GrasSmart2: A Multi-Robot Terrain Coverage System

Miri Weiss-Cohen, Oz Bar, and Shai Shapira,

***Abstract*—Our system named GrasSmart2 is designed to develop and implement a solution to the problem of building efficient coverage paths for a team of robots. The system generates an efficient multi robot coverage algorithm by constructing a coverage path for every robot, such that the union of all the generated paths means that the terrain is fully covered and that the total coverage time is minimized. This work modifies and improves the Create Tree for Efficient Multi-Robot Coverage algorithm and implements it in the GrasSmart2 system. Moreover, our system finds a position for a new robot when a group of robots are constrained in a geometric position. GrasSmart2 finds the best positioning coordinates that achieve a total minimal covering path.**

***Index Terms*—Terrain, coverage, multi robot systems, cell decomposition**

I. Introduction

Research has increasingly been focusing on multi agent and multi robot tasks. A branch of this research area is the use of multiple robots in coverage[12-15]. Sam and Fua [2] defined the coverage problem as the maximization of the total area covered by a robot's motion. The static coverage problem is as follows: how should a robot be deployed in a static configuration such that every point in the environment is known to the robot, and is analyzed to find the optimal or preferred path for covering the terrain. Many of the real world automated technologies involve area coverage tasks; mapping and validation of topological maps, automated mine-sweeping, and more [9]. As a result of the evolution of automated devices, simple and house-hold tasks such as vacuum cleaning, snow removal, painting, lawn mowing, and pool cleaning are being left to the care of robots [1,7,8,11].

To accomplish these tasks, a robot is given a bounded work area, which in many cases contains obstacles. The area coverage problem may be looked at as a geometric version of the Covering Salesman Problem [10]. As stated in [3], the multiple agent algorithms solve the covering problem in much more efficiency than a single robot solution for two main reasons. Firstly, by dividing the work area between them, multiple robots will complete the task more quickly than a single robot. Secondly, multi-robot systems are more likely to reach the assigned objective; if a single robot fails, the other robots in the team simply cover its assigned area [5].

Single-robot coverage problems are solved in polynomial time. The problem becomes significantly more complex when we try to minimize the cover time using multiple robots. The multi-robot coverage problem is NP-complete [12]. Hazon et al. [4] dealt with this problem by using the Spanning Tree Covering (STC) algorithm [6], which was generalized to the Multi-Robot Spanning Tree Coverage (MSTC) algorithm. Hazon et al. [4] improved the cover time by finding the longest segment in the possible path and dividing it evenly between two adjacent robots. This reduces the STC cover time by a factor of at least 2 (or *3/2*) for $K \geq 3$. The work by Agmon et al in [4] solved the multi-robot system by constructing an MSTC algorithm by constructing one search tree for each robot in the group.

II. The Improved Create Tree for an Efficient Multi-Robot Coverage Algorithm

GrasSmart2 is a simulation system which was developed for efficient Multi robot covering procedure. Our new algorithm modifies and improves the algorithm: Create Tree for Efficient Multi-Robot Coverage algorithm, which is detailed in [4]. Our system improved the time and space complexity of the paths found by the system. The user is free to input any work area, and any number of constraints and obstacles. As in MSTC [5,6], we defined the robot or agent's tool to be a square of size D. The work area is then approximately decomposed into cells, with each cell being a square of size 4D. As with other approximate cell-decomposition approaches [2], cells that are partially covered by obstacles or outside the bounds of the work area and are ignored.

When constructing the spanning trees, the algorithm tries to minimize the maximal distance between every two consecutive robots along the spanning tree path. Robustness and efficiency for the above is detailed in [5]. The Create Tree for Efficient Multi-Robot Coverage algorithm has a polynomial time complexity in the number of cells to be covered. Fig. 1 indicates that the structure of the spanning tree strongly influences the algorithm's coverage time. Hazon et al. [5] proved that any algorithm that follows a spanning tree path exactly, without having the robots bypass one another, does not necessarily improve the path achieved by a different tree path which is not optimal nor restricted by any coverage time criteria. Fig. 2 shows an example of two different spanning trees defined for the same terrain producing different coverage time results [4].

