Proceedings of the 2006 IEEE International Conference on Robotics and Automation
Orlando, Florida - May 2006

Constructing Spanning Trees for Efficient Multi-Robot Coverage

Noa Agmon Noam Hazon and Gal A Kaminka
The MAVERICK Group
Department of Computer Science
Bar Ilan University, Israel
{segaln, haoznn, galk}@cs.biu.ac.il

***Abstract*— This paper discusses the problem of building efficient coverage paths for a team of robots. An efficient multi-robot coverage algorithm should result in a coverage path for every robot, such that the union of all paths generates a full coverage of the terrain and the total coverage time is minimized. A method, underlying several coverage algorithms, suggests the use of spanning trees as base for creating coverage paths. Current studies assume that the spanning tree is given, and try to make the most out of the given configuration. However, overall performance of the coverage is heavily dependent on the given spanning tree. This paper tackles the open challenge of constructing a coverage spanning tree that minimizes the time to complete coverage. We argue that the choice of the initial spanning tree has far reaching consequences concerning the coverage time, and if the tree is constructed appropriately, it could considerably reduce the coverage time of the terrain. Therefore the problem studied here is finding spanning trees that will decrease the coverage time of the terrain when used as base for multi-robot coverage algorithms. The main contributions of this paper are twofold. *First*, it provides initial sound discussion and results concerning the construction of the tree as a crucial base for any efficient coverage algorithm. *Second*, it describes a polynomial-time tree construction algorithm that, as shown in extensive simulations, dramatically improves the coverage time even when used as a basis for a simple, inefficient, coverage algorithm.**

I. Introduction

The general problem of multi-robot coverage is a fundamental problem in robotic systems with applications in various domains, from humanitarian missions such as search and rescue operations through military operations such as de mining to agriculture applications such as seeding or harvesting (e.g. [7], [10], [5], [8]).

This paper discusses the problem of building efficient coverage paths for a team of robots. In this problem, a team of robots each equipped with some tool, for example a sensor, are required to jointly sweep the entire given terrain while minimizing the total coverage time. Following the taxonomy presented in [2], we deal with offline coverage of the terrain, where the terrain is represented by an approximate cellular decomposition.

Several methods are found in the literature for coverage by single and multi-robot systems. One basic method that has received considerable attention is the method presented by Gabriely and Rimon [3], where the authors describe a polynomial time *Spanning Tree Coverage* algorithm, better known as the STC algorithm. In this method, Gabriely and Rimon assume that the robot is equipped with a square shaped tool of size D, hence the area was divided into N cells of size D placed on a grid. The grid was then coarsen such that each new cell is of size $2D\ X\ 2D$, and a spanning tree was built according to this new grid. After such a tree was built, the robot follows the tree around, creating a hamiltonian cycle visiting all cells of the original grid. The idea was first broadened for a multi-robot system in [5], by the family of MSTC algorithms. A different variation on this idea was introduced in [10].

When building the tree in a single robot system, the influence of the structure of the tree is theoretically irrelevant for the coverage time[1]. This results from the fact that coverage time is linear in the size of the grid, since each cell except for the boundary cells is covered once, hence the total coverage time is N. On the other hand, in multi-robot systems, the structure of the tree can have crucial consequences on the coverage time of the terrain. The choice of the spanning tree can change the robots' initial positions from being concentrated, i.e., placed as a bundle, to being scattered along the spanning tree path - all without actually changing the physical initial position of the robots. In general, we show that if the tree is appropriately built, the structure of the tree itself can substantially decrease the coverage time obtained by algorithms based upon it.

Hence, this paper specifically deals with constructing spanning trees for offline coverage that reduces the total coverage time of algorithms using these spanning trees as base for coverage. When constructing the trees we try to minimize the maximal distance between every two consecutive robots along the spanning tree path. If such tree is obtained, we show that all versions of the MSTC algorithm ran on these trees achieves substantially better coverage time compared to their coverage time on other randomly generated trees. Note that these trees, along with decreasing the coverage time of the algorithms which use them as base for coverage, also enjoy the benefits of the algorithms themselves. Specifically, if used as base for the family of MSTC algorithms, it promises robustness.

