A Distributed Strategy for Human-in-the-loop Task Servicing using Multiple Robots with Stationary Base Station Connectivity Constraint

Student Name: Parikshit Maini

IIIT-D-MTech-CS-12-014

August 17, 2014

Indraprastha Institute of Information Technology

New Delhi

Thesis Committee

Dr. P. B. Sujit (Advisor)

Dr. Rahul Purandare (Internal)

Dr. Mangal Kothari (External)

Submitted in partial fulfillment of the requirements

for the Degree of M.Tech. in Computer Science

Keywords: multi-robot systems, multi-agent systems, connectivity constraint, network connectivity, task allocation, base connected, networked robotics, human-in-the-loop, stationary base station

Certificate

This is to certify that the thesis titled **"A Distributed Strategy for Human-in-the-loop Task Servicing using Multiple Robots with Stationary Base Station Connectivity Constraint"** submitted by **Parikshit Maini** for partial fulfillment of the requirements for the degree of *Master of Technology* in *Computer Science & Engineering* is a record of the bonafide work carried out by him under my guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. P. B. Sujit

Indraprastha Institute of Information Technology, Delhi

Abstract

Mobile robots are increasingly being used for tasks like remote surveillance, sensing and maintenance. Some of these tasks are critical and require intelligent decision making for successful completion. It is not always possible to rely exclusively on robot level intelligence to make high impact decisions and hence human supervision is needed during task execution. To facilitate *human-in-the-loop* task servicing, the task executing robot is required to remain connected to a remotely located human operator.

However, robot communication range is typically limited and hence multiple mobile robots might be deployed to perform the tasks. These robots must coordinate with each other to dynamically form and maintain a communication link such that network connectivity exists between the robot servicing the task and the human operator positioned at a stationary base station.

The development of connectivity aware coordination algorithms is complex due to limited communication range and presence of obstacles in the search region. In this thesis, we present a distributed multi-robot algorithm for task servicing with human-in-the-loop constraint. Robot control and mission execution is independent of the human operator and is fully autonomous. The algorithm facilitates indirect collaboration amongst the robotic agents and uses a combination of graph theoretic and gradient descent based approaches for path planning. Robots exercise independent decision making on task and role assignment by following a self allocation strategy. This allows dynamic task reassignments and role exchanges amongst the agents based on increased situational awareness. Our solution

successfully implements obstacle avoidance and deadlock resolution while being scalable and robust to network and robot failures. To substantiate the claims, we present results from extensive simulations.

Acknowledgments

This work would not have been possible without support from a number of people. Foremost, I would like to extend my deepest gratitude to Dr. P. B. Sujit for his expert guidance and for the extremely productive brainstorming sessions I had with him. I am grateful to my friends and lab colleagues who time and again offered fresh perspectives on my research. I am also thankful to IIIT-Delhi for providing excellent infrastructure and support. Last but never the least, I am immensely grateful to my parents, family members and close friends, for their invaluable support and unconditional love.

Parikshit Maini

Dedicated to,

mummy and papa

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots have come of age! From industrial automation to house cleaning, war site reconnaissance to medical surgeries; robotic systems find application in various domains. In recent times, multiple robots are used in outdoor environments to collaboratively execute tasks like radiation sensing, surveillance and search and rescue. In these applications, it is not possible to rely exclusively on robot level intelligence as crucial decisions with potentially high impact are involved. Hence, a human operator overseeing the operation is necessary to make these decisions. The robot servicing the task (also called service robot/agent) must have a continuous connection to the base station to enable human supervision of the task. In such scenarios the human operator typically administers the mission from a remotely situated base station. Hence the problem of human-in-the-loop task servicing is interpreted as the requirement of a communication link between the service robot and the base station. The human operator is not involved in mission planing or robot control. Her role is limited to overseeing and/or using robot sensors/actuators during task execution, which is not the subject of this work.

Communication between the service agent and the base station can be established using

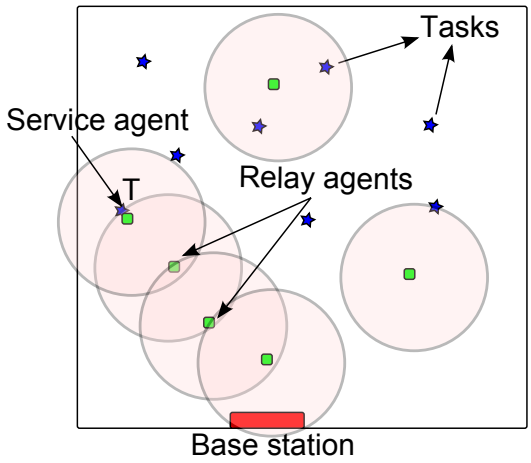


Figure 1.1: A sample scenario

radio or Wi-Fi, but the range is limited. Also the presence of obstacles pose line-of-sight issues. Satellite communication using iridium can be used but it is expensive and has limited bandwidth. Thus, multiple robots with limited communication range must be deployed to collaboratively service the task. The robots would need to coordinate with each other to act as communication relays (also called relay agents) for ensuring connectivity between the service agent and the base station.

We consider an application scenario where tasks appear randomly in a bounded region (as shown in Figure 1.1) which need to be executed by the robots under human operator supervision. Deployment of a static network is not feasible in many remote regions and hence a multi-hop network must be formed by the robots to ensure the operator is connected to the task servicing robot. The topology of the robot network changes dynamically as more tasks are executed and when new tasks appear. The robots have to shuffle between the roles of service agent and relay agent or sometimes assume both the roles. The environment is inherently unknown in these missions and hence pre-planning and feeding way-points to the robots is not an option.

The multi-robot task servicing problem can be modeled as a variation of the multi-robot routing problem (known to be NP-hard [1]) with base station network connectivity imposed as a constraint. However, for this virtual targets must be determined for robots to make the communication links which is complex and computationally intensive and requires a planner with global situation awareness.

In this thesis we present a distributed multi-robot algorithm based on graph-theoretic and gradient potential based approaches. The developed algorithm has the following novelties:

- The agents implicitly coordinate with each other to assign roles and tasks. There is no explicit coordination amongst agents for task and role assignment or switching.
- The developed algorithm allows Robots to perform self allocation of tasks and roles with dynamic switching of service agent and relay agent roles to create a network connecting the base station and the service agent. The concept of self allocation is inspired from natural swarms [2] and allows the robots to resolve deadlocks such as a head-on engagement between two robots.
- A Robot is able to exchange and transfer tasks to its neighbours to share the burden or improve performance when the neighbour has a lower cost to service the task.
- There are no leaders at local or global level and every agent performs its role without knowledge about the state of another robot.
- The system is robust to faults as the network does not depend explicitly on some agents.

