12-2013

Multi-Robot Task Allocation: A Spatial Queuing Approach

William H. Lenagh
University of Nebraska at Omaha

Multi-Robot Task Allocation: A Spatial Queuing Approach

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment
of the Requirements for the Degree

Master of Science in Computer Science

University of Nebraska at Omaha

by

William H. Lenagh

December 2013

Supervisory Committee:

Dr. Prithviraj Dasgupta

Dr. Sanjukta Bhowmick

Dr. Jack Heidel

Dr. Angelica Munoz-Melendez

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Multi-Robot Task Allocation: A Spatial Queuing Approach

William H. Lenagh, M.S.

University of Nebraska, 2013

Advisor: Dr. Prithviraj Dasgupta

Multi-Robot Task Allocation (MRTA) is an important area of research in autonomous multi-robot systems. The main problem in MRTA is to match a set of robots to a set of tasks so that the tasks can be completed by the robots while optimizing a certain metric such as the time required to complete all tasks, distance traveled by the robots and energy expended by the robots. We consider a scenario where the tasks can appear dynamically and the location of tasks are not known *a priori* by the robots. Additionally, for a task to be completed, it needs to be performed by multiple robots. This setting is called the MR-ST-TA (multi-robot, single-task, time-extended assginment) category of MRTA; solving the MRTA problem for this category is a known NP-hard problem. In this thesis, we address this problem by proposing a new algorithm that uses a spatial queue-based model to allocate tasks between robots while comparing its performance to several other known methods. We have implemented these algorithms on an accurately simulated model of Corobot robots within the Webots simulator for different numbers of robots and tasks. The results show that our method is adept in all proffered environments, especially scenarios that benefit from path planning, whereas other methods display inherent weakness at one

end of the spectrum: a decentralized greedy approach exhibits inefficient behavior as the robot to task ratio dips below one, whereas the Hungarian method (an offline algorithm) fails to keep pace as the robot count increases.

Acknowledgments

I would like to take a moment to acknowledge the invaluable guidance provided by Dr. Dasgupta throughout this journey; the culmination of which is evinced in this document.

I would be remiss if I did not also thank my wife, Kim, for her support amid the long hours requisite in such an undertaking, and to my family and friends for their words of encouragement.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Multi-Robot Task Allocation(MRTA) plays a key role in any system whose primary objective is to automate the performance of one or more tasks through autonomous robots. MRTA is used in numerous applications of robotic systems including reconnaissance [11], unmanned search and rescue operations ([4], [5]), cooperative transportation ([2], [16], [9]), autonomous exploration ([3], [12]), etc.

We consider the MRTA problem within the context of an automated landmine detection scenario. In this scenario, a set of robots are deployed within a bounded 2D environment with potential landmines. The location of the landmines is not known *a priori*. Robots are equipped with sensors that are capable of detecting landmine-like objects, albeit within a certain level of uncertainty due to sensor noise. Robots initially explore the environment and when a robot finds an object of interest that could potentially be a landmine, it requests other robots, possibly with different sensor types to visit the location of the detection and confirm the object on their sensors. Within this setting, a task corresponds to a set of robots visiting the location of an object of interest and recording the object's signature on their sensors. For legibility, we have referred to each robot's visit to the object's location and taking its reading, as the robot performing its portion of the task. Robots can perform a

task asynchronously by performing their portion of the task at different times, but within a certain deadline since the discovery of the object. A task is considered to be complete when the desired number of robots have performed their portion of the task. Finally, tasks can arrive dynamically as robots explore the environment and find objects of interest. Within this context, the MRTA problem corresponds to finding a suitable allocation of tasks to robots so that the total time required to complete the tasks is reduced.

The basic question answered by the solution to an MRTA problem is: which robots should execute which tasks? And when or in what order should they be executed? Given a set of alternative task schedules, an MRTA algorithm must choose a schedule based on certain domain specific constraints such as the time required to complete all the tasks, or, the total distance traveled by the robots, or, perhaps, the energy expended by the robots. The MRTA problem is known to be an NP-hard problem [1] and finding the optimal solution to the problem is not feasible beyond very trivial scenarios. Most of the existing research on MRTA focuses on developing heuristics-based solutions, while compromising overall system fitness for timely action.

