Tree Coverage (MSTC), a polynomial-time multi-robot coverage heuristic [5]. While MSTC provably improves the cover time of STC, it cannot guarantee its cover time to be close to optimal. In this paper, we describe Multi-Robot Forest Coverage (MFC), a polynomial-time multi-robot coverage heuristic based on an algorithm for finding a tree cover with trees of balanced weights, one for each robot [2].

*We thank Gal Kaminka for interesting discussions about multi-robot coverage and for making his paper available on the web. 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 author(s) and do not necessarily reflect the views of the National Science Foundation.

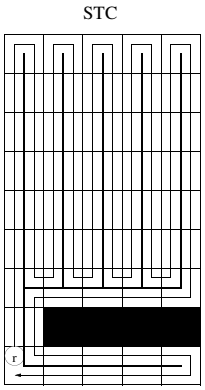

Fig. 1. Example of STC

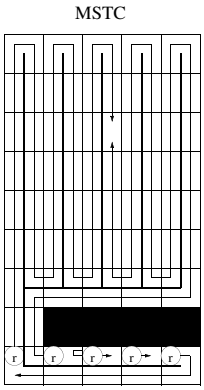

Fig. 2. Example of MSTC

Our analytical results prove the cover time of MFC to be at most eight times larger than optimal, and our experiments show it to be significantly better than the worst-case bound, and also superior to that 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. Assumptions

The terrain to be covered is discretized into large square cells, each of which is either entirely blocked or entirely unblocked, and contains four small square cells. The robots are of the same size as a small cell and also identical otherwise. We assume that the robots always know their current small cell, and can move between any two unblocked (horizontally or vertically) adjacent small cells without error. Unless specified otherwise, we assume that such moves takes unit time. For ease of exposition, we assume that several robots are able to occupy the same small cell simultaneously, and never block each other; a brief discussion in Section VI shows that this assumption is not truly essential.

Note that one could potentially use specialized region-based coverage algorithms on unit-cost grid graphs instead of the graph-based algorithms discussed in this paper. However, even though (some of) our theorems are restricted to unit-cost grid graphs, all of the coverage algorithms discussed in this paper work on general graphs with positive edge costs.

III. Single-Robot Coverage

Spanning Tree Coverage (STC) solves the single-robot coverage problem in polynomial time [3]. It first computes

a spanning tree of the graph whose vertices are the large cells, and whose edges connect adjacent unblocked large cells. The robot then circumnavigates the spanning tree. STC never visits any small cell twice and thus minimizes the cover time. In addition, the robot essentially returns to its initial small cell, facilitating its collection and storage. Figure 1 shows an example of STC in operation, including the large cells (squares), spanning tree (thick lines), robot (circle), and its trajectory (arrow).

IV. Complexity of Multi-Robot Coverage

While the single-robot coverage problem can be solved in polynomial time, the problem becomes significantly more complex when we try to minimize the cover time using multiple robots. Theorem 1 shows that two natural variants of the multi-robot coverage problem are NP-complete. We thus do not expect to be able to solve it exactly in polynomial time, and it becomes necessary to consider heuristics for solving it.

Theorem 1: It is NP-complete to determine whether the following two multi-robot coverage problems can be solved with cover times that are smaller than a given value:

1) multi-robot coverage problems with two robots, where the costs of moving from one small cell to an adjacent one can be non-uniform (and large); and
2) multi-robot coverage problems with k robots, where k is part of the input, the costs of moving from one small cell to an adjacent one are uniform, and all robots must return to their initial small cells.

Proof. Clearly, both versions of the multi-robot coverage problem are in NP since one can easily guess the trajectories of the robots and then verify their costs in polynomial time. To prove their NP-hardness, we reduce from partitioning problems, in which one seeks to partition a set of n integers into k sets of equal sums. For $k = 2$, the problem is NP-hard if the integers can be exponential in n, and is known as the PARTITION problem [4]. If k is part of the input, then the problem is NP-hard even if the integers have sizes that are only polynomial in n.