Chapter 2

Related Work

Network connected multi-robot control is an active field of research traditionally. Exploration and target tracking have been the mainstay of multi-robot research. Most of the current literature in network connected mission planning for multiple robots is also concentrated in these domains [3] [4] [5] [6] [7]. Vazquez and Malcolm [3] for instance, developed a behavior based scheme for multi-robot exploration under connectivity constraints, to maintain an ad-hoc network between the robots during exploration. Their work does not consider the effect of obstacles in the environment on communication range. Ulam and Arkin [4] developed a suite of high level reactive behaviors to recover from communication failures during environment exploration. Pandey et al. [5] have also performed multi-robot planning with communication constraint in space exploration context. They have addressed the issue of connectivity to a mobile base station, where one of the agents is assigned as the base station. To ensure connectivity to the mobile base station, their work involves generating robot positions upfront and the robots then move synchronously to these locations. It is not always possible to have perfect synchronization between the robots, which could lead to network failures under this scheme. Network Connectivity in target tracking applications has been analysed by [6] and [7]. Hollinger and Singh [6]

developed a scheme for cooperative target tracking with periodic connectivity between the tracking vehicles. Gans et al. [7] developed a feedback guidance law to ensure connectivity in a multi-UAV system for cooperative target tracking. Reich et al. [8] designed a scalable algorithm for iRobot nodes for area coverage. Their algorithm aims at physical layer connectivity and the robots freeze at their locations when risk of further movement endangers the connectivity. Michael et al. [9] experimentally demonstrated several network constrained control laws for robots. These works consider connectivity issues within the group of robots to allow sharing of information for situational awareness and/or for formation control. However, they do not impose a stationary base station connectivity constraint on robot motion planning.

There are few research contributions related directly to the problem of connectivity to a stationary base station. Pei et al. [10] modeled the base station connected exploration problem as a variant of the Steiner Minimum Tree problem to determine the nodes that will provide relay for a multi-robot system. They use a centralized planner that optimizes on the total system cost for change in the position configuration of the robot team. To ensure base station network connectivity in multi-robot systems in a task servicing context [11] and [12] have also used centralized planners. Robot locations are precomputed to form a Minimum Spanning Tree (MST) to the task location(s) in these schemes. Maini and Sujit [13] developed a dynamic team building strategy to maintain robot connectivity in a task servicing setting. They perform dynamic role assignment to form service teams. While robot trajectories are generated individually by the robots, role allocation is done by a central planner. Strategies with centralized planners that precompute robot path do not capture the dynamicity of the environment and increase network traffic to propagate situational awareness to the central controller. Also the central entity acts as a single point of failure for the entire system.

Recursive auction based algorithms such as $S + T$ by Viguria et al. [14] have also been used to solve the multirobot task allocation problem for surveillance tasks. Ponda et al. [15] used CBBA [16] for distributed planning in multi-robot missions to ensure network connectivity. Their strategy utilizes free agents as relays by generating relay tasks. However to create the relay tasks, a central planner needs to be actively involved in the task servicing mechanism. Team based schemes such as [15] and [13] lack the capability to dynamically restructure team sizes. When servicing multiple simultaneous tasks increased situational awareness may create the need to reassign robot teams, to reassign roles, to add robots in the middle of a network chain and to exchange tasks between the robot teams. These tasks increase the complexity of the coordination algorithm many fold and make it computationally intensive. Hence a strategy that allows dynamic team building with flexibility to add/remove members, reassign tasks and roles is desirable.

In this thesis, we present a fully distributed online multi-robot control mechanism to facilitate network connected task servicing. We present a formal problem definition in Chapter 3. We have used a combination of graph theoretic and gradient based approach to determine robot motion, as described in Chapter 4. Chapter 4 also gives a description of the robot model and the communication model used. Each robot uses its local environment perception, sensed and one-hop communication, to make a self role allocation as described in Chapter 5. Also, by switching to *steepestDescent* behavior in case of lost connectivity, the mechanism is robust to network failures and allows robots to reestablish connectivity. Chapter 6 describes the simulation model, results and analysis. Chapter 7 concludes the work and also gives possible future extensions.

Chapter 3

Problem Formulation

Assume that n robots are deployed in a region as shown in Figure 1.1. The base station shown here at the boundary is only for representation and can be located anywhere within the operational region. As shown several tasks are present and need to be carried out remotely using multiple robots, each having a limited communication range (shown as a circle around the robot). Since a single robot cannot communicate the task information to the base station, several robots have to form a networked link that the service agent can use to send data to the base station. A sample link is shown for task T in the figure, where a set of agents form a chain. To form a communication network, some robots need to act as relay nodes while others service the task(s). Considering every robot has the same capabilities, the role assignment should be dynamic such that on-the-fly exchange of roles amongst robots is possible.

The problem can be formulated as an optimization problem, where robots incur a non-negative cost $costMove_{k_1k_2}$ to move from location k_1 to k_2. We have considered $costMove_{k_1k_2}$ to be a linear function of the length of the least weighing path from k_1 to k_2. The computation of path weights is described in the next chapter. Depending on context and needs

of the mission, optimization criterion can vary as given and such others:

1. *MinSum1*: minimize the sum of the total cost incurred by every robot over the entire mission duration.
$$\min \sum_{i=1}^{numRobot} \mathrm{J}_i \tag{3.1}$$

2. *MinSum2*: minimize the sum of time taken (latency) to service all the tasks.
$$\min \sum_{j=1}^{numTask} \mathrm{latency}_j \tag{3.2}$$

3. *MiniMax*: minimize the maximum total cost incurred by a robot over the entire mission duration.
$$\min \max \mathrm{J}_i \tag{3.3}$$

where, J_i is the total cost incurred by robot R_i over the entire mission duration and $latency_j$ is the time taken to service task T_j.

The problem is formulated as:

Optimize objective function as defined above

$$\text{such that :} \quad \forall j,\ \exists\ t \ \ s.t. \ \ G_t(R_i, B) = 1 \tag{3.4}$$

$$\text{and} \ \ ||P_i(t) - L_j|| = 0 \tag{3.5}$$

where, B is the base station, $G_t(R_i, B) = 1$ if there exists a communication link between R_i and B at time t, $||\cdot||$ is the Euclidean distance, $P_i(t)$ is the position of robot R_i at time t and L_j is the location of task T_j.

Chapter 4

System Details

The proposed approach comprises of two design principles (i) robot with the least cost to service a task should be the service agent for the task and (ii) role allocation to a robot aims to minimize the total individual cost incurred by the robot over the entire mission duration. The principle (i) ensures that the time to execute the task is minimized as the nearest agent to the task is allocated. While (ii) ensures the agents make independent decisions to be a relay agent or service agent based on their cost. A control strategy that maintains a balance of the two principles is essential for an effective solution.