The tree construction algorithm we propose herein was tested both in a "clean" terrain, i.e., without obstacles, and also terrains with obstacles, where the obstacles are assumed to oc-

This work was funded in part by Israel's Ministry of Science and Technology

[1] In practice, it may affect efficiency due to the number of turns it requires, and other similar issues

cupy a full cell (or cells) of the coarse grid. The improvement results remained similar in all cases. This is despite the fact that when obstacles are added to a terrain, the task of coverage usually becomes more complicated. The algorithm we propose has a polynomial time complexity in the number of cells to be covered. This results in the surprising conclusion that as we add obstacles to the terrain, the complexity of the tree construction algorithm reduces, since the number of covered cells diminishes.

II. Background

The challenge of covering a terrain by a team of mobile robots has received considerable attention in the literature. Many approaches can be found in the literature for multi-robot coverage.

Coverage of terrains based on spanning trees is the approach which our research is based upon. As mentioned previously, this approach was first proposed by Gabriely and Rimon in [3]. They proposed the basic method of dividing the terrain into $2D\ X\ 2D$ cells, and described the polynomial time spanning tree coverage algorithm (STC) for complete offline and online coverage of the terrain. In [4], they suggest two different algorithms for building an online tree, but the motivation comes from the desire to create a spanning tree with a specific scanning direction.

The generalization of the single-robot STC algorithm to multi-robot systems was first introduced by Hazon and Kaminka in [5]. They have shown a simple offline algorithm for multi-robot coverage of a terrain by the MSTC algorithm, which guarantees robust, time efficient as well as complete coverage. They describe two versions of the MSTC algorithm: non-backtracking MSTC, and backtracking MSTC, herein referred to as NB_MSTC and B_MSTC, respectively. In the NB_MSTC algorithm the robots simply move in counterclockwise direction along the spanning tree path until reaching the initial position of the following robot if no faults occur, or take over the coverage path of the consecutive robot otherwise. In the B_MSTC the robots can backtrack over parts of their coverage path, i.e., they can go both clockwise and counterclockwise. They have shown that if the robots backtrack, the worst case performs up to twice as faster as in the non-backtracking case, even though re-covering was considered wasteful until then.

Later work by Zheng et. al. [10] proposed an additional multi-robot coverage algorithm, where their solution is based on dividing the spanning tree into k subtrees, not necessarily based on their initial location on the original tree. Their algorithm performs better compared to both MSTC algorithms, but their solution is not guaranteed to be robust. In addition, they note that different choices of trees result is different coverage time, but did not further discuss the issue.

Recent results by Hazon and Kaminka, described in [6], provide an optimal polynomial time coverage algorithm, herein referred to as Opt_MSTC. The algorithm is similar to the B_MSTC algorithm with modifications that assure the optimal coverage time given the initial locations of the robots and an initial spanning tree. The optimality is guaranteed only for the backtracking method, i.e., if the robots go back and forth *along the given spanning tree*. If the tree was to be constructed differently, perhaps suitable for the coverage problem, the results could substantially improve.

As mentioned previously, a survey of coverage in [2] proposes the notion of cellular decomposition in the coverage problem, which the basic approach of [3] belongs to. Another multi-robot coverage algorithm based on cellular decomposition is described in [9] by Wagner and Bruckstein. They explore the problem of room cleaning by a group of robots, and propose a robust algorithm for complete coverage of a terrain. In their case, the robots have limited capabilities and communicate with each other mainly using pheromones (in their case, dirt along the floor).