MRTA solutions can generally be divided into two categories: centralized and decentralized, or distributed. Centralized algorithms have the advantage of a global view, but introduce a single point of failure in the system. Decentralized or distributed approaches rely on communication between robots in order to achieve overall system fitness. Popular methods for tackling distributed problems are to partition the environment into logical regions ([7], [11]) and to use the market concept of auctions to enforce decisions based on fitness parameters ([18], [6], [5], [4], [3], [2], [15]).

Our solution is to locally build a queue of preferred tasks for each robot based on normalized matrix calculations involving the robot's location and the distance to known tasks within the environment. Auctions are then employed as a coordination mechanism for determining robots' intentions and selecting the most efficient assignment

at any time. Combined together, these two aspects enable our solution to quickly award task assignments in a distributed manner while allowing for dynamic change. Real world environments are not static and typically introduce change that must be dealt with quickly. Whether this change is manifested as new tasks, morphing priorities, or system malfunction, it must be modeled and adapted to. Our solution is flexible in that a schedule of tasks is not static, but the task queue at each robot is reevaluated when new information is collected; each robot need only bid based on the task located at the head of its queue.

The remainder of this document is structured as follows: the next chapter introduces a wealth of related works consulted in preparing for our work. Chapter 3 provides a formal definition of the base problem studied in this thesis. Chapter 4 illustrates the theory and structure of our spatial queuing algorithm. Chapter 5 lays out our experimental setup, defines the algorithms chosen for comparison, and reports the results and analysis of the completed simulations. The last chapter, chapter 6, summarizes conclusions drawn from our labors and points to future work in the vein of what is discussed herein.

Chapter 2

Related Work

MRTA encompasses a rather large range of possible problem domains and system configurations, and due to its increasing popularity and applicability there are many relevant recent studies that fall within its boundaries.

A good first step would be to properly define the types of problems, and their basic attributes, that a MRTA system would solve. This is exactly what [1] proclaims to do by constructing a formal taxonomy of task allocation in the context of multi-robot systems. Their taxonomy revolves around three axes, three dimensions of the problem space; these speak to the nature of the robot's abilities, the requirements of the tasks, and the time-centric availability of task information. Specifically, they define one axis as book ended by either single-task (ST) or multi-task robots (MT); signifying whether or not the robot's in question are capable of executing more than one task at a time. The second axis consists of single-robot (SR) or multi-robot tasks (MR) wherein a task requires either one (SR) or more than one robot (MR) in order to complete the requirements necessary for completion. Lastly, a third axis relating to the flow of tasks into the robot's environment: instantaneous assignment (IA) refers to a static scenario where future considerations do not apply, conversely time-extended assignment (TA) implies a dynamic setting in which allocations may

change or additional tasks enter the system in future iterations.

In [2] the same authors aim to minimize resource usage, task completion time, and communication overhead by implementing a fitness-based auction system with negotiation and task commitment. They then test their system in both tightly-coupled and loosely coupled coordination experiments, demonstrating that their solution is adaptable and resilient in a variety of situations. As part of their solution they introduce a publish/subscribe messaging protocol whereby messages are addressed by content as opposed to destination. Anonymous messages are broadcast over the network by producers; interested agents evaluate the metadata and either subscribe to or ignore messages in that vein. In this way environmental resources become topics of interest in the communication model.

Gil Jones, et al. focus on methods for forming ad-hoc teams at a moments notice in [3], where the constituents have little *a priori* knowledge regarding the tasks, other robots, these robots capabilities, or the environment in which they will be operating. A framework such as this would allow considerable flexibility in studying scenarios where teams must be formed on very short notice, such as in an emergency, and where team members may differ greatly and may malfunction at any time. They begin to investigate such a scenario in [4] where they introduce the concept of oversubscribed domains. Oversubscribed domains refer to scenarios in which tasks have deadlines that cannot all be met due to limited available resources. The authors pursue a regression-based algorithm for guiding decision making within a fire fighting disaster scenario. They demonstrate that a learning-enhanced approach can efficiently manage allocation while incurring few penalties and respecting the urgency of the system demands.

The authors extend their work in [5] to include time-extended allocation within a disaster response scenario. Route planning is complicated by debris forcing sub-auctions to bulldozer robots for clearing optimal routes based on scheduling priorities.