4.1 Environment Model

The environment is bounded and represented as a grid. Assuming a cell based decomposition of the configuration space such that the union of all cells is the entire configuration space is a well established decomposition methodology [17] [18]. We have decomposed the environment into a grid of rectangular cells. Cells in the grid are of length l and width w. Center of a cell is referred to as the cell center. Robots can take position only at the cell centers and a cell is fully occupied if a robot is present in the cell.

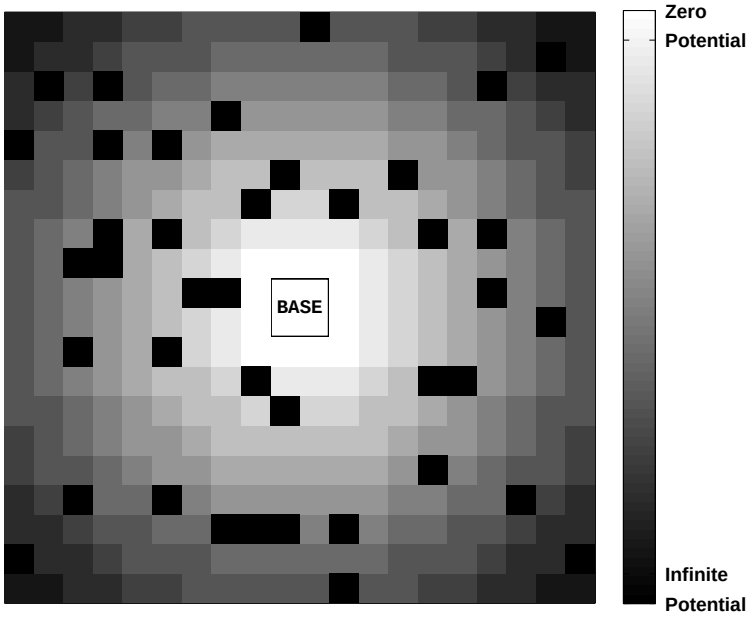


Figure 4.1: Potential Gradient Field

The environment grid is assumed to be a potential gradient field. World boundaries are considered at infinite potential and base station at zero potential as shown in Figure 4.1. The potential is constant over a single cell. The potential at a cell is computed as given in Eqn. 4.1. The cost of moving to a cell is directly proportional to the potential at the destination cell. A move to a higher potential cell costs more, while a move to a lower potential cell costs less than a move between cells at equal potential.

$$pot(k) = floor(dist(x, y)) \tag{4.1}$$

where, (x, y) represent the coordinates of the center of cell k, $pot(k)$ is the potential at cell k, $dist(x, y)$ gives the Euclidean distance of the coordinates from base and $floor(fraction)$ returns the largest integer smaller than $fraction$.

Further, the environment is represented as a weighted directed graph, referred to as $envMap$, with grid cells as nodes and edges representing valid transitions between nodes. Edge weight corresponds to the potential at the destination cell. Cells that are occupied by obstacles or the base station are referred to as obstructed cells and are not reachable by the robots. We assume that the obstacles occupy complete cells. Even if a cell is partially occupied we mark the cell as unreachable. A LOS communication model is used. Thus communication between the robots is not possible if the LOS is obstructed. A collision is said to have occurred if more than one robot is present on a cell or a robot enters an obstructed cell.

4.2 Robot Model

Our robot model considers real world discretization restrictions on movement. *Translation range or step size (r_t)* is the maximum distance a robot can travel in a given direction in unit time. Robots can only move from one cell (center) to another cell (center). *Communication range (r_c)* is defined as the distance within which bidirectional communication between two robots is possible. *Sensing range (r_s)* is the maximum distance up to which a robot can sense its environment for obstructed cells and other robots. A generic environment perception model with omni directional sensing ability within a limited range (r-disc), is considered. The three robot ranges are related as follows:

$$r_c > r_s > r_t \geq max(l, w)$$

All ranges are considered as distances in a 2D Euclidean environment.

4.3 Communication Model

We assume r-disc model for communication range. If $||P_i(t) - P_{i'}(t)|| \leq r_c$ we say that robot R_i and $R_{i'}$ are in communication range. A network connectivity graph of the robots, where nodes represent robots and each edge represents a bidirectional communication link between two robots, is shown in Figure 4.2. We further assume that if R_i and $R_{i'}$ are not in LoS (Line of Sight) then they cannot communicate i.e. two robots cannot communicate if there is an obstacle between them.

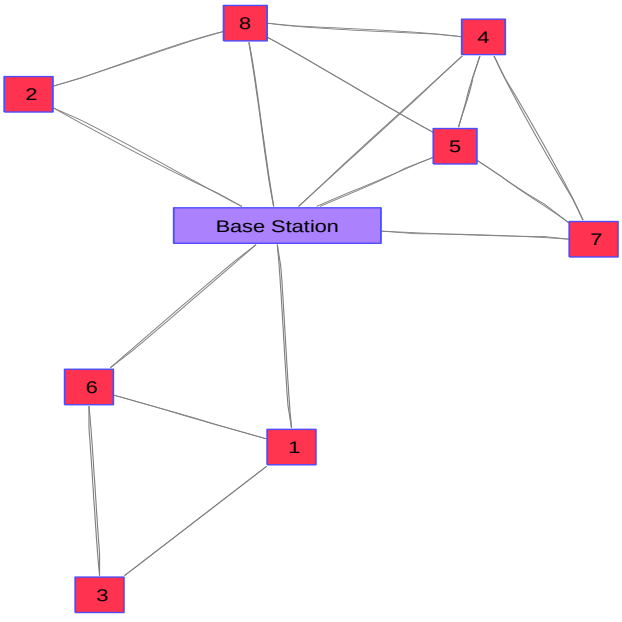


Figure 4.2: Snapshot of the dynamic ad-hoc network

Chapter 5

Strategy

Our system uses agent level rationale to model intelligent group behavior. The underlying idea is to make distributed decisions based on simple rules, using locally sensed information. Every robot is assumed to be an autonomous entity and follows the pipeline shown in Figure 5.1. We will explain the various stages of the pipeline for a single robot and refer to it as the host robot.

5.1 Environment Sensing

Robots can discover other robots within r_c (Eqn. 5.1) and check for one hop connectivity to the base station. Also a robot can detect occlusions and other robots within its sensing range along the Line Of Sight (LOS). Since $r_s < r_c$, it can also communicate with the robots detected within r_s. This information is collated to differentiate between robot occupied cells and obstructed cells. We differentiate between obstructed cells and robot occupied cells as hard (or stationary) obstacles and soft (or mobile) obstacles, respectively.