Many other approaches, other than ones based on cellular decomposition of the terrain, can be found in the literature for multi-robot coverage. For example, in [1], Batalin and Sukhatme offer two coverage algorithms by a multi-robot system in which the robots spread out in the terrain, and move away from each other while covering the area and minimizing the interaction between the robots. In their work, they aim to achieve optimal coverage *area*, and do not prove any formal statement regarding optimality of coverage time. Yet, similarly to their work, our algorithm uses the "spreading out" principle in building the coverage tree.

III. Motivation for building new spanning trees

In this section we describe the motivation behind our construction scheme of the trees. First, we show that the structure of the spanning tree has crucial role in the coverage time obtained by algorithms that use the tree as base for coverage. We prove that any algorithm that follows exactly the spanning tree path without having the robots bypass one another, does not come close to the possible improvement achieved by a different tree. Second, we show that a spanning tree which alone obtains the optimal coverage time does not necessarily exist, hence the theoretical optimal coverage time might remain unreachable in some cases. Last, we describe our definition of optimal spanning trees and explain the rational behind this definition.

As mentioned previously, an optimal time coverage algorithm for a system with k robots will (theoretically) result in total coverage time of $\lceil \frac{N}{k} \rceil$. The most basic multi-robot coverage algorithm will result in such a coverage time if the robots are uniformly placed along the spanning tree path, i.e., within distance of at most $\lceil \frac{N}{k} \rceil$ from one another.

We argue that the choice of spanning tree has crucial consequences on the coverage time obtained by algorithms using the spanning tree as base for coverage. This is more evidently seen in algorithms that do not diverge from the spanning tree path, such as the MSTC algorithms. An illustration of the importance of the right choice of spanning tree is given in Figure 1. The figure presents an example for a terrain in which $N = 36$, $k = 3$ and two different trees are suggested as base for coverage. The spanning tree is described by the bold lines,

and we use the different kinds of dashed lines to describe the spanning tree path, each dashed line represents the distance between two adjacent robots along the path. In order to clarify the example, the section between each two adjacent robots is given a different background as well. Note that in both grids the robots are initially located in the same positions. The tree in Figure 1a. places the robots uniformly along the tree path, thus a coverage time of $\lceil \frac{N}{k} \rceil$ is easily obtained if the robots simply follow the tree path in a counterclockwise direction. However, in Figure 1b. the robots are placed arbitrarily along the tree path, thus any multi-robot coverage algorithm, based on the spanning tree, will find it hard to result in such coverage time.

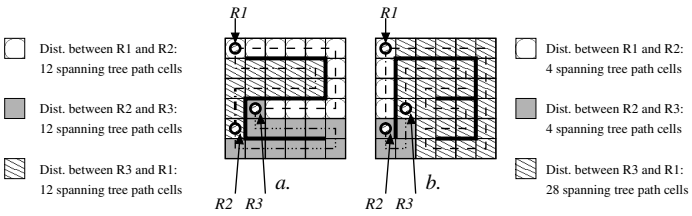

Fig. 1. Illustrating how different trees can influence coverage time.

A formal statement regarding the possible improvement in coverage time obtained by algorithms vs. improvement obtained by changing the tree is given by Theorem 1. First, let us introduce the following definition.

Definition: Given the initial positions of k robots on a terrain with N cells, let M be the coverage time of the terrain obtained by the basic NB_MSTC algorithm. A procedure $\mathcal{P}$ or an algorithm $\mathcal{A}$ are said to ensure an *improvement factor* t, if the coverage time obtained by NB_MSTC after applying $\mathcal{P}$ on the input, or the coverage time obtained by $\mathcal{A}$ on the same input is $\frac{M}{t}$.

Theorem 1: Any multi-robot coverage algorithm for homogenous robots based on a spanning tree which does not divert from the spanning tree path will result in a maximal improvement factor of at most 2.