We first give the reduction for the second version. The 3-PARTITION problem, known to be strongly NP-complete [4], 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?

Given an instance of the 3-PARTITION problem, we construct a multi-robot coverage problem with n robots as follows: We start with a "corridor" consisting of $3n$ vertically adjacent large cells, numbered from 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. All n robots start in the lower left small cell of the first corridor cell. This completes the construction, which can be done in polynomial time.

We claim that the smallest cover time is at most $B \cdot 24n + 12n$ 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, then we let the j^{th} robot cover the i^{th} tunnel for each $i \in S_j$, ending in its initial small cell. It thus traverses the tunnels for a cost of at most $\sum_{i \in S_j} 4 \cdot a_i \cdot 6n = B \cdot 24n$, and the corridor for a cost of at most $12n$. The total cost thus is at most $B \cdot 24n + 12n$, 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 cost 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 cost of any robot is at most $B \cdot 24n + 12n$, which implies that $\sum_{i \in S_j} a_i \leq B + \frac{1}{2}$. 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.

This reduction has to be slightly adapted to prove the NP-hardness of the first version. When reducing from the PARTITION problem, the given integers a_i can be exponential in n, so 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, but make the cost of entering and leaving the large cell from the corridor $a_i \cdot 10n$. Once we use non-uniform costs for moving between large cells, we can also avoid the requirement that the robots return to their initial small cells, by adding two more large "destination cells" which have very high cost ($36n^2 \cdot \sum_i a_i$) to enter and leave. Then, both robots end in destination cells if their cover time is small, and thus have entered and left each of their tunnel cells. A very similar proof to the one above then shows that the cover time is bounded by a certain constant if and only if the given integers can be partitioned evenly into two sets. ■

Currently, we cannot prove that the second version of the multi-robot coverage problem is NP-hard without the requirement that the robots return to their initial small cells. It is also open whether the second version is NP-hard for fixed k, although we conjecture it to be.

V. Existing Multi-Robot Coverage Algorithms

While single-robot coverage algorithms have received a lot of attention, there are currently many fewer algorithms for the multi-robot coverage problem. An overview is given in [1]. Many of the multi-robot coverage algorithms are for robots that interact and plan only locally [9], often called ant robots [8]. Naturally, global planning can lead to significantly smaller cover times since it allows the robots to coordinate their trajectories much better.

Recently, STC was generalized to Multi-Robot Spanning Tree Coverage (MSTC), a polynomial-time multi-robot cov-

erage heuristic [5]. MSTC first computes the same spanning tree as STC, and considers the tour that circumnavigates the spanning tree. Each robot follows the tour segment clockwise ahead of it, with one exception: To improve the cover time, 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 time of STC by a factor of at least 2 (or 3/2) for $k \geq 3$ robots (or two robots, respectively). Each small cell is visited by only one robot, so there are never any collisions or blockages. Figure 2 shows an example of MSTC in operation.

While the improvement in cover time of MSTC over STC is significant for two or three robots, it does not necessarily increase further as the number of robots grows. Indeed, Figure 2 gives a bad example for MSTC, showing that the factor remains two even when a much larger speedup is possible. This 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 costs 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 are balanced.

VI. A New Multi-Robot Coverage Algorithm

We now describe Multi-Robot Forest Coverage (MFC), a new polynomial-time multi-robot coverage heuristic. It is based on an algorithm by Even et al. [2] that gives a four-approximation for the problem of finding a tree cover with given roots, minimizing the weight of the heaviest tree.

A. Algorithm

MFC operates on the graph whose vertices are the large cells, and whose edges connect adjacent unblocked large cells. If r robots start in a large cell, then MFC makes r identical copies of that vertex. MFC first finds a rooted tree cover for this graph in polynomial time, where the roots are the vertices that contain robots. (The graph is allowed to be disconnected, so long as each of its components contains at least one robot.) Each robot then circumnavigates its tree.