They experiment with a clustering methodology and genetic algorithms; the latter yielding better results at the cost of computation time. This approach is best suited when solution quality is of paramount importance rendering computation time less of a priority.

Li, Sun, and Yang consider the problem of reallocating tasks online, during execution, so as to counteract dynamic change within the environment in [6]. Changes within the environment at run time may invalidate assumptions used in estimating work during the initial assignment, and therefore may render the solution inadequate for the new state of affairs. The authors propose adopting a market based approach, but with a well-defined communication protocol for inter-robot communications with the goal of achieving a fault-tolerant network for dynamic reallocation.

Liu and Shell focus on a large-scale online MRTA algorithm in [7] that mixes both centralized and decentralized approaches in order to take advantage of spacial and temporal efficiencies while reducing overall global communication. The assumption they are building off of states that once an initial assignment has been computed, subsequent reassignments require only a fraction of global information present in the simulation. Their algorithm identifies clusters, or partitions, of strongly connected robot-task pairs and operates on these clusters in parallel. Dynamic online assignment reduced the time complexity by a factor of K^3, where K represents the number of partitions; in addition only 10% of the utility values needed to be calculated and communicated as compared to a strictly centralized approach.

Liu and Shell introduce the interval Hungarian algorithm in [8] which focuses on measuring the effects of uncertainty on the outcome of task allocation. This algorithm assigns robots to tasks in a one-to-one ratio, but also calculates an interval representing the tolerance of the assignment to outside forces which may disrupt or invalidate the value of the assignment. These outside forces are modeled using probability distributions. Their algorithm is a generic solution to the optimal assignment

problem as it does not rely on domain-specific information regarding the structure of the problem or the source of the uncertainty. Their results show that the interval Hungarian algorithm greatly reduces the risk of misallocation when utility estimations are uncertain or unreliable. In [13], Liu and Shell propose an incremental allocation system based on the Hungarian algorithm. Beginning from a weighted bi-graph, a replacement allocation can be devised in linear time and tuned to balance previous match resilience. They document another incremental approach in [14] which produces an optimal solution with increasingly efficient feasible solutions at each intermediate step. Their task swapping structure lends itself to a decentralized approach, but a centralized implementation offers fewer stages albeit with more communication overhead.

Another work focusing on uncertainty is Sucan and Kavraki's simultaneous task and motion planning (STAMP) in [17]. They argue that motion planning cannot be decoupled from task planning as any infeasibility in executing physical motions renders the task planning useless. They use their concept of a Task Motion Multigraph (TMM) to encode hardware capabilities into the task graph and then implement a Markov decision process to guide the robot by incorporating feasibility probabilities into the decision making process. Their experimentation shows a significant improvement in feasibility probability using this method as compared to previous work in STAMP.

Seow, Dang, and Lee apply task allocation to the real-world taxi dispatch system of Singapore in [9]. Using the infrastructure of the centralized system currently in place they propose a distributed model whereby the on-board computers act as agents on behalf of the drivers. The environment is partitioned into logical regions, grouping taxis within those regions who then negotiate concurrent assignments of pending requests with a focus on group average. This reduces average customer wait time as well as empty taxi cruising time.

In a similar vein, Lim and Rus test their stochastic path planning solution against

a Singapore road network incorporating historical traffic and travel data in [16]. The stochastic properties of their model describe the real-world situation in which the optimal sub-structure of a shortest path does not hold when travel time probabilities are factored into the equation. In their work a generalized resource network is modeled as a graph representing agents, resources, and destinations with intermediate node constraints forming a broad outline of the network's flow. They use convex hulls within a mean-variance plane to locate the most efficient sequences between constrained regions.

Dasgupta describes a cooperative foraging scenario with shared task execution in [12]. Task locations are not known *a priori* and must be located by the workers at run time. Once found a robot cannot complete a task on its own, but must request the collaboration of other robots in order to fulfill the task requirements. Each robot can perform their portion independently, and after doing so deposits virtual pheromone representing the portion of the task it completed; this pheromone decays over time and as such all portions of the task must be completed within the deadlines set forth by the decay rate in order for a task to be completed successfully. He goes on to detail four heuristic algorithms for approaching this scenario, the most successful being Most Proximal Task First whereby the robot selects tasks nearing completion with the least number of robots closer than itself. This algorithm avoids inefficient allocations that result in robots approaching a task that is completed by others before it arrives on location, wasting resources in the process.