Proof: Denote the distance between the initial location of robot R_i and R_{i+1} on the spanning tree path by D_i (also known as the segment D_i), and let $D_{max} = \max_{1 \le j \le N}\{D_j\}$. Clearly, the coverage time obtained by NB_MSTC is exactly D_{max}. Also, D_{max} determines the coverage time of any coverage algorithm $\mathcal{A}$ that does not divert from the spanning tree path. As the robots are homogenous and cannot bypass one another (assuming they are nonfaulty), an improvement in the coverage algorithm can reduce from D_{max} to $\lceil \frac{D_{max}}{2} \rceil$ if robots on the extremity of D_{max} should simply walk towards one another while covering the terrain. If there is some other segment D_j which requires coverage time of some $t' > \lceil \frac{D_{max}}{2} \rceil$, then the new coverage time is t'. Note that t' can be smaller than the distance D_j if an algorithm allowing backtracking is permitted. In other words, the improvement factor is

$$\frac{D_{max}}{\max\{\lceil \frac{D_{max}}{2} \rceil, t'\}} \le 2$$

While the change of the coverage algorithm can result in an improvement factor of at most 2, the example described in Figure 2 leads us to the conjecture that improvement factor can reach almost the value of k just by changing the spanning tree. As seen in Figure 2b., the coverage time obtained by NB_MSTC is $N - k = 56 - 3 = 53$, while the coverage time obtained by the same algorithm on a spanning tree constructed in a way that places the robots in an equally scattered way along the tree (Figure 2a.) is $\lceil \frac{N-k}{k} \rceil = 19$, hence the improvement factor obtained by changing the tree is $\frac{53}{19} \approx 2.8$, which is almost k.

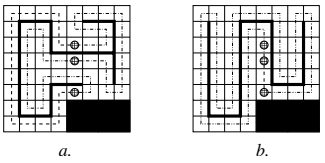

Fig. 2. An example of a case in which the improvement factor is almost k if the tree is appropriately constructed.

We have established the fact that the choice of a spanning tree can have far reaching consequences on the coverage time of the terrain, possibly more than the choice of the coverage algorithm. Moreover, a spanning tree that places all robots within distance of at most $\lceil \frac{N}{k} \rceil$ will, by itself, result in the optimal coverage time. Unfortunately, such a tree does not necessarily exist. For example, in Figure 3, $N = 16$, $k = 2$ and all possible spanning trees are described. The minimal maximal distance between two consecutive robots over all possible spanning trees is 10 cells, where $\lceil \frac{N}{k} \rceil = \lceil \frac{16}{2} \rceil = 8$.

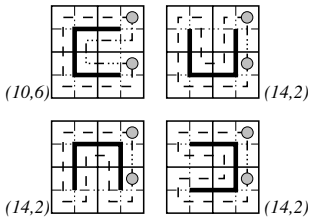

Fig. 3. An example of a case in which there is no spanning tree that have maximal distance of $\lceil \frac{N}{k} \rceil = \lceil \frac{16}{2} \rceil = 8$ between consecutive robots along the spanning tree path. The numbers in parenthesis describe the distance between two robots along the spanning tree path.

In our tree construction scheme we will try to approximate this optimal dispersion of robots along the spanning tree. We will do that by trying to satisfy the following objective, as much as possible. First, let $\tilde{G}$ be a grid with $N/4$ cells, possibly containing obstacles (the obstacles are not counted as cells). Let G be $\tilde{G}$'s fine grid after dividing each cell into four cells of size D.

Objective: Given the initial locations of k robots on cells of G, find a spanning tree of $\tilde{G}$ that minimizes the maximal distance between every two consecutive robots along the spanning tree path.

The idea behind this objective is that it spreads the robots as uniformly as possible along the spanning tree path. Consider, for example, the Opt_MSTC algorithm. They create optimal paths along the spanning tree for the k robots, not allowing nonfaulty robots to bypass one another. There, even in the worst initial distribution case in which all robots are bundled in their initial position, the best possible improvement will result in an improvement factor of approximately 2 : from $N-k+1$ to $\frac{N-k}{2}+1$. On the other hand, the improvement by spreading the robots along the spanning tree can reach nearly a factor of k : from $N-k+1$ to $\frac{N}{k}$.