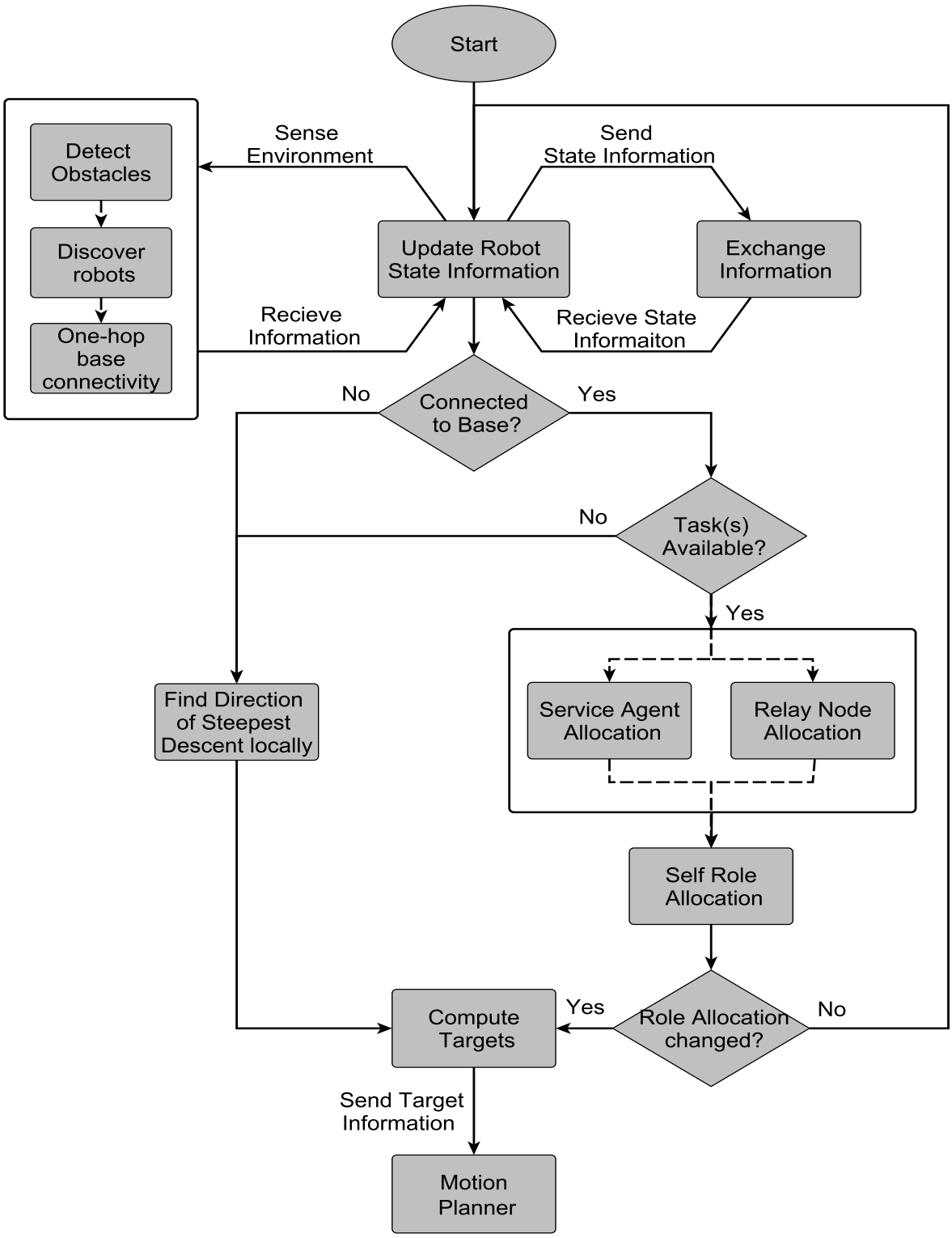


Figure 5.1: Robot level control flow diagram: Strategy Planner

$$agentsInRange_{i'}(t) = \{R_i : ||P_i(t) - P_{i'}(t)|| \leq r_c\} \tag{5.1}$$

Each robot maintains a local copy of the $envMap$ and updates it with it's sensory perception of the environment. Robots can detect obstacles, hard and soft, within r_s. A hard obstacle detection sets the potential of the corresponding cell to infinity (Eqn. 5.2) and also modifies the cost computation for transition to and from those cells. This may be implemented by permanently removing all edges to and from the obstructed cell in $envMap$ (Eqns. 5.3 and 5.4).

$$pot_{i'}(k) = \infty, \forall k \in \{hardObs_{i'}(t)\} \tag{5.2}$$

$$cost_{i'}\{:, k\} = 0, \ \ \forall k \in \{hardObs_{i'}(t)\} \tag{5.3}$$

$$cost_{i'}\{k, :\} = 0, \ \ \forall k \in \{hardObs_{i'}(t)\} \tag{5.4}$$

where, $pot_{i'}(k)$ is the potential at cell k in agents i's local copy of $envMap$.

Communicating robots share their current location $P_i(t)$ and next feasible locations vector $nextLocs_i(t)$ as defined in Table 5.1. Robot $R_{i'}$ for instance, checks which of its neighbours $nextLocs_i(t)$ locations overlap with $nextLocs_{i'}(t)$, and could lead to a possible collision. If a neighboring robot lies in $nextLocs_{i'}(t)$, it is detected as a soft obstacle (Eqns. 5.1 and 5.5). Also locations common to $nextLocs_{i'}(t)$ and $nextLocs_i(t)$ of a neighboring robot could lead to a possible collision and hence are also considered as potential obstacles (Eqn. 5.7). A relevant change in the cost computation of a direct move to obstacle locations is desirable. It may be implemented by appropriately changing the weight of the directed

edge from the robot's current location to the obstacle cell (Eqns. 5.6 and 5.8).

$$softObs_{i'}(t) = \{P_i(t) \cap nextLocs_{i'}(t), R_i \in agentsinRange_{i'}(t)\} \tag{5.5}$$

$$cost_{i'}\{P_{i'}(t), k\} = \infty, \forall k \in \{softObs_{i'}(t)\} \tag{5.6}$$

$$collAvoidCells_{i'}(t) = \{nextLocs_i(t) \cap nextLocs_{i'}(t), \forall R_i \in agentsInRange_{i'}(t)\} \tag{5.7}$$

$$cost_{i'}\{P_{i'}(t), k\} = cost_{i'}\{P_{i'}(t), k\} + w, \forall k \in \{collAvoidCells_{i'}(t)\} \tag{5.8}$$

where, w is the penalty term. The value of w depends on r_t and should be chosen in a way such that the least costing path to any other cell from $P_{i'}(t)$ does not include k.