The MRTA problem has also been studied recently in the context of multi-vehicle routing. In [10], the authors investigate a scenario much like a time-extended version of our environment: routing policies for multiple vehicles servicing tasks with multiple classes of demands (priorities). Tasks enter the system according to a Poisson process and are uniformly distributed within the environment. Each vehicle is assigned a unique region of responsibility; the vehicles then calculate a service route for each

demand class according to their priorities. The authors prove a lower and upper bound and show that their queuing policy performs within a constant factor of this lower bound, dependent only upon the number of demand classes present in the scenario. Celik and Modiano frame MRTA within a mobile data collecting network in [11]. The worker vehicles, given knowledge of sensors that require harvesting, must build a service route schedule that minimizes average message wait time given their limited radius of communication. The authors describe policies for single collector scenarios, as well as multiple collector scenarios with and without possible interference between assigned sectors and frequency bands.

Zhang, Collins and Barbu build on stochastic clustering auction research in [15] which uses simulated annealing to explore an allocation space. Tasks are clustered by an auctioneer based on cost estimations submitted by participating worker robots. These values are combined to form a synergistic view of the task graph; connected components are then transferred or swapped between randomly selected robots based on stochastic probabilities.

Luo generalizes the competitive analysis of the online weighted bipartite matching problem for groups of tasks in [18]. Under the same assumptions as previous work, he shows that the competitive ratio of repeated greedy auctions is independent of the number of tasks and robots in the problem, and becomes constant given that either the size of the tasks groups or the budget for each robot remain constant.

In contrast to the works above, our spatial queuing approach introduces an element of inter-relation between tasks not found in any of the strategies referenced above. It takes advantage of the knowledge that tasks require multiple inspections and grades each task based on its overall proximity to all other known tasks. A task, x, equidistant to a robot's current position, as compared to some other task y, but in close proximity to a third task z, will have a higher overall score and therefore be more attractive to the robot as it presents an opportunity to efficiently visit multiple

tasks in one sortie. Our intuition is that this consideration for future assignments should reduce the time and distance required to complete all task demands when compared to an approach that is only concerned with finding a task schedule that has the instantaneous best fit.

Chapter 3

Task Allocation: A Spatial Queuing Approach

As mentioned before our work focuses on a scenario classified by [1] as MR-ST-TA: multi-robot tasks, one task per robot at any singular time, with time-extended assignment of tasks. In the motivating domain of landmine detection this translates to robots focusing on one task: to investigate an assigned area and perform the sensory analysis required to detect any buried landmines in that region. Task regions are embodied by a single point in 2D space signifying the centroid of a region as identified by a different subsystem such as a coverage planner. For the sake of this study the robots need only arrive at the centroid to complete a task while execution of the task would be handled by the robot's controller in a task execution subsystem. For the purpose of redundancy and accuracy, each task region must be inspected by multiple robots to confirm their peers' results, driving down the statistical error inherent in any physical system. However, in order to better utilize resources across multiple regions while at the same time limiting competition within shared space, task regions are only assigned to one robot at a time. Our scenario is classified as time-extended assignment due to the fact that task regions are not known *a priori*, and in

fact are presented to the task allocation system by another, higher layer subsystem over time. The task allocation layer must then take this data and manage it among a team of robots that are likely dispersed over a wide area. Also, the system should be able to handle changes in the environment, whether that be additional task regions that are presented during run time, or changes to the priorities of regions based on the findings of other robots in the system.

Computation of an optimal solution is known to be NP-Hard [1], and is in fact a variation on the traveling salesman problem. To address this complexity our solution concentrates on building a queue of preferred tasks for a robot using a heuristic that is based on the inter-robot and inter-task distances.

The specific problem we are investigating offers up some attributes that can be used to our advantage when formulating a system for task allocation. First, we know that the task locations are stationary, and while it is possible for additional task locations to present themselves dynamically during run time, once included into the system they remain unchanging. Second, it is guaranteed that each objective will require multiple inspections, and therefore two or more robots must be allocated each task at some point during execution. Based on these observations, providing for some level of interrelation between tasks should increase performance by incorporating a measure of "look-ahead" into the overall score computed for a task.