We now explain what a rooted tree cover is. Let $G = (V, E)$ be a graph and $R \subseteq V$ be a set of roots. An R-rooted tree cover of G is a forest of $|R|$ trees that cover V. The trees can share vertices and edges, but their roots have to be distinct vertices from R. The weight of a rooted tree cover is the largest weight of any of its trees. The (min-max) rooted tree cover problem then is to find a weight-minimal rooted tree cover for a given graph and given roots. This problem is NP-complete, which can be proved by reducing the NP-hard bin-packing problem to it [2]. However, TREE COVER is a polynomial-time algorithm by Even et al. that finds a rooted tree cover whose weight is at most four times larger than optimal [2]. It performs a binary search to determine the smallest value B such that it can find a rooted tree cover of weight at most $4B$. If TREE COVER does not succeed in finding such a tree cover for the given B, then there is no rooted tree cover of weight at most B. As a result, TREE COVER gives a four-approximation for the rooted tree-cover problem. It operates as follows:

1) Remove all edges with edge costs larger than B.
2) Contract all roots into a single vertex, find a minimum spanning tree for the resulting graph, and then uncontract the single vertex again, splitting the spanning tree into $|R|$ trees.
3) Decompose each tree into subtrees that can share vertices but no edges. The weight of each subtree is in the range $[B, 2B)$, with the possible exception of a leftover subtree that contains the root of the tree, and whose weight is less than B. (See [2] for details on this step.)
4) 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 leftover tree of the root (or root itself) are at distance at most B. If some non-leftover subtrees cannot be matched, this is proof that no rooted tree cover of weight at most B exists.
5) For each root, return a tree consisting of the root, the leftover subtree of the root (if any) of weight at most B, the single non-leftover subtree matched to the root (if any) of weight at most $2B$, and a cost-minimal path of weight at most B from the non-leftover subtree to the leftover subtree (or root). The weight of each tree is at most $4B$, resulting in a rooted tree cover of weight at most $4B$.

We enhance TREE COVER in two ways. While these improvements do not affect its worst-case guarantee, they can potentially reduce the weight of the returned rooted tree cover:

1) The smallest value of B for which TREE COVER finds a rooted tree cover may not be the value of B resulting in the rooted tree cover of the smallest weight. Thus, the improved version of TREE COVER stores all rooted tree covers that are computed during the binary search, and returns the best one rather than the last one.
2) When TREE COVER computes the maximum matching in Step 4, it does not take the weights of the resulting trees into account. In the improved version of TREE COVER, a non-leftover subtree can therefore be matched to a root only if the non-leftover subtree and leftover tree of the root (or root itself) are at distance at most B, and the weight of the resulting tree is at most B'. The improved version of TREE COVER then searches for the smallest value B' for which such a matching can be found.

Figure 3 shows an example of MFC in operation. The left figure shows the initial spanning tree for $|R| = 5$ after it was split into one tree for each robot. The other figures show the non-leftover subtrees (solid thick lines), the cost-minimal paths from the non-leftover subtrees to the roots (dashed thick lines) and the trajectories of the five robots (arrows). In this case, there are no leftover subtrees. MFC sends all robots through the narrow passage and thus utilizes them to cover