Soft obstacle and collision avoidance related cost computation changes are temporary and rolled back when the corresponding cells are detected as empty. As environment sensing (and obstacle detection) is along the LOS, there also occur blind spots around obstacles. These are rare occurrences and can lead to collision between robots.

5.2 Information Exchange

The Information exchange stage involves communication within the one hop network around the robot. Communicating robots exchange vectors as given in Table 5.1. When in one hop base connectivity the robots share the following vectors with the base station: $task\_status_{i'}(t)$ and $P_{i'}(t)$. The base station also shares the $task\_status_i(t)$ and $P_i(t)$ of the other one hop connected robots.

$hardObs_{i'}(t)$	an array of hard obstacle cell indices
$cost\_vector_{i'}(t)$	robot's service cost to currently active tasks (1 x numTask)
$task\_status_{i'}(t)$	possible task states: {inactive, active, serviced} (1 x numTask)
$P_{i'}(t)$	robot's current location
$nextLocs_{i'}(t)$	robot's next feasible locations
$baseConnection_{i'}(t)$	shortest connectivity link to the base station

Table 5.1: Robot exchange vectors

5.3 Service Agent Selection

Each robot individually computes the service robot for all currently active tasks.

$task\_costs_{i'}$ (numRobot x numTask) is the robot-task cost matrix where,

$$task\_costs_{i'}(i,j) = C_{ij} \tag{5.9}$$

where, C_{ij} is the cost for robot R_i to service task T_j.

For robots not in r_c, the cost to service a task is ∞,

$$task\_costs(r,:) = \infty, \forall r \notin \{agentsInRange\} \tag{5.10}$$

The robot with the lowest cost to service a task is selected as the service robot for the

task.

$$serviceRobot(j) = \underset{i}{argmin}\{task\_costs(i,j)\}, \forall j, i = 1...numRobots \tag{5.11}$$

In case of a tie, the lexicographical ordering of the robots is used to break the tie. The reason for using lexicographical ordering is to make sure all robots in the same neighborhood select the same service robot.

5.4 Relay Node Allocation

A robot can be a relay node to a robot that has a lower cost to service a task. We define the cost to service a task as the length of the least costing path to the task location in the robot's local world graph: $envMap$. Amongst the robots within r_c, the ones with cost to service a task less than that of the host robot form the set of candidates to which the host node can $follow$ as relay node (Eqn. 5.12). If the number of candidates is higher than a threshold the host robot backs off from the task and does not select any leader for the task. This behavior avoids clustering amongst robots and also aides in servicing multiple tasks simultaneously. We use a threshold of 3 in simulations. From the set of leader candidates to a task for a robot, the one with highest service cost is selected as the leader (Eqn. 5.13).

$$\forall j \quad candidateSet_{i',j} = \{i : task\_costs_{i'}(i,j) < cost\_vector_{i'}(j)\} \tag{5.12}$$

$$\forall j \qquad leader_{i'}(j) = \underset{i}{argmax}\{task\_costs_{i'}(i,j)\}, \quad i \in candidateSet_{i',j} \tag{5.13}$$

Since cost is defined in terms of shortest distance to the task location, the selected leader for a task is the candidate physically nearest to the robot. This selection procedure enforces a hierarchy in the set of robots that are involved in the service of a task. The one with immediate higher cost assumes the relay node role for a robot. Robot with the highest cost is closest to the base station or last in the service chain and robots with successively lower costs are farther away from the base station. The relay node assignment stage culminates by computing cost to be a relay node for each of the active tasks. $costToRelay_{i'}$ vector is of length $numTask$ and $costToRelay_{i'}(j)$ is the cost for robot $R_{i'}$ to be a relay robot for task T_j to robot $R_{leader(j)}$ and is equal to the length of the least cost path from$P_{i'}$ to $P_{leader(j)}$. The $costToRelay_{i'}(j)$ for an inactive task is ∞.

5.5 Self Role Allocation and Target Computation

A robot can perform multiple roles at the same time. The maximum number of parallel roles a robot can assume is a system parameter that must be set a priori. *numRoles* is the maximum number of roles a robot can perform simultaneously. If a robot finds itself to be selected as the service agent for a task(s), it assumes the role of service agent for the task with minimum service cost (Eqn. 5.14). A robot can assume the role of service agent for atmost one task.

$$serviceTask_{i'} = \underset{j}{argmin}\{cost\_vector_{i'}(j) : \ leader_{i'}(j) = i'\} \tag{5.14}$$

Amongst all tasks the one for which $costToRelay_{i'}(j)$ is minimum is then selected as a relay task. This process is repeated until the total number of roles assigned to a robot equal $numRoles$. The process stops midway if more roles are not available, either because

$numRoles$ > number of active tasks or within r_c the robot has the smallest cost to service all the active tasks. Each role corresponds to a target location. In case of service agent role, target is the task location, while in case of relay node assignment it is the location of the leader being followed. Target computation is performed as given in Algorithm 1.

Algorithm 1 computeTargets ($R_{i'}$.info, numRoles, task_status, numTask)

if isempty($serviceTask_{i'}$) **then**
  start = 1
else
  $targets_{i'}(1) = serviceTask_{i'}.location$
  start = 2
end if
taskIndex = 1
for i = start:1:numRoles **do**
  if taskIndex > numTask **then**
    break
  end if
  relayTask(taskIndex) = $\underset{j}{argmin}\ \{costToRelay_{i'}(j) :\ task\_status(j) =$ "$active$"$\}$
  $costToRelay_{i'}$(relayTask(taskIndex)) = ∞
  $targets_{i'}(i)$ = $leader_{i'}$(relayTask).location
  taskIndex += 1
end for

5.6 Move!

This is the motion planning stage of the pipeline and is shown schematically in Figure 5.2. The first step towards calculating the next move is to find a path to each of the target locations computed in the previous step. We run Dijkstra's algorithm [19] on the potential gradient environment map *envMap*, to compute the shortest (least costing) path to each target. A path is a vector of cell indices the robot must visit to reach the target location. Next, we find the *weighing parameter* associated with each target. It is a function of the