Manuscript received November 10, 2010; revised December 25, 2011.
Miri Weiss-Cohen is a Senior lecturer in the Software Engineering Department of Braude College of Engineering, POB 78, Karmiel 21982, Israel (e-mail: miri@braude.ac.il).
Oz Bar is with ISCAR Ltd.. TEFEN Industries Kfar Veradim Israel (e-mail: Ozbar@gmail).
Shai Shapira is with CMT Medical Technologies Ltd Hacarmel Street, Building 7/2 P.O.B. 111 Yokneam Ilit 20692, Israel (email: shai@cmt.co.il)

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Procedure Create_Tree

1) Build k subtrees as follows.
   For every robot R_i, $1 \leq i \leq k$ **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. 1. Create_Tree.

The algorithm, shown in Fig. 1, has two stages. First, a sub-tree is created gradually for each robot, starting from the robot's initial position such that in each cycle either one or two cells are added to each sub-tree. This is done by trying to find the longest possible path for the tree. When the algorithm fails (cannot find the longest path), it tries to perform a Hilling Procedure (see Fig. 3), in which it looks for two contiguous, unoccupied cells adjacent to the path. If the algorithm finds such cells, it adds them to the path as demonstrated in Fig. 3. If the algorithm fails to find more hills, it expands the tree, along both sides of the path, attempting 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 sub-trees.

After k sub-trees are generated, the algorithm must find $k-1$ bridges in order to connect the k sub-trees so that there will be one tree covering the entire grid. These bridges should be chosen in a way that the resulting tree does not contain cycles but covers the entire grid. Create Tree randomly picks a valid number of $k-1$ bridges, and calculates the maximal distance between two adjacent robots on the tree according to the fine grid. It repeats this process K^2 times, and reports the best tree observed, according to the above criterion.

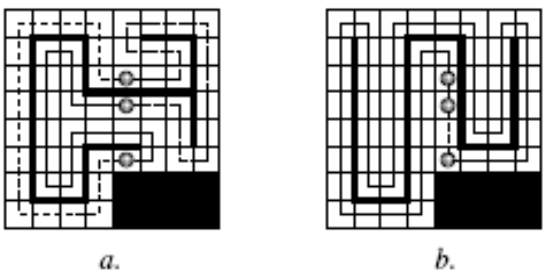

Fig. 2. Two options for constructing a tree [4].

The time complexity of the Create Tree Algorithm is $O(N^2 + K^2 N)$ [4].

Our system, which we call GrasSmart2, improves the Create Tree Algorithm by twofold to achieve better complexity, as follows:

Instead of the hilling, which occurs when the algorithm fails to continue (one robot cannot continue to move away from the other robot or robots), it tries to look for two contiguous, unoccupied cells adjacent to the path. If it finds such cells, it adds them to the path. When our algorithm fails to continue, it retraces its path along the tree and searches for the first point in which it leaves the path and continues to another direction until it fails. At this point it again retraces the path and so on.

Another improvement is introduced for the situations when the coverage time without any of the bridges is better than the time with the performing bridges. The original algorithm first calculates the time without any bridges and then the time with the best bridges. Once this is completed, it determines and uses the minimal between the two (that is, it takes K^2 random bridges, and from among these, finds the best set of K-1 bridges). Our algorithm takes 3*K*N random bridges, where K is the number of robots and N is the size (number of cells) of the terrain, and from among these, finds the best set of K-1 bridges.

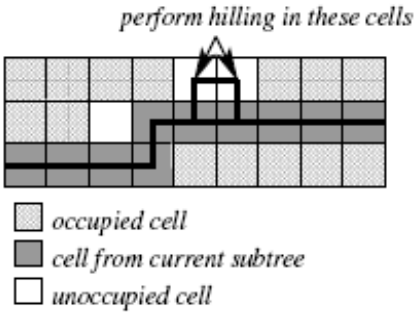

Fig. 3. Hilling Procedure [4].

The Pseudo Code of the algorithm with the relevant changes we made is as follows:

1. Build K sub-trees as follows
   For each robot R, $0 < i < k$, do
   1.1 For each possible next cell (up, down, right, left), compute the Manhattan distance from the current location of all other robots.
   1.2 If more than one possible next move is exists, then pick the one whose minimal distance to any other robot is maximized.
   1.3 If there is no next possible move, then move back on the tree until finding one possible move, if there is more than one possible move select the next move like 1.2 (Note: the next move will start from the new position).
   1.4 If failed to find an unoccupied cell then branch out, otherwise go back to step 1.1
2. Each robot circumnavigates his tree
   Best_Result $\leftarrow$ Maximum (time that took to each robot to circumnavigates his tree)
3. Pick 3*K*N random bridges between the k trees.
4. For each set of K-1 bridges do
   4.1 Set bridges.
   4.2 Compute the set S_i of distances between every two consecutive robots on the tree.
   4.3 If the maximal value in S_i is lower than the maximal value in **Best_Result**, then **Best_Result** $\leftarrow S_i$
5. For **Best_Result** add outside tree cell as follows
   For each robot R, $0 < i < k$ run over the tree

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

5.1 If you find an unoccupied cell and it is not inside a cell that is already occupied, then add this cell to your tree (if more than one robot wants to add this cell, give priority to robots with the lowest number).