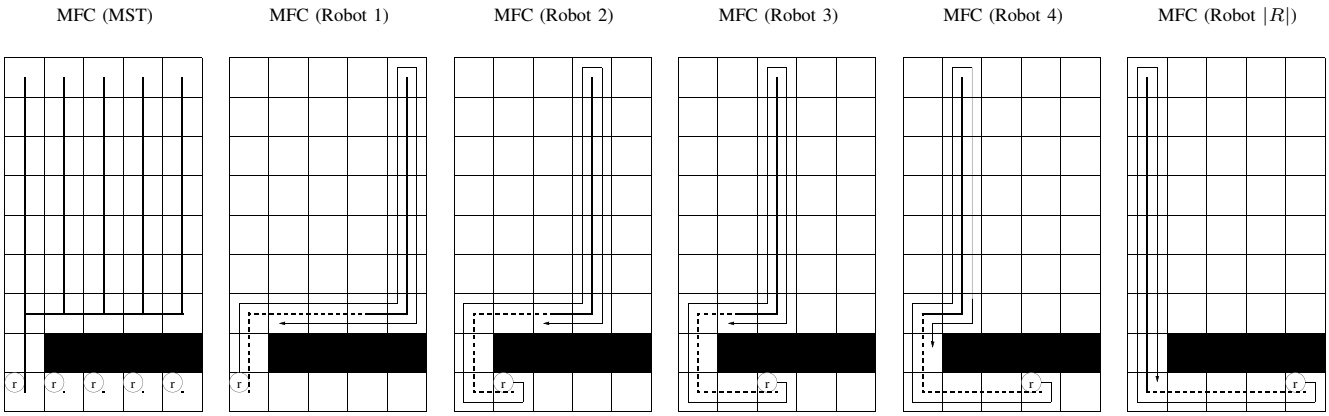

Fig. 3. Example of MFC

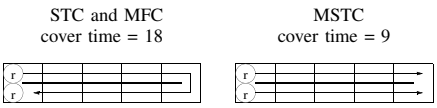

Fig. 4. MFC and MSTC versus STC

the terrain on the other side, resulting in balanced travel costs of the robots and a small cover time of 45. Figure 2 showed already that MSTC sends only two robots through the narrow passage, resulting in unbalanced travel costs of the robots and a cover time of 73.

B. Properties

If there is only one robot, MFC reduces to STC and thus minimizes the cover time. If there is more than one robot, recall that MSTC reduces the cover time of STC by a factor of at least 2 (or 3/2) for $k \geq 3$ robots (or two robots, respectively). MFC cannot make such a strong worst-case guarantee about how good its cover time is with respect to the smallest cover time of a single robot.

Theorem 2: The cover time of MFC can be equal to the cover time of STC, but cannot be worse than it.

Proof. The cover time of MFC cannot be worse than that of STC because MFC makes every robot circumnavigate a tree that can be extended to a spanning tree. On the other hand, Figure 4 shows an example where the cover time of MFC is equal to the cover time of STC (where, in the case of STC, the second robot does not move), no matter how long the corridor is, even though the cover time of MSTC is only half the cover time of STC. ■

However, MFC can make a much more powerful guarantee, namely a worst-case guarantee about how good its cover time is with respect to the smallest cover time for the number of available robots: it is only a constant factor larger than optimal.

Theorem 3: The cover time of MFC is at most a factor of eight larger than optimal (plus a small constant).

Proof. Let C be the weight of the rooted tree cover found by TREE COVER, $\hat{C}$ the weight of the weight-minimal rooted tree cover, T the cover time of MFC, $\hat{T}$ the smallest cover time, and $\hat{T}_{\text{ul}}$ the smallest cover time if the robots only need to cover the upper left small cells of all unblocked large cells. First, because circumnavigating a tree of weight C requires entering $4C + 4$ small cells, we have $T \leq 4C + 4$. Second, by the approximation guarantee proved in [2], the tree cover found by TREE COVER is at most four times larger than optimal, so $C \leq 4\hat{C}$. Third, because the weight-minimal rooted tree cover (shifted slightly up and to the left) connects exactly all of the upper left small cells, it provides a lower bound on the smallest cover time if the robots only need to cover the upper left small cells, so $2\hat{C} \leq \hat{T}_{\text{ul}}$. The factor of two results from the fact that traversing each edge between large cells requires entering two small cells. Finally, because only a subset of the small cells need to be covered if the robots only need to cover the upper left small cells, we have $\hat{T}_{\text{ul}} \leq \hat{T}$. Putting all of these inequalities together, we obtain $T \leq 4C + 4 \leq 16\hat{C} + 4 \leq 8\hat{T}_{\text{ul}} + 4 \leq 8\hat{T} + 4$. ■