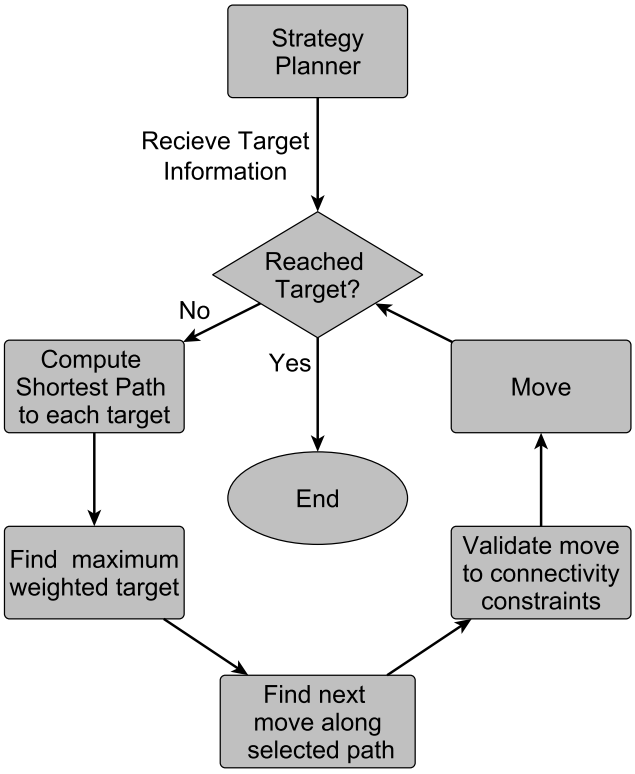


Figure 5.2: Robot level control flow diagram: Motion Planner

length of the shortest (least costing) path to the target location and role that the robot assumes corresponding to the target. The first move along the highest weighted path is then selected as the next move.

The move is validated to satisfy network connectivity constraints to ensure the robot is connected to the base station at the new location. There must exist either a one hop link or a relay node, that provides connectivity at the next location. The procedure to ensure the connectivity requirements is described next.

The host robot can detect the existence of a direct connectivity link to the base station. If such a link is not present, the next location must be within the communication range of a robot that is connected to base. We now present strategies based on two different

models of the environment dynamics.

5.6.1 Static Environment Model

In this approach the environment is modeled as a static world and makes the following assumption: all obstacles, hard and soft, are static and only one robot can move at a time.

Under the model, robots currently within communicable distance of the host robot and connected to the base station form the candidate set for relay agent at the next location. Robots within one-hop communication range of base station are also included in the candidate set. Robots connected to the base station through the host robot must not be included in the candidate set as this could lead to a network loop disconnecting both the robots from the base station. A robot that consists of the host robot as a hop on the connectivity link to base would have link length at least equal to host robot's link length + 1. Further if both the robots have equal link length, the first hop in both the links could be same. This could lead to situations, where both robots move assuming the other would provide connectivity and get disconnected. Hence to ensure connectivity, neighboring robots with a link to the base station of length at most equal to the host robot's link length - 1 are considered.

If the new location is within communication range of any of the robots in the candidate set the host robot is allowed to move. This provides connectivity at the new location under static world assumption.

5.6.2 Dynamic Environment Model

We now extend network connectivity checks to a dynamic environment. The Environment is modeled as a dynamic world and makes the following assumption: all robots/soft obstacles can move simultaneously.

To ascertain connectivity in this scenario, non favorable movement of neighboring robots must also be accounted for. This is done by assuming worst possible direction of movement of all candidate robots. To realize this we use (r_c - r_t) as the communication range of robots instead of r_c. If the new location is within (r_c - r_t) distance away from any of the candidate robots connectivity can always be guaranteed at the new location. Once the connectivity check is satisfied, the robot moves to the new location while remaining connected to the base station.

5.6.3 Recovery Behavior

The robots could still get disconnected due to a relay node going astray or other unforeseen circumstances such as link failures. Hence, it is imperative that the system is robust to such failures and is able to recover. We define here behavior which the robot must follow in a state when it has lost connectivity to base. We name it as *steepestDescent* behavior. The disconnected robot computes the direction of steepest descent of potential gradient in its local neighborhood and starts moving towards it. It would then be moving towards the base station. Also since a relay node, to which it might have lost connectivity, is typically in a lower potential band, this increases the probability of reconnecting. Robots switch to the same behavior, to return to base station once all tasks are serviced.

Chapter 6

Results and Analysis

6.1 Simulation Setup

A simulation framework was built in Matlab for validating the strategies. The simulations were done in a discretized environment of size 20 units $\times$ 20 units space. The base station of size 2 units $\times$ 2 units was placed at the center. Obstructed space was varied from 0% to 16% (hard and soft). We have used 20 randomly generated environments for simulations where tasks were generated at random locations dynamically during mission execution. A sample environment is shown in Figure 4.1. Upto two tasks appeared simultaneously at any given point in time. Simulations were carried out for various combinations of the parameters. Table 6.1 summarizes the parameters used and the corresponding values varied for the simulations. A total of 1000 simulations were carried out to evaluate the performance of the algorithm.

Typically, the the time taken for one information exchange cycle amongst one-hop connected robots is less than the time consumed in a robot move. We capture this aspect in

Number of Agents	{ 8, 12, 16, 20, 24 }
Number of Tasks	{ 2, 4, 6, 8, 10 }
Obstructed Cells (%)	{ 0,20 }

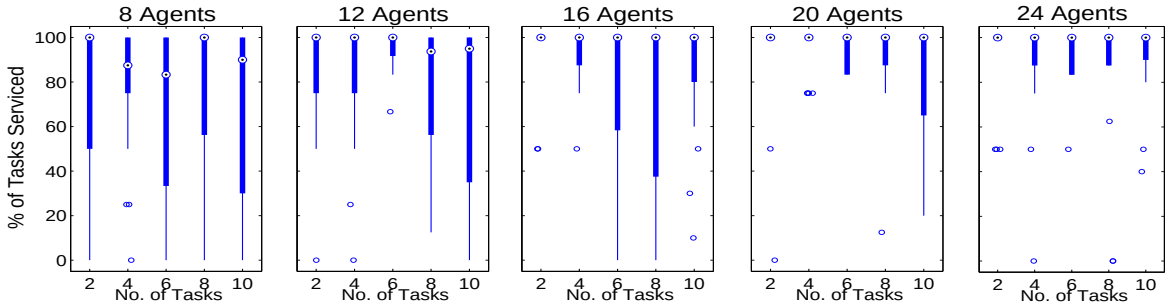
Table 6.1: Simulation Parameters

simulation by using a ratio of 4:1, i.e. for every four communication cycles there is one movement cycle.

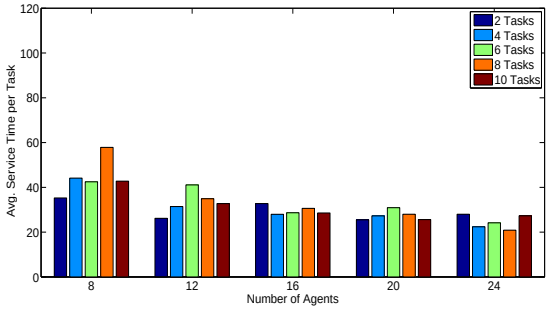
The number of factors affecting algorithm performance is very large. We provide a careful analysis of the various factors involved and their interplay that leads to resultant system behavior. To establish the ground truth, we conducted 500 simulations in an obstacle-free environment. 25 different mission scenarios were designed and 20 different starting configurations were generated for each scenario. The extensive simulations lead to some interesting observations on system performance. We present the same here.