6. Return the tree associated with **Best_Result.**

Fig. 4. The Improved Create_Tree Algorithm

The complexity of the algorithm with our changes is:

$O(N^2 + 3KN^2)$

The algorithm includes a step in which it tries to build all the bridges between the robots' separate spanning trees (Steps 3-4 in Fig. 4), provided the terrain is large enough so that the length of each spanning tree is sufficiently long and the algorithm has enough bridges to test. Thus the algorithm randomizes only 3*K*N bridges and tests only them and find the best pick of K-1 bridges in order that the coverage time will be optimal. Because of this solution, when having a big map there might be different paths to the robots and different results between some results running the algorithm because each time it randomizes different 3*K*N bridges and finds the best choice of K-1 bridges, respectively.

In addition to the above, the algorithm can also run without the random pick of 3*K*N bridges. (This option might take longer to run, especially if a big area has to be covered).

The complexity of this option is:

$O(N^2 + (N/K)^K = O(N^K)$.

III. Initial Robot Positioning

In many cases in the real world, a new robot must be introduced into a set of existing robots that have a fixed initial position. We suggest a calculation based on the geometry of the terrain, which provides some heuristics in order to detect the most efficient coordinates to place the new robot so that all the robots' coverage time will be optimal. Fig. 5 depicts a terrain where two robots have already been placed and marked by points. We want to find the best placement for a third robot that will also give us the optimal coverage time.

We define the locations of the two robots as (*x1*,*y1*) and (*x2*,*y2*), respectively and *Xmax, Xmin, Ymax, Ymin* ,the coordinates of the terrain.

The following steps are calculated:

1. Define *A* and *B* for the known terrain as depicted in Fig. 5:

$$A = \frac{\text{Xmax-Xmin}}{\text{NumOfRobots}} \qquad B = \frac{\text{Ymax-Ymin}}{\text{NumOfRobots}}$$

NumOfRobots – total number of robots to be placed in the terrain

2. Calculate the distance between every two robots and then average all the distances:

$$D = \frac{1}{n}\sum D_{ij}$$

n is the number of distances, and D_{ij} is the distance between Robots *i* and *j*, with the locations (x_i,y_i) and (x_j,y_j), respectively), given by

$$D_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

*R = R * 1.2* expresses the radius in which the new robot cannot be placed.

Combining all the steps gives:

$$R = \frac{A + B + D}{3} \times 1.2$$

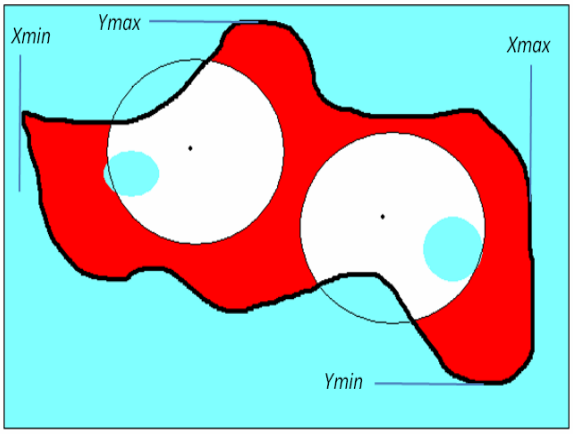

Fig. 5. Initial robot positioning; circles depicted to calculate possible new robot coordinates.

In Fig. 5, the red area has been calculated to include all the possible positions where the robots can be placed. A number of points are processed randomly so that the best spanning tree path can be chosen. The option of combining the robots' static and dynamic locations provides an optimal and less constrained solution. The full details are given in Example 3.