MSTC cannot claim that its cover time is only a constant factor larger than optimal. Consider again Figure 2, but this time for an arbitrary number of robots $|R|$ in a terrain of size $|R| + 4$ by $|R|$ large cells instead of nine by five large cells. The unblocked terrain of size $|R|+2$ by $|R|$ large cells above the wall contains $4|R|^2 + 8|R|$ unblocked small cells. MSTC covers this terrain with only two robots, and its cover time thus is at least $2|R|^2 + 4|R|$. On the other hand, the smallest cover time is no larger than the travel cost needed for each robot to completely circumnavigate its tree shown in Figure 3 (left to extreme right), which is $8|R| + 12$. Thus, the cover time of MSTC is at least a factor of $(2|R|^2+4|R|)/(8|R|+12)$ larger than optimal, and this factor grows unboundedly as $|R|$ increases. MFC also has disadvantages. For example, even if the robots start in different small cells, it is possible for several robots to occupy the same small cell at the same time. Thus, some robots might have to wait for other robots to leave their cell if our assumption that several robots are able to occupy the same small cell simultaneously is unjustified.

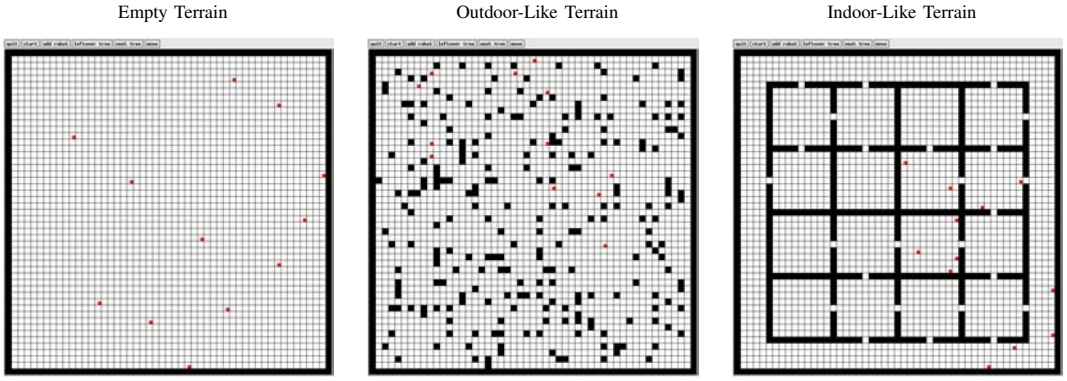

Fig. 5. Screenshots of Different Kinds of Terrain

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. 6. Experimental Results for MFC and MSTC ("Max" = Cover Time)

VII. Experimental Results

We now compare the cover times of MFC and MSTC experimentally. We implemented the backtracking version of MSTC as described in [5]. We evaluate them on two different tasks, namely coverage ["cover"], and coverage with the additional requirement that all robots return to their initial small cells after coverage ["cover and return"]. Both MFC and MSTC can easily be extended to the second task. For MFC, each robot simply circumnavigates its tree until it reaches its initial small cell. For MSTC, each robot that has covered its cells backtracks until it reaches its initial small cell. Thus, in both cases, the robots continue to move around the tree(s), extending the original multi-robot coverage algorithms in a very simple way.

We evaluate MFC and MSTC for both tasks 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. Figure 5 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. A clustering percentage parameter x determines how strongly their initial small cells 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. (We ensure that no two robots will be placed in the same small cell.) Thus, a small value of x results in a high clustering of initial small cells, while $x = 200$ is equivalent to no clustering at all [none]. For each scenario, we report data that has been averaged over 100 runs with randomly generated

terrain (if applicable) and randomly generated initial small cells. All cover times have been rounded to the nearest integer.

