3.1 Introduction

Robots are widely used in domestic and commercial applications, typically involved with a specific task. Autonomous exploration and mapping become increasingly strong on single robots, but the next challenge is to extend these techniques to multiple robots. The coordinated multi robots exploration is a fundamental problem in robotics. The use of multiple robots is often suggested to have several advantages over single robot systems [Parker, 2000]. Many Terrain Coverage Planning (TCP) algorithms use a cellular decomposition of the free space, which is discussed in chapter one. TCP algorithm enables applications such as scouting and cleaning [Dlouhy et al., 2000], robotic demining [Nicoud & Habib, 1995], and lawn mowing [Chandler et al., 2000].

This chapter discusses about how coordination among the robots is achieved by assigning targets through GA technique. Moreover, for fitness calculation, it uses two parameters: the utility of unexplored areas and the cost for reaching these areas. This cost and utility method has been integrated already in [Burgard, 2000]. But in this proposed scheme, it has been defined some specific points as targets in the occupancy map [Yamauchi, 1998]. Many classical path planning algorithms and some coverage approaches assume that
the robot knows the layout of the environment prior to the planning event. One benefit of map-based approaches is that a path can be generated more efficiently using a map. This work considers TCP technique as an approach that determines a path for a robot to pass over all free points in the partially known plane space. The robot has the capability of exploring the environment on its own.

The key problem to be solved in the context of multiple robots is to choose appropriate target points for the individual robots so that they simultaneously explore different regions in the environment. Here Genetic Algorithm (GA) is used to assign the targets for the individual robots through which the coordination within multiple robots is achieved. A combination of cost and utility methods is used for fitness calculation. The cost means the cost of traveling the distance to the target and the utility of a cell means the size of the unexplored area that a robot can cover with its sensors upon reaching the target location or near by location. The terrain coverage technique is used to cover the visible environment by the robot. The proposed scheme is implemented and tested using simulation. It is demonstrated that this scheme is well suited to control the motions of a team of robots in various partially known environments by using different experiments.

3.2 Genetic Algorithm

A new approach in robotics is evolutionary robotics, which uses evolution as a tool to create better robot controllers. Genetic Algorithms (GAs) have shown to be powerful procedures for solving complex and multi criterion optimization problems [Goldberg, 1989]. These algorithms mimic models of natural evolution, having all the benefits of a population based search method. Population represents a solution and each chromosome of the population is given a fitness value computed by an evolution function. When all the chromosomes in the population have been evaluated, reproduction, crossover,
and mutation operators are applied to create a new population. In all generation, two best parents are selected, and one-point crossover is performed with a given probability. Then, values of the newly obtained strings are mutated with a given probability. Thus, in this evolutionary system, the new population replaces the old one maintaining only the better offsprings. After a number of generations, this process terminates when the best chromosome that represents the near-optimum solution is found.

There are many differences between GAs and traditional search algorithms; one, traditional search algorithms work with one solution at a time while GAs work with many solutions. Two, traditional algorithms are often deterministic while GA is probabilistic in nature. GA has some weak points as well. They are very slow when compared with other traditional algorithms, and also there is no guarantee that the solution is efficient enough. Considering the advantages and weaknesses, one may conclude that GAs can be of greater value to those problems for which there is no optimized search algorithm.

This work presents a new scheme to the Target Assignment Problem (TAP) that describes a GA controller system which evolves chromosome with target points. These points define the target location to each robot. The size of the chromosome is determined by the number of robots involved in the process. In this experiment, four robots have been used hence the size of chromosome is four. The chromosome contains target for each robots as values, and the first gene is assign to robot #1, and second gene is assign to robot #2 and so on. The chromosome structure is depicted in Figure 3.1. In each generation a fitness value is assigned to each chromosome.

\[
\begin{array}{c}
1 \\
3 \\
5 \\
7
\end{array}
\]

Figure 3.1: Chromosome Structure
3.3 The Proposed Scheme

The goal of an exploration process is to cover the whole environment in a minimum amount of time. It is assumed that at every point of time both, the map of the area explored so far and the positions of the robots in the map are known. To represent the environment, Occupancy Grid Maps (OGM) [Yamauchi, 1998] technique is used. Each cell of an OGM contains a numerical value representing the probability that the corresponding area in the environment is covered by an obstacle.

Genetic Algorithm technique is used to coordinate the robots by determining appropriate target locations to each robot. By directing a robot to an unexplored target cell, one can expect that it gains information about the unexplored area when it arrives at its target location. In this scheme several robots will not move to the same target location and robots do their exploration independently. The robots are simple with limited sensing and computational capabilities. Cost of reaching a target cell and the utility of that cell are used to calculate the fitness value. For each robot, the cost of a cell is proportional to the distance between the robot and that cell. The utility of a cell depends on the number of robots that cover the cell.

