CHAPTER 4

MULTI-ROBOT TERRAIN EXPLORATION AND COVERAGE BY CONSTRUCTING MULTIPLE SPANNING TREES

4.1 Introduction

In this chapter, online scheme is proposed to explore and cover a continuous bounded terrain by multiple robots for tasks such as floor cleaning, lawn mowing, field de-mining and humanitarian work. We use the multiple ant-type robots which have the advantages of both fault tolerance and parallelism. This work emphasizes the importance of achieving on-line coverage, based on the robot's sensors. The goal of this work is that, all the ant-type robots must completely cover the obstacle free location closer to them simultaneously in a distributed manner, while minimizing the metrics such as travel distance and time. This proposed online scheme is based on Spanning Tree Coverage (STC) [Gabrielly & Rimon, 2001; Hazon & Kaminka, 2005].

4.2 The Model and Assumptions

The robots have no prior knowledge of the terrain except the initial position. They must use their on-board sensors to detect obstacles and markings, and construct their own spanning trees for the grid, representing the terrain. The terrain is divided into square shape grid cells and the size (2D) of the cell is determined by the sensor range (r sensor) of the robot. When a robot is in the centre of the cell, it must be able to cover the entire area with its
sensor. The size (2D) must be equal to $\sqrt{2} \text{ (r sensor)}$. It is also assumed that the robot’s range sensor is capable of identifying the obstacles in the three orthogonal neighboring cells of the robot’s current cell.

During the spanning tree construction, the robot subdivides every cell it encounters, into four identical sub-cells of size $D$, each being identical to the tool size. Every robot runs the algorithm individually to make separate spanning tree during exploration process. Each robot’s initial cell location is given as input to the algorithm, and output is the spanning tree of that robot. The robot may scan orthogonal neighbors (up/down, left/right) in counter clockwise direction starting from its parent cell as shown in Figure 4.1. The robot moves to the right side of the spanning tree and finally reaches the starting sub cell position.

![Figure 4.1: Cells Sensing Directions](image)

It is assumed that when this scheme is implemented in physical robot, the robot has the ability to leave marks in the sub-cells it covers irrespective of the particular nature of the marks, as long as the marks allow the robot to identify sub-cells that have been already covered. There must be a detection device in the physical robot, capable of inspecting the sub-cell marks in the current cell and its three immediate neighbors. To keep track of the marks created by robots in the terrain, in this simulation program, we use the matrix called *visited* and is updated by all the robots simultaneously. After scanning of orthogonal neighboring cells, obstacle-free explored unvisited cells are
pushed onto a stack $S$, later they are popped out one by one from the stack and marked as visited in our simulation program.

The robots, used in this experiment, are homogeneous and all cells in the terrain are accessible from any starting cell. It is to be noted that the execution of the algorithm is completely decentralized, as each robot executes its own independent copy of the program, concurrently. Starting from the respective initial robot’s positions, each robot independently constructs its spanning tree and moves along the path which circumnavigates its spanning tree along the counterclockwise direction as depicted in Figure 4.2.

![Figure 4.2: Spanning Tree and Robot’s path](image)

The following assumptions are made, as in the previous work [Gabriely & Rimon, 2001], to describe the proposed terrain exploration and coverage schemes:

- The terrain to be covered is a planar, which is continuous, and surrounded by an outer boundary.
- The terrain is partitioned into square shaped cells of size $2D$ and each one consists of 4 sub-cells of size $D$. 
Robots can move continuously, between any two unoccupied adjacent sub cells in horizontal (left/right) or vertical (up/down) direction and the sub cells which are fully occupied with obstacles and boundary cells, are eliminated.

A path of complete coverage is created through which a robot sweeps all areas of seen space in the terrain, in a systematic manner and the robot finishes its movement at the same place (sub cell) where it started.

The robot’s one move, takes a unit time and several robots are not able to occupy the same cell simultaneously, and also they never block each other.

4.3. The Proposed Scheme

The proposed scheme named as Simultaneous Multiple Spanning Trees Construction (S-MSTC algorithm) tries to explore a terrain by incrementally building multiple spanning trees by multiple robots simultaneously, using Depth First Search (DFS) technique. It is assumed that the robots are located in their initial positions (S1...Sk) of the terrain. The challenge is to find $k$ spanning trees for $k$ robots simultaneously such that once all the robots complete making their spanning trees, the entire work-area is covered. Since this is an on line approach, it is impossible to add weights to edges to make true optimal solution (minimum spanning tree). But we found an optimal solution because each robot visits each sub cell only once [Senthilkumar & Bharadwaj, 2008c].