6.2 Results and Analysis

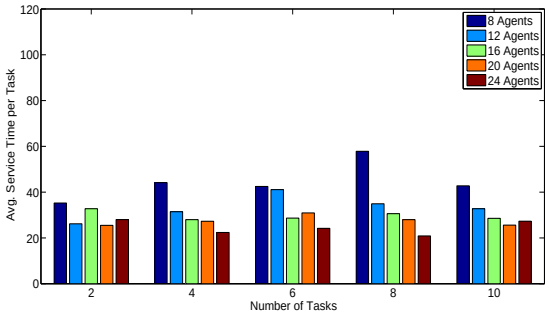
The algorithm has high rate of convergence for the number of tasks serviced successfully. As is clearly observable from Figure 6.1a, of the 25 different simulation scenarios, the median for the percentage of tasks completed in 20 different configurations is 100% in 20 scenarios i.e. in 80% of the total scenarios. In the other 5 cases the value fluctuates between 81%-98%. This is because in these 5 scenarios the *tasks per agent* ratio was high. The smallest median value for the percentage of tasks completed in a mission is 81% (8 Agents, 6 Tasks). This also shows that the algorithm does not get held up due to deadlock situations. The self role allocation allows for dynamic role reassignment amongst agents,



(a) Percentage of Tasks Serviced vs Number of Tasks for an obstacle-free environment. The definite increase in median value shows that higher number of robots successfully reduce the average service time per task.



(b) Average service time vs number of agents for obstacle-free environments

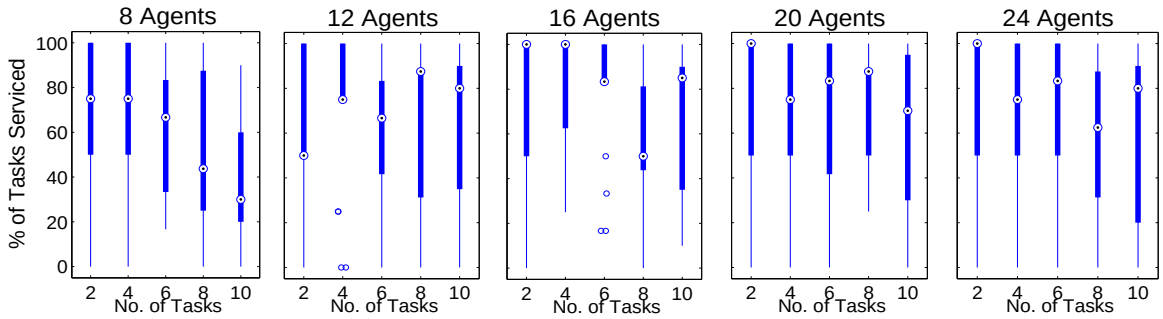


(c) Average service time vs number of tasks for obstacle-free environments

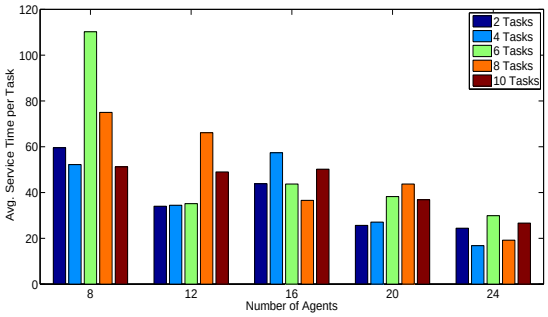
Figure 6.1: Simulation results for obstacle-free environment

thus avoiding deadlocks.

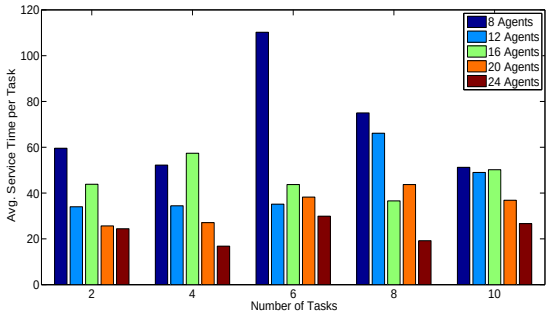
As the number of agents increases the mission completion rate increases. Further, standard deviation shows a consistent decline as the *agents per task* ratio increases. The average time to service a task also shows a uniform drop in magnitude with more number of agents. Figures 6.1b and 6.2b capture this behavior. Figures 6.1c and 6.2c present the variation in average service time per task with increase in number of agents for various number of tasks. A steady improvement in service times is observed for missions of various sizes. Thus the algorithm is scalable and is successfully able to utilize extra number of available agents.



(a) Percentage of Tasks Serviced vs Number of Tasks for an obstacle-rich (10% static) environment.



(b) Average service time vs number of agents for obstacle-rich environments (10% static)



(c) Average service time vs number of tasks for obstacle-rich environments (10% static)

Figure 6.2: Simulation results for obstacle-rich environment

In Figure 6.1a, the plot for 8 agents shows the algorithm performing badly with relatively low mission completion rates and high standard deviation. Standard deviation is also high for missions with large number of tasks and decreases only gradually with increase in the number of agents. In Figures 6.1b and 6.2b also, the average time taken to service a task is very high for 8 agents. This is because the *agents per task* ratio decreases to very low values, even <1 at times, in these missions. Even though all tasks do not appear at the same instant, by virtue of random generation, tasks could appear far and wide in the world and the number of available agents falls short to adequately service all tasks.

To study the performance of the algorithm in real world environments we also conducted 500 simulations in obstacle-rich environments. The same scenarios and initial configura-

tions as obstacle-free environments were used. We envisioned the resulting plots to have a similar pattern as in the case of obstacle-free environments with a modest decrease in performance. As was expected, the mission completion rate increases with increase in the number of agents available. However after a certain value the performance starts to deteriorate with further increase in the number of agents. The presence of static obstacles in the environment constrains the movement space and inflates the effect of soft obstacles, read other agents, as the number of agents keeps increasing. These mobile obstacles have a major impact on performance. Large number of mobile obstacles severely affects performance metrics and even jeopardize mission completion as seen in Figure 6.2a. The mission completion rates show improvement with increase in agents from 8 to 16; thereafter with further increase in the number of agents the performance starts to decline. The soft obstacles cover as much as 6% (in addition to the 10% static obstacles) of the configuration space in case of 24 agent teams. The increase in the number of outliers with large team size observed in Figure 6.1a is also attributed to the same reasoning. It is argued that for a given environment size there is an upper limit on the maximum number of agents up to which the performance improves. With anymore increase in number of agents there is a decline in performance. These results are consistent with the existing literature ([3] and the references therein).