The construction of an *optimal* tree, that will achieve exactly the objective, is believed to be $\mathcal{NP}$-hard [10]. Hence our tree construction algorithm can be considered as a heuristic algorithm for the problem of finding the optimal tree for the coverage task.

IV. Tree construction algorithm

In this section we describe the algorithm Create_Tree. This algorithm creates spanning trees while considering the initial location of all robots in the team and the objective described above, i.e., it tries to minimize the maximal distance between any two adjacent robots on the tree.

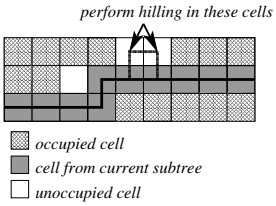

Fig. 4. Illustration of the Hilling procedure.

The algorithm, described in Figure 5, is composed of two stages. First, a subtree is created gradually for each robot starting from the initial position of the robot, such that in each cycle either one or two cells are added to each subtree. Denote the subtree originated in R_i by T_{R_i}. The cells are chosen in a way that maximizes the distance from current expansion of all other trees. First, the algorithm tries to find the longest possible path for the tree. When it fails to continue, it tries to perform Hilling, in which it looks for two joint unoccupied cells adjacent to the path. If it found such cells, then it adds them to the path as demonstrated in Figure 4. If the algorithm failed to find more hills, then it expands the tree, from both sides of the path, as symmetrically as possible. First it attempts to add one cell to its right, then one cell to its left, and so on, until the entire grid is covered by all k disjoint subtrees.

After such k subtrees are generated, it is only left to connect them (second stage). Denote an edge connecting two different trees T_{R_i} and T_{R_j} by $\mathrm{br}(T_{R_i}, T_{R_j})$. As we are given k subtrees to be connected to one tree covering the entire grid, it is required to find $k-1$ bridges. These bridges should be chosen in a way that the resulting tree does not contain cycles or, equivalently, cover the entire grid. For example, if $k = 4$ then possible valid choice of bridges are $\{\mathrm{br}(T_{R_1}, T_{R_2}), \mathrm{br}(T_{R_1}, T_{R_3}), \mathrm{br}(T_{R_3}, T_{R_4})\}$, where $\{\mathrm{br}(T_{R_1}, T_{R_2}), \mathrm{br}(T_{R_2}, T_{R_3}), \mathrm{br}(T_{R_1}, T_{R_3})\}$ is invalid, as T_{R_4} remains disconnected. Create_Tree picks randomly a valid choice of $k-1$ bridges, and calculates the maximal distance between two adjacent robots on the tree according to the **fine** grid. It repeats the process k^2 times, and reports the best tree it observed, according to the above criterion.

Clearly, the algorithm provides complete coverage of the terrain, as the first stage of constructing subtrees does not end before every cell is occupied by some subtree. The first stage terminates, as in each cycle at least one cell is added to at least one subtree, hence given a finite terrain the algorithm halts.

Procedure Create_Tree

1) Build k subtrees as follows.
   For every robot R_i, $1 \le i \le k$ **Do**:
   a) **For** each possible next cell (up, down, right, left), compute the Manhattan distance from the current location of all other robots.
   b) **If** more than one possible next move exists, **then** pick the one whose minimal distance to any other robot is maximized.
   c) **If** there is no next possible move, **then** perform Procedure Hilling from the last main branch.
   d) **If** failed to find an unoccupied cell in Hilling, **then** branch out, as symmetric as possible, from the main branch to all possible directions.
2) Find all possible bridges between the k trees.
3) **For** $i = 0$ to k^2 **Do**:
   a) At random, find a valid set of bridges B_i between trees such that they create one tree with all N vertices.
   b) Compute the set S_i of distances between every two consecutive robots on the tree.
   c) Best_Result is initialized with S_0.
   d) **If** the maximal value in S_i is lower than the maximal value in Best_Result, **then** Best_Result $\leftarrow S_i$.
4) Return the tree associated with Best_Result.