In S-MSTC algorithm, only fully obstacle-free cells were considered and the partially occupied cells were not considered. This scheme did not consider narrow openings at this stage. If one robot fails without blocking the unexplored area in the terrain, the other robots take the responsibility to explore and cover the remaining terrain. A robot can block the unexplored
area for other robots by creating its spanning tree which goes in the middle of the terrain by dividing the terrain into two regions. To eliminate this problem of blocking, the proposed scheme lets the robot to explore and cover the neighboring cells first and expands its covered area. The S-MSTC Algorithm and its input and output are given below:

**Input**: Starting locations of all the robots, and orientation sensor data.

**Output**: \(k\) individual spanning trees for each robot.

In the following algorithm:

- \(x\) - Current cell,
- \(p\) - previous cell,
- \(i\) - initial location of the robot,
- \(S\) - stack

Visited - a matrix that contains '1' for cell which is visited by the robot and '0' for unvisited cell, to represent the “markings” in the terrain.

Parent - a matrix in which previous cell \(p\) of the current cell is stored to keep track of the path.

**Algorithm**: Simultaneous Multiple Spanning Trees Construction (S-MSTC)

\[
\text{DFS} (i)
\]
1. Create a stack \(S\)
2. Push cell \(i\) onto \(S\)
3. While (stack is not empty) do
   4. Pop a cell address from \(S\), make it as current cell \(x\)
   5. Lock matrices (visited & parent)
   6. If \((x\) is not covered) do
      7. Visit the cell \(x\) and mark it as visited
      8. Update parent using visited cell’s previous cell \(p\)
   9. While (orthogonal neighboring cells to current cell \(x\) exist) do
      10. If (orthogonal neighboring cell to cell \(x\) is free & not visited) do
          11. Push the neighboring cell onto \(S\)
      End if
   12. End while
13. End if
14. Unlock matrices (visited & parent)
15. End while
For simultaneous process:
Main Function ()

Begin
Create threads for each and every robot
Input initial location for all robots
Call DFS (i) function for all the robots separately
Stop running threads

End

4.3.1. Properties of the S-MSTC Algorithm

The terrain in which the multiple robots work, is represented as 2D grid of square cells and each cell is composed of four sub cells. These large cells are fully or partially occupied. The size of the robot is equal to the size of a sub cell and the robot can move diagonally to any adjacent unblocked cell. Such a move is assumed to take a unit time for all the robots. One cell can be occupied by only one robot at a time. We define a repetitive coverage of a sub-cell as the situation where the sub-cell is being covered by a robot, and subsequently the same sub-cell is covered by the same or another robot. The scheme explores and covers the sub-cells of any cell in a counterclockwise direction. The robot moves the covering tool along a sub-cell path which locally follows the right-hand side of the spanning-tree as shown in Figure 4.3.

The robot enters the new neighbor of the current cell C in counterclockwise order. This property ensures that the covering tool can always gain access to a new neighboring cell of the current cell C through empty sub-cells of C which follow the right-hand side of the spanning tree. If current cell C has new neighbors, then robot selects the first neighbor in counterclockwise order, starting with the parent cell P of C. The covering tool (robot) next follows the right-side of the spanning tree to a sub-cell of C which allows it to exit into the new neighbouring cell N. As depicted in Figure 4.3.
(a-g), motion along the right-side of the spanning tree implies that the tool moves through sub-cells of C in counterclockwise order. If a current cell C has no new neighbors, then the tool circumnavigates the sub cell in the current node in counterclockwise order as depicted in Figure 4.3(h). All cells are covered by the counterclockwise pattern, which always starts with a sub-cell adjacent to the parent cell and will thicken the visited cell region. This feature is crucial for our scheme, which relies on this fixed coverage pattern in order to uniquely identify the parent cell using sub-cell marks.

4.3.2 The Robot Behaviors

Behaviors are used to control the robot during the exploration and coverage task. Four behaviors are identified: cell-selection, boundary-identification, collision-avoidance and move.

1. Cell-Selection - The scheme receives its current location and neighbouring free cell information and then target cell is selected by the scheme by using this behavior. The scheme selects the nearest unvisited free cell first by using the heuristic, that is the robot should
explore and cover the close neighboring cells first and expands its covered area.

2. Boundary-Identification - The terrain boundaries are treated as static obstacles and identified by using this behavior. Then the robot will try to avoid collision with them by using collision avoidance behavior.

3. Collision-Avoidance - If any obstacle or boundary is identified, this behavior will be activated to avoid collision. And appropriate cell will be selected to move next by using Cell selection behavior.

4. Move – The robot moves to the target cell through the sub-cells by circumnavigating the spanning tree by using this behavior.