Figures 6.1b, 6.1c, 6.2b and 6.2c consider only those tasks which were successfully serviced to compute the average time taken to service a task. While the improvement in average service time with larger number of agents is homogeneous (Figure 6.2b), the variation in service time with more number of tasks in a mission has an interesting pattern. All team sizes (number of agents) show a decline in performance initially. The peaks after which the change in behavior is observed is different for different team sizes. The peak for 8 agents occurs at 6 task missions. For 12 and 20 agent teams it occurs at 8 task missions

and appears early at 4 tasks for 16 agent teams. A peak observed at 6 task missions for 24 agents is only a local maxima. The global peak is not captured in these simulations and would require missions with higher number of tasks. Agents need to be sufficiently well spread out in the world to be able to service new tasks that may appear at random locations. Missions with more tasks allow the agents to cover more area in time, which reduces the service time for tasks that appear later as also the average service time. As agents move away from each other and cover a larger fraction of the environment, the effect of soft obstacles also diminishes. This explains the distinctive pattern observed in Figure 6.2b and on a smaller scale in Figure 6.1b.

6.3 Discussions

Potential Gradient Mapping: Reasons for using potential gradient based environment mapping, with base station at zero potential and a linear increase in potential with distance from the base station, are twofold. Firstly, it gives a bias to path selection using Dijkstra. Of the multiple equal costing paths (if available), Dijkstra always selects the one that has maximum proximity to the base station. Biasing the path selection in this manner results in a major reduction in the total number of relay nodes required to service a task. Secondly, the *steepestDescent* behavior for the robots is also modeled using potential gradient. The direction of local steepest descent in a robot's neighborhood is always in the direction of the base station. When a robot gets disconnected it starts to move in the direction of steepest local descent. Nearly always the relay node for a robot would be at a lower potential than the robot itself, hence when connectivity is lost, moving in the direction of local steepest descent increases the probability of retrieving the lost link and in the worst case allows the robot to move within direct communication range of the

base station. The same behavior is used to make robots return to base on task completion.

Network Connectivity: The network connectivity checks applied on robot motion as discussed in Section 5.6 ensure robot connectivity during task service. These checks allow the robot team to form networked chains to service tasks specially those that appear at a greater distance away from the base station. The possible points of losing connectivity during network formation are when a robot takes a turn around an obstacle (blind spots) or a relay node changes its self allocation. Such scenarios are taken care of using the disconnected behavior as define in Section 5.6.

One Hop Communication Only one hop robot communication is used amongst the robots for information sharing. This allows for a modular design and reduces the network load relative to multi hop information sharing.

Collision and Obstacle Avoidance: By classifying obstacle detections as hard and soft obstacles and performing corresponding action sequences as mentioned in Section 5.1 robots are able to successfully prevent collisions with obstacles and other robots.

Chapter 7

Conclusion and Future Directions

We have developed a distributed online multi-robot mechanism for mission planning. This work provides strategies to facilitate network connected task servicing. We have shown it to be scalable to large number of agents and having a high convergence rate. Potential gradient mapping and graph theoretic approach has allowed us to port well established principles to our application. Only one-hop communication is used and also there is no negotiation or explicit cooperation amongst agents for task and role allocation. The idea of self role allocation is inspired from natural swarms and allows dynamic role reassignment and role exchange amongst robots. This plays a very crucial role in avoiding deadlocks which have been widely reported in multi-robotics literature. *steepestDescent* behavior allows robots to regain connectivity in case of disconnections. This adds robustness to the algorithm to network failures.

The algorithm is extensible to other applications like target tracking and cooperative localization. Extension to cooperative localization would in turn help relaxing the perfect localization assumed in the text. Stricter network considerations involving network traffic and link bandwidth are possible extensions to this work. Also analyses of the algorithm in a continuous environment could provide interesting insights.

Bibliography

[1] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing." in *Robotics: Science and Systems*, vol. 5, 2005.

[2] P. Miller, *Smart swarm.* Collins London, UK, 2010.

[3] J. Vazquez and C. Malcolm, "Distributed multirobot exploration maintaining a mobile network," in *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*, vol. 3, June 2004, pp. 113–118 Vol.3.

[4] P. Ulam and R. Arkin, "When good communication go bad: communications recovery for multi-robot teams," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 4, April 2004, pp. 3727–3734 Vol.4.

[5] R. Pandey, A. K. Singh, and K. M. Krishna, "Multi-robot exploration with communication requirement to a moving base station," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on.* IEEE, 2012, pp. 823–828.

[6] G. Hollinger and S. Singh, "Multi-robot coordination with periodic connectivity," in *Robotics and Automation (ICRA), IEEE International Conference on*, 2010, pp. 4457–4462.

[7] N. Gans, J. Shea, P. Barooah, and W. Dixon, "Ensuring network connectivity of uav's performing video reconnaissance," in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, Nov 2008, pp. 1–7.

[8] J. Reich, V. Misra, D. Rubenstein, and G. Zussman, "Connectivity maintenance in mobile wireless networks via constrained mobility," *Selected Areas in Communications, IEEE Journal on*, vol. 30, no. 5, pp. 935–950, 2012.

[9] N. Michael, M. M. Zavlanos, V. Kumar, and G. J. Pappas, "Maintaining connectivity in mobile robot networks," in *Experimental Robotics*. Springer, 2009, pp. 117–126.

[10] Y. Pei, M. W. Mutka, and N. Xi, "Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5460–5465.

[11] A. Mosteo, L. Montano, and M. Lagoudakis, "Guaranteed-performance multi-robot routing under limited communication range," in *Distributed Autonomous Robotic Systems 8*. Springer Berlin Heidelberg, 2009, pp. 491–502.

[12] B. Brggemann, M. Brunner, and D. Schulz, "Spatially constrained coordinated navigation for a multi-robot system," *Ad Hoc Networks*, vol. 11, no. 7, pp. 1919 – 1930, 2013.

[13] P. Maini and P. B. Sujit, "Multi-robot base connectivity constrained task servicing," in *Intelligent Unmanned Systems (ICIUS), 2013 International Conference on*, 2013.

[14] A. Viguria, I. Maza, and A. Ollero, "S+t: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, May 2008, pp. 3163–3168.

[15] S. S. Ponda, L. B. Johnson, A. N. Kopeikin, H. Choi, and J. P. How, "Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 5, pp. 861 –869, June 2012.

[16] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, August 2009.

[17] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1-4, pp. 113–126, 2001.

[18] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor, "Maintaining network connectivity and performance in robot teams," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 111–131, 2008.

[19] E. W. Dijkstra, "A note on two problems in connexion with graphs," vol. 1, pp. 269–271, 1959.