Fig. 5. Description of Create_Tree algorithm.

Theorem 2: The time complexity of Create_Tree algorithm is $\mathcal{O}(N^2 + k^2 N)$.

Proof: In the stage where k subtrees are created, in the worst case when adding one cell to a subtree the algorithm runs over all current cells in the subtree (during Hilling or while branching out), hence the complexity is at most $\mathcal{O}(N^2)$. In the second stage, where the trees are connected, k^2 different choices of trees are examined, each time the entire tree is traversed, thus the complexity of this stage is $\mathcal{O}(k^2 N)$. Hence the entire complexity of the algorithm is $\mathcal{O}(N^2 + k^2 N)$. ∎

V. Experimental results

We have evaluated the effect of the tree construction algorithm Create_Tree on the coverage time obtained by NB_MSTC, B_MSTC and Opt_MSTC while taking two other parameters under consideration. First, the number of robots - from 3 to 30 robots. The second parameter was the density of obstacles in the terrain. The coverage time obtained by the

above algorithms on the trees constructed by Create_Tree was compared against coverage time obtained by the algorithms running on randomly generated spanning trees. The terrain over which the experiment was ran was a $20X30$ coarse grid (600 coarse cells, or 2400 fine cells). We have first performed the experiment on a grid with no obstacles ("clean" grid), then added at random $40\,(6.6\%)$, $80\,(13.3\%)$, $100\,(16.7\%)$ and $160\,(26.7\%)$ obstacles to the coarse grid.

Each trial was run for every number of robots (from 3 to 30) and for every density of obstacles in the terrain. First, we have created 200 input lines by each tree construction method: randomly generated trees and Create_Tree generated trees, where each input line represents a random initial distribution of the robots (altogether 400 experiments for each basic setting). These input lines were later given to the NB_MSTC, B_MSTC and Opt_MSTC algorithms and the coverage times obtained by these algorithms were compared.

The average coverage times obtained by algorithms NB_MSTC and Opt_MSTC are brought in Figures 6, 7, 8, 9 and 10. The results show clearly that the average coverage time obtained by running algorithms NB_MSTC and Opt_MSTC on trees constructed by algorithm Create_Tree are statistically significantly better (using paired two-tailed t-test, where the p-value is less than 10^{-12}) than the average coverage time obtained by those algorithms when ran on randomly generated trees. Moreover, the coverage time obtained by running the simplest non-backtracking MSTC algorithm on the trees generated by Create_Tree is, in most cases, even lower than the *optimal* MSTC algorithm ran on randomly generated trees. These results repeated in both dimensions in which the experiment was conducted: number of robots and density of obstacles. The results from running the experiment on B_MSTC are omitted for clarity reasons of the display, but they are compatible with all other results.

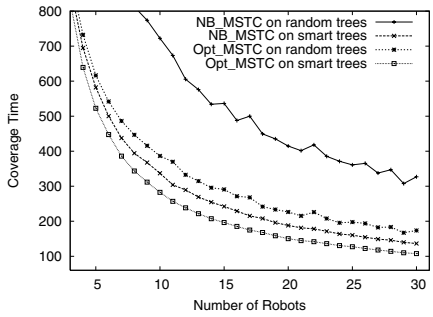

Fig. 6. Results from comparing random trees with trees generated using Create_Tree algorithm on a grid with no obstacles.