3.3.1 Cost Calculation

The cost of reaching the current target cells is determined by calculating the optimal path from the current position of the robot to all cells. This is based on a deterministic variant of value iteration, a popular dynamic programming algorithm. As mentioned above, the cost for traveling to a grid cell is proportional to its occupancy value. The cost calculation is done independently for each robot. The steps used to calculate minimum-cost path is as follows.
Chapter 3

Initialization Step:

\[ V_{x,y} = \begin{cases} 
0, & \text{if } (x, y) \text{ is Robot's position} \\
\infty, & \text{otherwise}
\end{cases} \]

Updating step:

For all grid cells \( C_{x,y} \) do

\[ V_{x,y} = \min( V_{x+\Delta x,y+\Delta y} + \sqrt{(\Delta x)^2 + (\Delta y)^2} \cdot p(C_{x+\Delta x,y+\Delta y})) \]

Where \( \Delta x, \Delta y \in \{-1, 0, 1\} \),

\[ p(C_{x+\Delta x,y+\Delta y}) \in [0, \text{occ}_{\text{max}}] \]

The maximum visibility range of the robot is assumed as 2 grid cell positions, so the value for \( \text{occ}_{\text{max}} \) is assigned as 2. This technique updates the entire grid cell’s cost \( (V) \) by the value of their best neighbors, plus the cost of moving to these neighbors. Cost is equivalent to the probability \( p(C_{x,y}) \) that a grid cell \( C_{x,y} \) is occupied times the distance to the cell. The computation of \( V \) is done separately for each robot.

3.3.2 Utility Calculation

In this work, it is assumed that if there is a robot that moves to a particular target cell, the utility of that cell can be expected to be lower for other robots. Since sensors of a robot cover the terrain around a target cell as soon as the robot reaches there, the expected utility of those cells in the locality of the robot’s target point is also reduced. Initially each cell has the same utility values. Whenever a target point is selected for a robot, the utility of the adjacent cells \((a)\) in distance \(d\) (2 grid cells) from the target cell \((t)\) is reduced, according to the probability \( P_{\text{vis}}(a,t) \).
Chapter 3

The formula for the Utility Calculation is as follows:

\[ U_i = U_i - p_{\text{vis}}(a,t) \]

A linear function is used to represent \( p_{\text{vis}}(a,t) \) which is given below:

\[
p_{\text{vis}}(a,t) = \begin{cases} 
1 - \frac{\|a - t\|}{\text{dis} \tan \text{ce}}, & \text{if } \|a - t\| < \text{dis} \tan \text{ce} \\
0, & \text{otherwise}
\end{cases}
\]

Where the distance means the maximum distance that the robot can sensor from the target position.

3.3.3 Target Selection Process

The targets are assigned to each robot using GA technique with the cost and the utility which have been considered separately for individual robots. To decide the assignment of target for robots, an iterative approach using GAs is used. In each generation, initially the targets are randomly assigned to all the robots and the fitness for that particular chromosome is calculated by using the following equation.

\[
\text{fitness} = \sum_{i=1}^{n} U_i - V_i
\]

Where \( V_i \) is the cost of traveling by robot \( i \) to target \( t \) and \( U_i \) is the utility of target \( t \) which is assigned to robot \( i \). Number of robots are \( n \). The GA terminates when the best fitness is achieved for a set of targets. This scheme puts the constraint which assures that there is no duplication of target assignments.
This process is repeated until the whole environment is covered. The algorithm is as follows:

1. Calculate the cost \( V \) for reaching the entire cells in the environment by all robots separately.
2. Initialize the utility \( U \) for all the cells to 1.0.
3. While (there is some more unexplored area) do
4. Determine the target for all robots by using GA
5. Reduce the Utility of all target points and the adjacent points in the covered area from the targets of all the robots.
   \[ U = U - p_{vis}(a,t) \]
6. End while

3.4 Cell Exploration

In order to explore a cell, the robot must enter into it. Once robot is in a cell, it knows which of the four orthogonal neighboring cells are available, and which has obstacles or boundary. Each cell in the robot’s covered area must be visited at least once. The robot moves forward using left hand rule. That is, from each cell that is visited, the robot tries to move to the adjacent unexplored cell, following a left turn over a forward step over right turn. When ever all 4 adjacent cells are explored or blocked, robot will try to return along the path until it comes across an unexplored cell. This process will continue until all the cells in the covered area are explored. For example, Figure 3.2(a), 3.2(b), and 3.2(c) depicts the way in which a robot explores the area. In the environment, ‘R’ indicates the robot’s position and ‘O’ indicates the obstacle position.
3.5 Implementation and Results

This scheme has been implemented by using visual C++ language and performed simulation experiments in a Pentium M machine. Four robots are used, starting in different arbitrary locations in the environment with 12 targets. Target are predefined and decided in a way they do not appear in the current visibility range of the robots. The experiment is designed to demonstrate the capability of proposed scheme to efficiently cover the environment with multiple robots. Figure 3.3 shows the map of the environment with targets. The size of the environment is 15x20 grids and is represented by an occupancy grid map.

Figure 3.2: Cell Exploration Directions

Figure 3.3: Robots’ Initial Positions and Targets
The experiment was carried out in an environment without obstacles and with static obstacle (a wall) in two different environments. It is noticed that this scheme can efficiently guide the multiple robots to coordinately explore the environment. Figures 3.4 and Figure 3.5 depict the map of the environment after some time of exploration and the final map respectively.

This approach prevents robots to explore the place that are already explored by other robots. Since each cell is identical and the robot visits every cell, the resulting cell path completely covers the work-area grid in optimal
time $O(N)$, where $N$ is the number of free grid cells. This scheme has been implemented and tested on different simulated environments. Experiments show that this algorithm is capable of coordinating multiple robots during exploration in an efficient way. Despite these encouraging results, there are several aspects which need to be improved. It is assumed that the environment is partly known with static obstacles. In many situations, this assumption may be unrealistic. Instead, the robot must use its on-board sensors to acquire information about the dynamic environments with moving obstacles and perform coverage on-line.