IV. Examples

Fig. 6a-c depict the result of the system simulation. Fig. 6a shows the initial positioning of the four robots. Note the excellent results when the robots are located as far as possible one from another. Fig. 6b shows an in-between state of the robots' collective procedure and the tree that has been built. The robots paths' are colored in pale green. Fig. 6c depicts the end result of the area covered by the four robots. In this simple example, the terrain does not contain obstacles or constraints. Fig. 6d is a table showing the statistical calculations made by the system, such as percentages of cells that were covered more than once, percentages of cells that were covered more than once, number of turns made by all the robots, percentages of total area that was covered by the algorithm, time, and percentage of coverage as compared to the optimum theoretical value.

Example 2 contains two obstacles and constrained areas. Fig. 7a and b depict two intermediate stages of three robots and three trees. Fig. 7c is the statistical results. It can be seen that 94% of the area is covered, with a minimal percentage of coverage duality.

Example 3 is an initial positioning example where three robots were positioned randomly (the user can define a fixed position), and the fourth robot is positioned by calculating one of the best coordinate options. Fig. 8a illustrates the initial state of the robots and the calculated position. Fig. 8b shows what happens after the improved algorithm is applied and simulated. It can be seen in the examples that the blue robot has finished traversing his tree, where the other robots are still on their working path. This situation is the result of the robots being placed very close to one another at the start.

In a perfect world, the robots should be as spread apart as possible. In reality, this is often not possible (in many real life applications, robots start from a single initial point, and must return to a single finishing point.)

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

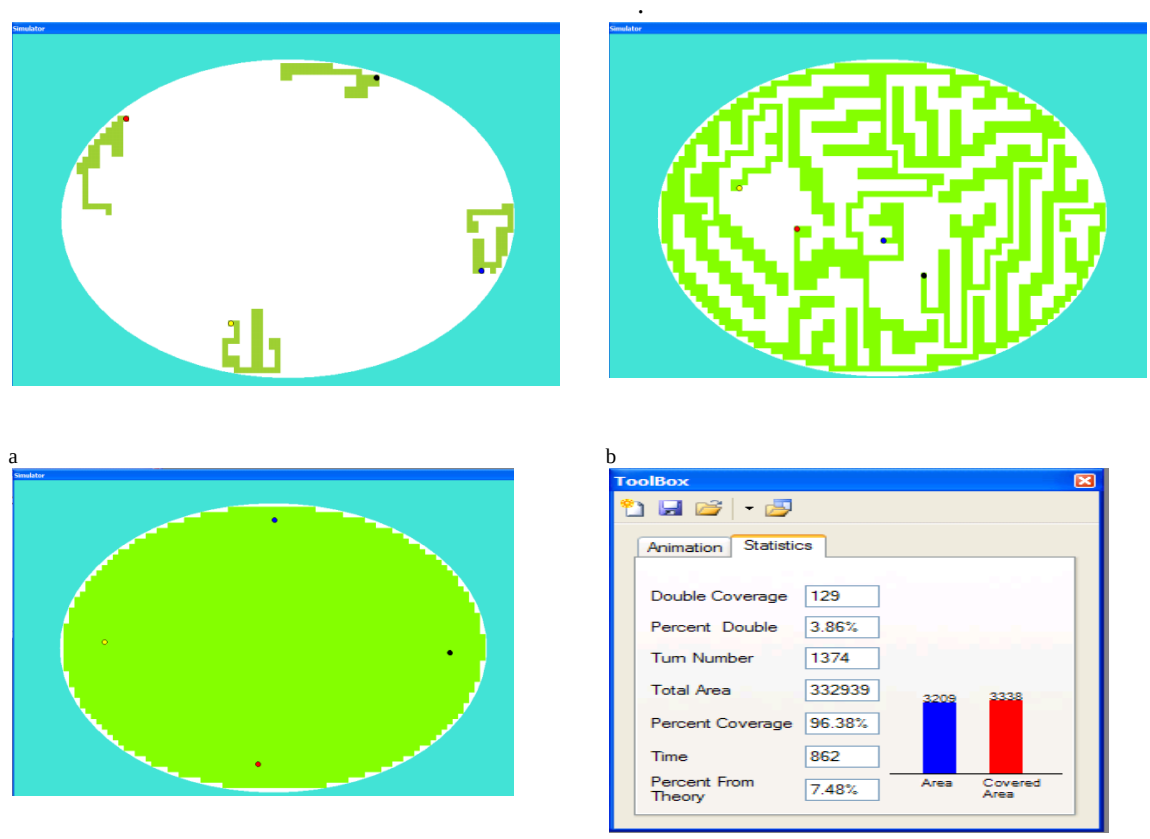

Fig. 6. (a)-(c) A simple example of the multi robot procedure. (d) The statistical results.