An interesting result follows from comparing the improvement in coverage time obtained by the algorithms after performing Create_Tree with different density of obstacles in the terrain. While the improvement in the coverage time obtained by the algorithms after running Create_Tree remains statistically significant compared to randomly generated trees, as the obstacles become more dense the improvement lessens. For instance, the improvement percentage for 30 robots with

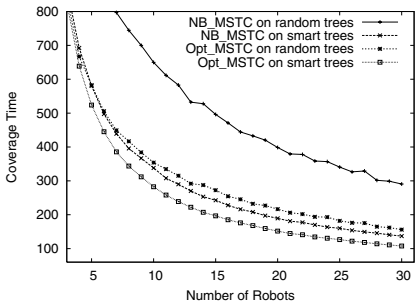

Fig. 7. Results from comparing random trees with trees generated using Create_Tree algorithm on a grid with 6.7% of the cells blocked by obstacles.

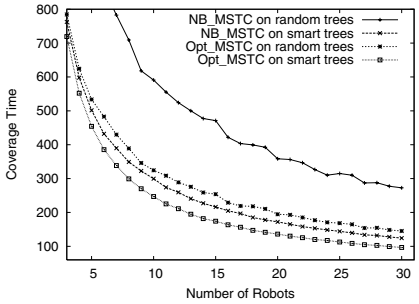

Fig. 8. Results from comparing random trees with trees generated using Create_Tree algorithm on a grid with 13.3% of the cells blocked by obstacles.

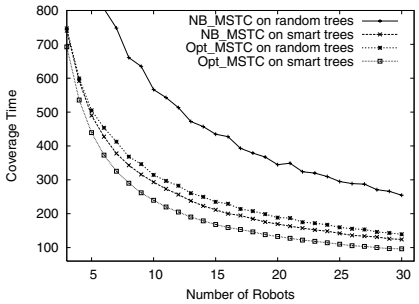

Fig. 9. Results from comparing random trees with trees generated using Create_Tree algorithm on a grid with 16.7% of the cells blocked by obstacles.

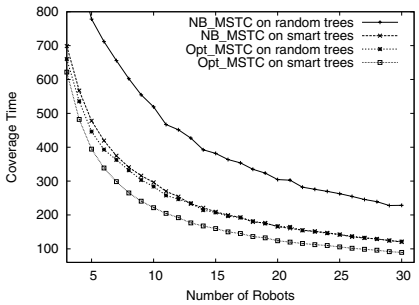

Fig. 10. Results from comparing random trees with trees generated using Create_Tree algorithm on a grid with 26.7% of the cells blocked by obstacles.

no obstacles for the NB_MSTC and Opt_MSTC algorithms are 58% and 38%, respectively. When the density of obstacles is 26% the improvement percentage decreases to 48% and 28% (respectively).

Figure 11 presents as an example the improvement percentage in coverage time between Create_Tree generated trees vs. randomly generated trees followed by the execution NB_MSTC algorithm. Note that the repetitiveness of the phenomenon is *not* absolute over all number of robots, to our opinion most likely due to the structure of the terrain, but the trend is clear. The reason for this phenomenon is simple: the Manhattan distance computed in the first step of the algorithm does not take into account obstacles along the way. Hence, if obstacles are added, the algorithm might create the subtrees based on a distorted conception of the terrain thus the improvement, while still significant, lessens.

An additional interesting results follows from comparing between the percentage of improvement of the results obtained by the Opt_MSTC algorithm (Figure 12) and the NB_MSTC algorithm (Figure 11). In both cases the improvement percentage from using the Create_Tree generated trees is relatively high, although using the NB_MSTC coverage algorithm results in much higher improvement percentage. This change is originated in the fact that if using the simple NB_MSTC algorithm the change in coverage time is much more evident. The Opt_MSTC algorithm by itself performs some improvement in coverage time, so there is less to improve from that point.

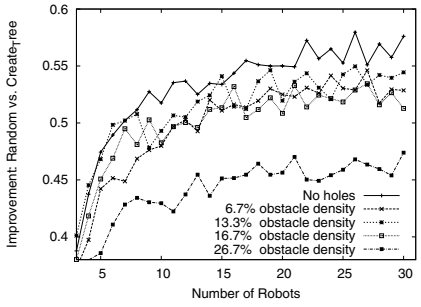

Fig. 11. Comparison between the improvement percentage in coverage time obtained by algorithm NB_MSTC after generating trees randomly vs. using Create_Tree algorithm with different density of obstacles in the terrain.