Table 6 reports for each scenario a lower bound that represents an idealized cover time [ideal max]: it simply divides the number of unblocked small cells by the number of robots, and subtracts one, since the initial small cells of the robots are automatically covered. The ideal cover time would be the cover time if no robot needed to pass through already covered small cells to reach other small cells that it needs to cover. The table also reports the smallest [min] and largest [max] travel cost of any robot for each combination of a multi-robot coverage algorithm, scenario and task. The largest travel cost is the cover time, and the difference between the smallest and largest travel costs gives an indication of how balanced the travel costs of the robots are. In addition, the table also reports the ratio of the actual cover time and the ideal cover time [ratio], giving an upper bound on how far the actual cover time is away from optimum. The ratio is indeed only an upper bound, since the ideal cover time may not be achievable. For instance, several cells must be visited by multiple robots in the example of Figure 3.

We make the following observations: The ratio of the cover time and the ideal cover time increases with the number of robots for both MFC and MSTC since the overhead (defined as the number of already covered 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 time of MFC remains close to optimal 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 time of MSTC remains small if robots start in nearby cells – a common situation since robots are often deployed or stored together. The ratio changes insignificantly for MFC if the task is changed from "cover" to "cover and return", but increases by about a factor of two for MSTC (because the robot with the largest travel cost 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 collection or storage.

Overall, the ratio is small for MFC (at most 1.51) in all tested scenarios, and in fact significantly smaller than the factor of eight guaranteed by Theorem 3. The ratio is much larger for MSTC (7.81 for "cover" and 15.61 for "cover and return"). The reason is that MSTC does not balance the travel costs of the robots as well, as evidenced by a large difference between the smallest and largest travel costs of the robots. When interpreting these results, however, one needs to keep in mind that the cover times of both MFC and MSTC depend on the initial spanning trees. Among the (large) number of spanning trees for a given unit-cost grid graph, some may yield significantly better cover times but we have not yet experimented with different ways of constructing the initial spanning trees.

We now discuss one important optimization. One can reduce the cover times of both MFC and MSTC by moving robots on cost-minimal paths to their initial small cells rather than along the tree(s). This applies to "cover and return" when the robots return to their initial small cells. For MSTC, it also applies to "cover", when the robots backtrack to their initial small cells during coverage. We refer to a version of MSTC with these improvements as optimized MSTC. We observe that the improvements make almost no difference for "cover" but a large difference for "cover and return," where the ratio is reduced by a factor of two and then no longer differs significantly from the ratio for "cover." However, even without such optimizations, MFC continues to have much smaller cover times than optimized MSTC, for both tasks in all scenarios. The reason is that MFC takes the objective, minimizing the cover time, already into account when finding a tree for each robot to circumnavigate, whereas MSTC takes the objective only into account when it decides how the robots should circumnavigate the single tree.

VIII. Conclusions and Future Work

In this paper, we introduced a new multi-robot coverage algorithm, called Multi-Robot Forest Coverage (MFC). Our experimental results show that the cover time of MFC is smaller than the one of Multi-Robot Spanning-Tree Coverage (MSTC) and close to optimal in all tested scenarios. It is future work to make MFC robust in the presence of failing robots, a property that MSTC already has. We intend to augment MFC to handle robot failures by replanning trajectories for the functional robots that cover the remaining uncovered small cells. Furthermore, it looks very promising to combine the ideas behind MSTC and MFC, especially if several robots start in nearby small cells. We also intend to investigate ideas from other multi-robot coverage algorithms, such as [6] and [7].

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 print). In *Proceedings of the International Conference on Robotics and Automation*, 2005.

[6] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida. Cooperative sweeping by multiple mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 1744–1749, 1996.

[7] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot exploration of an unknown environment. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1340–1345, 1997.

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

[9] 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.