Considering that movement is the most taxing energy sink in a land roving scenario, optimizing energy use, and by way of supposition, distance, should be the ideal we pursue. To accomplish this we turn to the concept of spatial queueing, whereby each robot constructs a sorted preference list of tasks; each task receives a score that represents its likelihood of selection based on its Euclidean distance, both from the robot's current position and also in relation to the other tasks present in the environment.

Let $E \subset \mathbb{R}^2$ represent a bounded two-dimensional environment and $\mathrm{R} = \{r_i : 1 \leq$

$i \leq m\}$ represent a team of m robots deployed within E. In addition, there is a set of n tasks, $T = \{\ \tau_i : 1 \leq i \leq n\ \}$ dispersed throughout E, each of which requires a predetermined number of inspections, $d_{\tau_i} \epsilon \mathbb{Z}$. The positions of the robots and tasks are denoted by $\rho_{r_i} \epsilon E$ and $\rho_{\tau_i} \epsilon E$ respectively. These position variables are used to calculate the Euclidean distance $d_{i,j} = \left| \rho_{\tau_i} - \rho_{\tau_j} \right|$ between two tasks and $\hat{d}_{i,j} = \left| \rho_{\tau_i} - \rho_{r_j} \right|$ between a robot and a task.

Inter-task distances form the basis of our method as they are used in composing the transition matrix; a data structure generated and maintained by each robot. The values of the transition matrix are the normalized inverse Euclidean distances between every task pair, as described by the definition of transition matrix M_t in Figure 3.1

$$M_t = \begin{bmatrix} \pi_{11} & \pi_{12} & \dots & \pi_{1n} \\ \pi_{21} & \pi_{22} & \dots & \pi_{2n} \\ & \dots & & \\ \pi_{n1} & \pi_{n2} & \dots & \pi_{nn} \end{bmatrix}, \text{ where } \pi_{i,j} = \frac{\frac{1}{d_{i,j}}}{\sum_{j \neq i} \frac{1}{d_{i,j}}}$$

Figure 3.1: Formal definition of the transition matrix

Each entry π_{ij} of M_t represents the probability of a robot to select task τ_j following τ_i, based on the distance between their locations. Note that $\pi_{ii} = 0$ and therefore the diagonal of the matrix is composed of all zeros.

Initially, the transition matrix is computed for all task pairs, but as the robots' operations progress, each robot recalculates the matrix whenever they complete a task, or when the required number of inspections for a task have been fulfilled by other robots. At this point the completed task can be removed from consideration, hence transitional probabilities referencing this task are set to zero and redistributed among all remaining tasks. Figure 3.2 graphically displays the initial transition matrix, as all robots see it, for the first of our six task environments.

After the initialization phase completes in which team members announce their presence, forming a loose collective, and known task information is disseminated, the

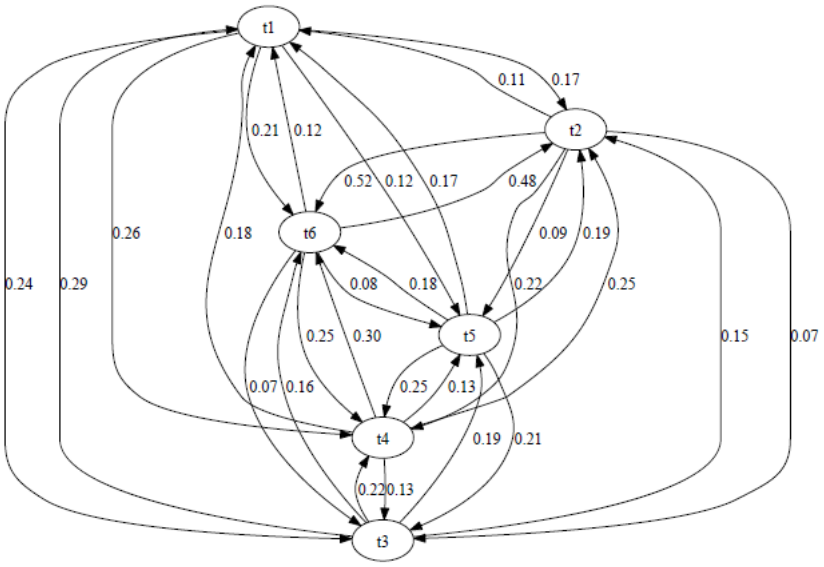


Figure 3.2: Graph of environment 1, with inter-task transition probabilities

task allocation phase can begin. Upon receiving the task coordinates every agent builds the transition matrix as detailed above. Next, they each compose a custom queue of tasks, sorted in descending order based upon the score each task receives when the transition matrix is applied to the robot's state vector.