4.4 Simulation Results

In this section, simulation results are given by using snap shouts of the simulator outputs. The dimension of the terrains are 20 x 20, and 30 x 30 grids size map containing several obstacles to pose a challenge to the S-MSTC algorithm. Figure 4.4 depicts the simulation output of the terrains used for experiments. The ant-type robot could move to each of the three orthogonal neighboring cells of its current cell (Right, Front and Left) provided that the next cell is unoccupied and unvisited. Cells are not traversable if they contain either walls or obstacles. The terrain is transferred to the simulation environment and the results of the proposed schemes are compared to existing grid based approaches.

Figure 4.4: Different Types of Terrains

a. Obstacle free Environment
b. Indoor Environment
c. Outdoor Environment
S-MSTC algorithm has been implemented in our own simulation program using the C++ programming language in Linux operating system. The exploration and coverage task is distributed among multiple robots simultaneously. Robots were assigned initial positions and permitted to create the individual spanning trees concurrently. The spanning trees reflect the sensing ability of the robots and shape of the terrain. Simulation outputs of S-MSTC algorithm are shown in Figure 4.5, where in one experiment two robots and another experiment four robots are used. It is noticed that S-MSTC algorithm tries to avoid blocking other robots to do their task. So if any robot fails and stops doing the task the other robot close to that robot explores and covers the rest of the area in the terrain.

![Figure 4.5: Simulation outputs of S-MSTC Algorithm](image)
4.5 Analysis of the S-MSTC Algorithm

The proposed S-MSTC algorithm is assessed by showing that this scheme is complete, robust and optimized. Collision is a common problem with multiple robots. Since this scheme incorporates collision avoidance behavior, and no spanning tree crosses the other robots' spanning trees, there is no collision among the robots. There is the possibility of reducing the coverage time by using multiple robots to do the exploration and coverage task simultaneously. The proposed scheme is implemented in parallel by using homogeneous robots to ensure the optimal efficiency. It is assumed that in each time frame, each robot moves one step at a time. The algorithm strives to embed Hamiltonian cycle for each robot, having the largest number of cells in the given terrain. The resulting Hamiltonian cycle (union of $k$-spanning trees) is automatically an optimal covering path in terms of path length, since the tool covers every cell exactly once.

4.5.1 Complete Coverage

The proposed scheme generates $k$ spanning trees that explore and cover every cell which is accessible from the respective starting cell ($S_1...S_k$). It is assumed that every unblocked sub-cell is part of an unblocked large cell. Every obstacle-free cell in the terrain is represented by a node in the graph. When the robot constructs the spanning tree using this scheme, every sub-cell touches the spanning tree either at a point (a leaf node), or along a segment (an edge-segment originate from a node). Since every sub-cell touches the spanning tree, the sub-cell path, generated by circumnavigating the spanning tree, passes through every sub-cell of the unblocked cells from cell $S$. When each unblocked cell is completely covered once, and its four sub-cells are visited by the robot, then all cells accessible from cell $S$ are covered. If there is no robot failure, the scheme makes sure that the robots visit all these cells only once. As a result, each robot creates its own optimal spanning trees. When all
these spanning trees get accumulated and become one then the complete coverage of the terrain is achieved.

4.5.2 Robust Coverage

Robustness means, even if only one robot remains in operation, it will be able to carry on and complete the terrain exploration and coverage task. The scheme guarantees that the coverage will be completed in finite time, even with up to \((k-1)\) robot-failures. The creation of spanning trees is an online process for \(k\) robots in this approach. If one robot fails, the rest of the robots \((k-1)\) will cover the terrain. In this scheme, the robot gets to know whether the adjacent (up/down, left/right) grid cells are visited, by marks, left by the visited robot on the cell. So, if any robot that has not visited any grid cell will be visited by the other robot. Sometimes, it may not be an optimized solution, but the scheme will definitely cover the whole terrain. We do not need to reconfigure the group of robots every time, when a robot fails. Also it does not matter which robot failed or how many robots failed at the same time. Hence, this scheme is robust to catastrophic failures, where robots fail and can no longer move. This result relies on an assumption that robots which fail do not block live robots.

4.5.3 Optimal Coverage

We wish to prove that this scheme does not repetitively cover any sub cell unnecessarily. The robots move their tool from one sub-cell to an adjacent sub-cell in the direction, at right angles to the edge, common to both sub-cells. Since each sub-cell is identical to the tool, in shape and size, there is no repetitive coverage movement between adjacent sub-cells. In addition to this, the construction of any two spanning-tree edges which do not originate from the same cell lies at least in two sub-cells apart. The curve that is circumnavigating the spanning-tree cannot cross the same sub-cell twice. As a result, in general the robot tool cannot unnecessarily visit any sub-cell twice
and no repetitive coverage is possible. Therefore, the robot cannot visit the same sub cell more than once, which makes an optimal coverage.