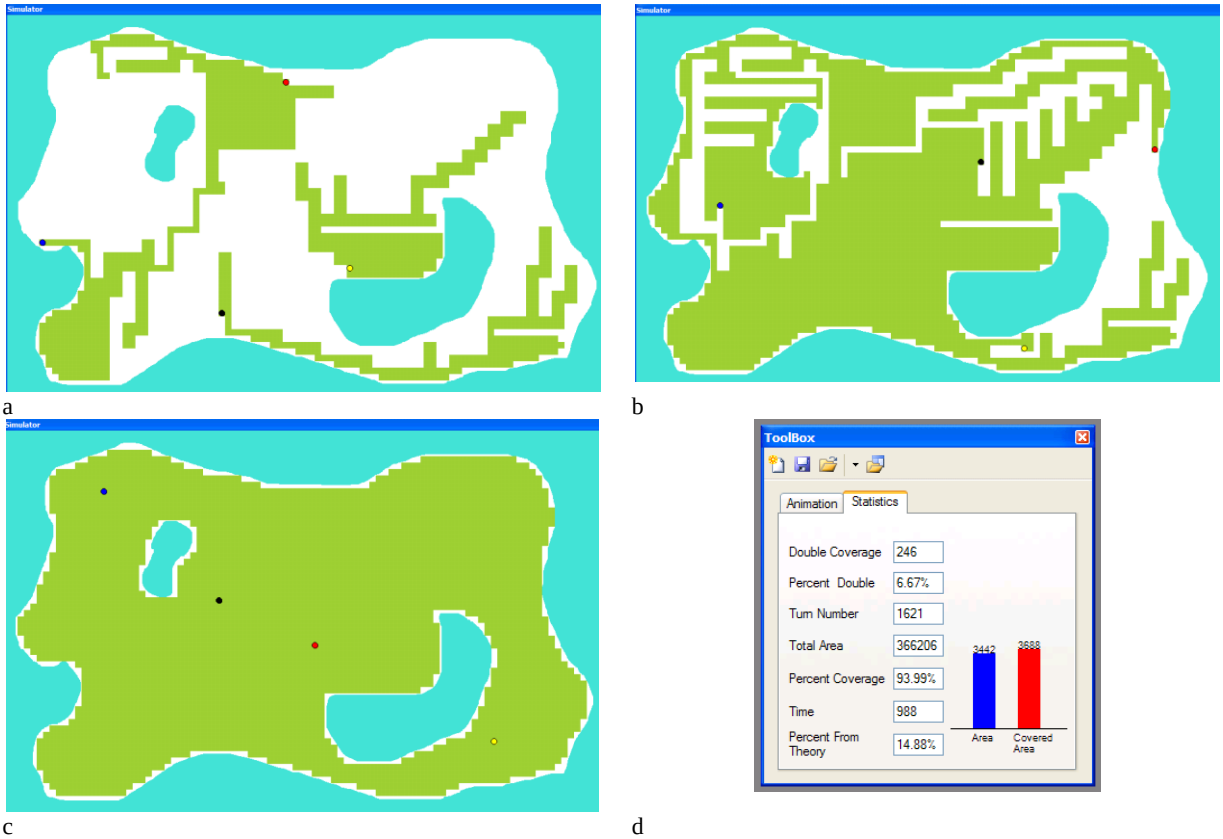

Fig. 7. (a)-(c) An example of the multi robot procedure. (d) The statistical result.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