A robot's state vector is similar to the transition matrix in that it is composed of inverse Euclidean distances, from its current position to each task location (see Figure 3.3). However, to promote valid comparisons from a global perspective, which is a necessity when making comparisons and, subsequently, decisions in a distributed manner, these values are not normalized and do not, therefore, estimate a probability distribution. If normalized, as work is completed an unfair bias will be afforded to robots with fewer task options, the end result of which is non-optimal allocations. Instead, raw inverse distance values are used; leveling the playing field when contrasting bids submitted by two or more robots. In general, a robot's state vector can be formalized as:

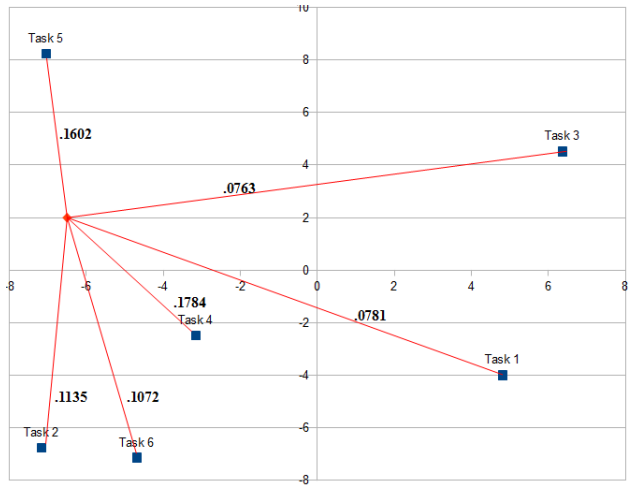


Figure 3.3: Visual representation of Corobot 1's state vector from initial deployment position

$$V_{r_i}(\rho_{r_i}) = (\hat{\pi_{i1}}, \hat{\pi_{i2}}, ..., \hat{\pi_{in}}) \, where \; \hat{\pi_{ij}} = \frac{1}{\hat{d}_{i,j}} \tag{3.1}$$

The product of the state vector and the transition matrix is a vector of task proximity ratings that are then used in the bids for the tasks they represent. In essence this is the overall likelihood of selecting the reference task based upon the viewpoints of each of the other tasks, but weighted by the robot's current position. This conveys the likelihood of a task being selected, not in the present iteration, but one step in the *future*. By then adding the state vector to this resultant vector you achieve a score for each task that incorporates the present with one future period. The robot is considering both immediate distance and relative proximity to other tasks in its decision making calculations. At time t, the task scores are represented by the vector $\acute{V}_{r_i}$ such that:

$$\acute{V}_{r_i}(t) = V_{r_i}(t) \times M_{t,r_i}(t) + V_{r_i}(t) \tag{3.2}$$

The results of the computation are then filtered, to remove occupied or completed tasks, and sorted producing the final preference queue.

$$Q_{r_i}(t) = \{\ q_1, q_2, \ldots, q_n \mid q_i \geq q_{i+1}\ \forall\ i\ \}$$

Initially, no robot-task assignments exist and therefore all robots submit the head of the queue, their most preferred task, for evaluation by the collective. Once all bids have been received the robot with the global optimum, or highest bid, is awarded its preferred task (by virtue of being the only unblocked robot). This triggers a chain reaction in which subsequent high bids are awarded where no conflict of interest exists. If a robot is unblocked only to discover that its preferred task has already been awarded to a more deserving agent it increments the head node of the queue and submits a new bid based on its second most preferred task. If a robot processes its entire queue without winning a task it must sit idle until a task becomes available.

When a task is completed the producing robot, along with any idle robots, must rebuild their queues to incorporate changes in the state of the environment. Once all tasks have been completed, either from the perspective of an individual, or as a whole, they terminate their controller.