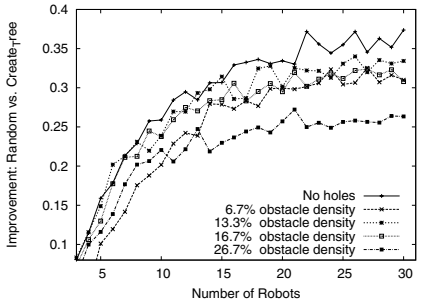

Fig. 12. Comparison between the improvement percentage in coverage time obtained by algorithm Opt_MSTC after generating trees randomly vs. using Create_Tree algorithm with different density of obstacles in the terrain.

VI. Conclusions and Future work

In this work we have discussed the importance of the structure of the spanning tree on the coverage time obtained by algorithms that use this tree as base for coverage. First, we have shown that the structure of the tree can have crucial consequence on the coverage time. We have presented an algorithm for constructing trees that is motivated by the objective similar to the one defining an optimal tree, a problem that is thought to be $\mathcal{NP}$-hard. We have extensively tested the influence of the spanning tree structure on the coverage time obtained by NB_MSTC, B_MSTC and Opt_MSTC while taking several parameters under consideration. Simulation results show that when using trees generated by Create_Tree the resulted coverage time obtained by all algorithms were statistically significantly better than the results obtained by running the algorithms on randomly generated trees. Moreover, the average coverage time obtained by the simplest NB_MSTC algorithm on spanning trees created by Create_Tree were, in most cases, better than the average coverage time obtained by Opt_MSTC on randomly generated trees.

There are still several areas we plan to pursue in future work. First, we would like to examine by simulation the compatibility of the trees created by algorithm Create_Tree to coverage algorithms that use the spanning tree as base for coverage, but divert from the spanning tree path from time to time (for example, the algorithm described in [10]). Second, we would like to establish the complexity class which the problem of building optimal trees belongs to. It is believed (strengthened by [10]) that this problem is $\mathcal{NP}$-hard, although it is not proven yet. In addition, we would like to expand our algorithm to dealing with complete cellular decomposition of the terrain, i.e., include also cells that do not exist completely in the coarse grid. Finally, we would like to find a way to improve our treatment of obstacle in the terrain.

References

[1] M.A. Batalin and G.S. Sukhatme. Spreading out: A local approach to multi-robot coverage. In *Proc. of the 6th Internat. Symposium on Distributed Autonomous Robotic Systems*, page 373382, 2002.

[2] H. Choset. Coverage for robotics a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, 2001.

[3] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. *Ann. Math. Artif. Intell.*, 31(1-4):77–98, 2001.

[4] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comp. Geometry*, 24:197–224, 2003.

[5] N. Hazon and G. A. Kaminka. Redundancy, efficiency and robustness in multi-robot coverage. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[6] N. Hazon and G. A. Kaminka. Redundancy, efficiency and robustness in multi-robot coverage. Technical report, MAVERICK Group, Bar-Ilan Univ., 2005.

[7] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31:7–40, 2001.

[8] I. Rekleitis, V. Lee-Shue, A. Peng New, and H. Choset. Limited communication, multi-robot team based coverage. In *IEEE International Conference on Robotics and Automation*, pages 3462–3468, 2004.

[9] I. A. Wagner and A. M. Bruckstein. Cooperative cleaners - a study in ant robotics. In *Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath*, pages 289–308. Kluwer Academic Publishers, 1997.

[10] X. Zheng, S. Jain, S. Koenig, and D. Kempe. Multi-robot forest coverage. In *Proc. IROS*, 2005.