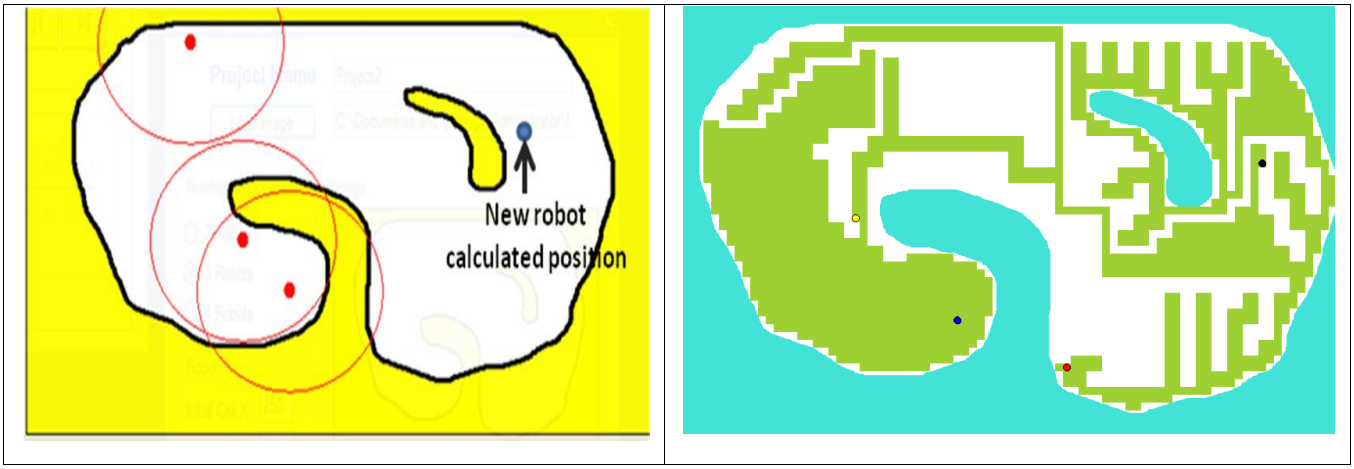

Fig. 8. (a) Calculating a new position for the 4th robot. (b) An interim state of the simulation.

V. CONCLUSION

In this work we improved the Create Tree Algorithm as a basis for solving the coverage path problem. GrasSmart2 is a simulation system where a number of robots are placed on a terrain with obstacles, and the paths are calculated and simulated. The system takes into consideration new parameters intended to improve the STC algorithm. In the suggested algorithm a mobile robot, given a bitmap of a known geometric area as input, derives an optimal coverage path for a given area. The results of GrasSmart2, the program that mimics a grass cutting robot's path and provides statistical calculations for testing optimality, presented and validated our improved algorithm. Run-time results in terms of area over-coverage and edge-completeness in terms of the relative number of cells in the coverage. The positioning of robots is addressed and calculated. Some examples of the system are presented.

REFERENCES

[1] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," *IROS*, 2005, Edmonton, Canada.

[2] S. Sam and C. Fua, "Complete multi-robot coverage of unknown environment with minimum repeated coverage," *IEEE Int. Conf. on Robotics and Automation*, 2005, Barcelona, Spain.

[3] N. Agmon, N. Hazon and G. Kaminka, "Constructing spanning trees for efficient multi-robot coverage," *IEEE Int. Conf. on Robotics and Automation*, 2006, FL, USA.

[4] N. Hazon and G. A. Kaminka, "Redundancy, efficiency, and robustness in multi-robot coverage," *IEEE ICRA*, 2005, Barcelona, Spain.

[5] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Ann. Math. AI.*, vol. 31, pp. 77–98, 2001.

[6] M. A. Batalin and G. S. Sukhatme, "Spreading out: A local approach to multi-robot coverage", *Int. Sym. on Distributed Autonomous Robotic Sys.*, 2002, Fukouka, Japan.

[7] I. Rekleitis, G. Dudek, and E. Milios. "Multi-robot exploration of an unknown environment," in *Proc. of the Int. Joint Conf. on AI*, 1997, Nagoya, Japan, pp. 1340–1345.

[8] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Ann. Math. AI.*, vol. 31, pp. 7–40, 2001.

[9] H. Choset, "Coverage for robotics. A survey of recent results," *Ann. Math. AI.*, vol. 31, pp. 113–126, 2001.

[10] E. M. Arkin, and H. Refael, "Approximation algorithms for the geometric covering salesman problem," *Discrete Appl. Math.*, vol. 55, pp. 197–218, 1994.

[11] Z. Chai, and Z. Peng, "Cooperative coevoulutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot system," *J. of Intelligent and Robotic Systems*, vol. 33, 2002, pp. 61–71.

[12] J. Svennebring and S. Koening, "Building terrain-covering and robots," Autonomous Robots, vol. 16, 2003, pp. 313–332.

[13] C. S. Kong, N. A. Peng and I. Rekleitis, "Distributed coverage with multi robot system," *IEEE ICRA*, 2006, Orlando , Florida, USA

[14] A. Davoodi, P. Fazli, P. Pasquier, and A. K. Mackworth. "On multi-robot area coverage," *JCCGG*, 2009, Japan.

[15] K. Williams and J. W. Burdick, "Multi-robot boundary coverage with plan revision," *IEEE ICRA* 2006, Orlando, FL, USA.

ISBN: 978-988-18210-3-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)