To analyze the number of steps required to complete the coverage, the running time is defined as the maximum number of steps that each robot has to go [Wagner et al, 1999]. Hence running time is calculated by using \( \text{max}(\text{step}(i)) \), where \( \text{step}(i) \) is the total number of steps taken by robot \( i \in k \).

![Figure 4.6: Best Case Scenario with Two Robots](image)

**Best Case [Wagner et al, 1999]:** The best running time for the scheme is \( 4\left\lfloor \frac{n}{k} \right\rfloor - 1 \), where \( n \) is the number of grid cells and \( k \) is the number of robots. The best-case is when the starting positions such as \( S_1, \ldots, S_k \) place the robots at almost equal distance from each other. It is proved that the running time is critically dependent on the initial positions of the robots. The Figure 4.6 shows the actual situation and the best running times for different number of robots are tabled in Table 4.1.

**Worst Case [Wagner et al, 1999]:** The worst running time for the proposed scheme is dependent on the number of robots and their initial positions. The worst-case scenario is where generally all the robots start at cells next to each other, or adjacent cells. If any robot blocks the growing process of the other robot’s spanning tree then the situation will become worst. Some typical examples are depicted in the Figure 4.7 and the worst running times for different number of robots are tabled in Table 4.1.
Chapter 4

Running time of S-MSTC algorithm is compared with the existing algorithm such as Non-Backtracking MSTC (NB-MSTC), Backtracking MSTC (B-MSTC). In NB-STC and S-MSTC algorithms, each robot simply circumnavigates its tree until it reaches its initial small cell, but in B-MSTC, each robot that has covered its cells backtracks until it reaches its initial small cell. Hence, in all cases, the robots continue to move around the tree(s), extending the original multi-robot coverage algorithms in a very simple way. In all STC and MSTC algorithms, the construction of spanning tree is done first and later it is divided up into sections, and assigned to each robot. Hence, the output travel cost is unbalanced. But in the S-MSTC algorithm, spanning trees are created in parallel by each robot, and the travel cost unbalanced problem is overcome.

All the MSTC schemes are evaluated in different scenarios, namely different kinds of terrains, varying number of robots. They made the following observations: The cover time for best case increases with the number of robots.
for all schemes. A comparison with the worst case, implying that all robots are starting at the cells which are close to each other, shows that on the completion of the S-MSTC shows an improved result. However, when interpreting these results, we need to keep in mind that the cover time for all MSTCs depends on the initial position of the robots. A simulation out for this situation is depicted in Figure 4.8, here an indoor environment and it is explored by three robots which start from different initial positions. The result demonstrates that the initial position of the robot within the terrain can seriously affect the coverage time. But we found that the proposed S-MSTC algorithm produced improved worst case results compared to other STC based algorithms.

Figure 4.8: Three Spanning Trees and Different Starting Locations

<table>
<thead>
<tr>
<th>Number of Robots (k)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB-MSTC Best case</td>
<td>119</td>
<td>59</td>
<td>39</td>
<td>29</td>
<td>19</td>
</tr>
<tr>
<td>NB-MSTC Worst case</td>
<td>119</td>
<td>117</td>
<td>115</td>
<td>113</td>
<td>109</td>
</tr>
<tr>
<td>B-MSTC Best case</td>
<td>119</td>
<td>59</td>
<td>39</td>
<td>29</td>
<td>19</td>
</tr>
<tr>
<td>B-MSTC Worst case</td>
<td>119</td>
<td>79</td>
<td>59</td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>Best case</td>
<td>119</td>
<td>59</td>
<td>39</td>
<td>29</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>Worst case</td>
<td>119</td>
<td>105</td>
<td>69</td>
<td>69</td>
</tr>
</tbody>
</table>
### Table 4.2
Total Number of cells Covered by each Robot in Different Terrains

<table>
<thead>
<tr>
<th>Terrain Type</th>
<th>Starting positions of robot #</th>
<th>Episode 1</th>
<th>Episode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obstacle free Terrain</td>
<td>(1,1)</td>
<td>(20,20)</td>
<td>(20,1)</td>
</tr>
<tr>
<td>Obstacle free Terrain</td>
<td>140</td>
<td>165</td>
<td>140</td>
</tr>
<tr>
<td>Indoor with obstacles</td>
<td>145</td>
<td>163</td>
<td>146</td>
</tr>
<tr>
<td>Outdoor with obstacles</td>
<td>148</td>
<td>130</td>
<td>143</td>
</tr>
</tbody>
</table>

The proposed S-MSTC approach is simulated in different environments with and without obstacles for four robots. The Table 4.2 shows the results for the same initial locations of robots in three different terrains (20x20). It is found that the initial position of the robots decides the number of cells, which can be covered by each robot. Here episode 1 is one set of initial positions of the robots and episode 2 is another set of initial position of the robots.