The complete algorithm can be reviewed in Algorithm 1

Algorithm 1 Spatial Queing Multi-Robot Task Allocation

```
1: Input: T = { τ_i : 1 ≤ i ≤ n }
2: Build transition matrix M_t as shown in Figure 3.1
3: bid ← 0
4: while an eligible task remains do
5:     if I have not bid in this round then
6:         Calculate state vector as shown in 3.1
7:         Generate resultant vector using equation 3.2
8:         Remove tasks completed by me or occupied by other robots
9:         Sort queue in descending order
10:        if the queue is empty then
11:            wait until task-performed signal received from another robot
12:            if a performed task is now complete then
13:                Rebuild transition matrix M_t
14:            end if
15:        else
16:            bid[r] ← queue[head]                ▷ value of most preferred task
17:            sendBroadcast(BID, bid)
18:        end if
19:    else
20:        wait until competing bids received from all other robots
21:        if I am the highest bidder for my preferred task then
22:            if If task I have bid on is still available then
23:                sendBroadcast(ACCEPT, task)
24:                Execute task
25:                Rebuild transition matrix M_t
26:            else
27:                Move to next entry in queue
28:                if there is a task available in the queue then
29:                    bid ← queue[head]
30:                    sendBroadcast(BID, bid)
31:                else
32:                    wait until task-performed signal received from any active robot
33:                    bid ← 0
34:                    if a performed task is now complete then
35:                        Rebuild transition matrix M_t
36:                    end if
37:                end if
38:            end if
39:        end if
40:    end if
41: end while
```

Chapter 4

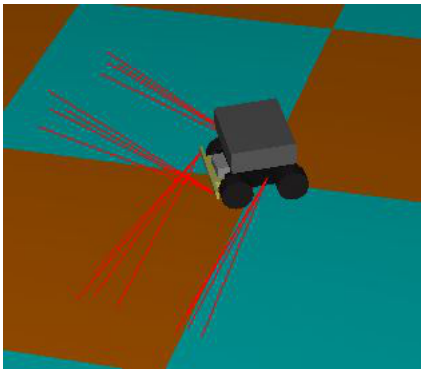
Experimental Setup and Results

4.1 Webots

All of our simulations were built and and executed on the robot simulator Webots (version 6.3.0), http://www.cyberbotics.com/. Webots is a fully integrated design and coding platform, allowing for both virtual robot and environment design and robot-controller architecture construction. Each Webots world file is entirely customizable allowing detailed replicas of real-world robots and objects to be incorporated with the simulator's physics engine. Webots' integrated design environment includes the necessary compilers for C and C++ controllers files as well as a set of APIs, ported in many languages, for interacting with the sensors and actuators of a robot residing in the simulated world.

The robot prototype utilized in all of our simulations is that of the Coroware Corobot robot, an indoor, four-wheeled, skid-steer robot meant to represent a scaled down version of the land mine detecting robots used in an outdoor setting. This robot is equipped with four infrared distance sensors, one on each side of the sagittal plane, oriented sixty degrees from the front of the vehicle, and two cross beam sensors mounted on the front bumper (Figure 4.1a). Coupled with a Braitenberg controller

[20] these four sensors enable the robot to "feel" obstacles around it and react accordingly. Other key devices are the emitter and receiver used for bidirectional wireless communication with other robots in the environment, and a GPS node requisite for localization.



(a) A virutal Corobot, with visible distance sensor rays

(b) A photo of a Corobot robot

Figure 4.1: Virtual and actual snapshots of the Coroware Corobot robot

Simulations take place in a bounded three-dimensional world; movement and object placement are limited to a 20 x 20 meter^2 plane aligned to a Cartesian coordinate system originating at the center of the plane (Figure 4.2). Walls extending into the third dimension exist at the boundaries of this square; robots interpret the walls as obstacles via their infrared sensors.

Robot behavior is expressed primarily by movement in the two-dimensional plane. This behavior is guided by the subsumption architecture [19] shown in Figure 4.3. A subsumption architecture works by directing sensor inputs to encapsulated control units representing the internal processes responsible for controlling outward behavior. The outputs of these units either feed into more complex functions or, ultimately, result in actionable vectors in response to the stimuli provided. These control units are organized into layers based upon the level of complexity associated with the behavior. All of the tiers for a specific behavior converge to one point, therefore the architecture must specify which outputs have priority, overriding lower priority

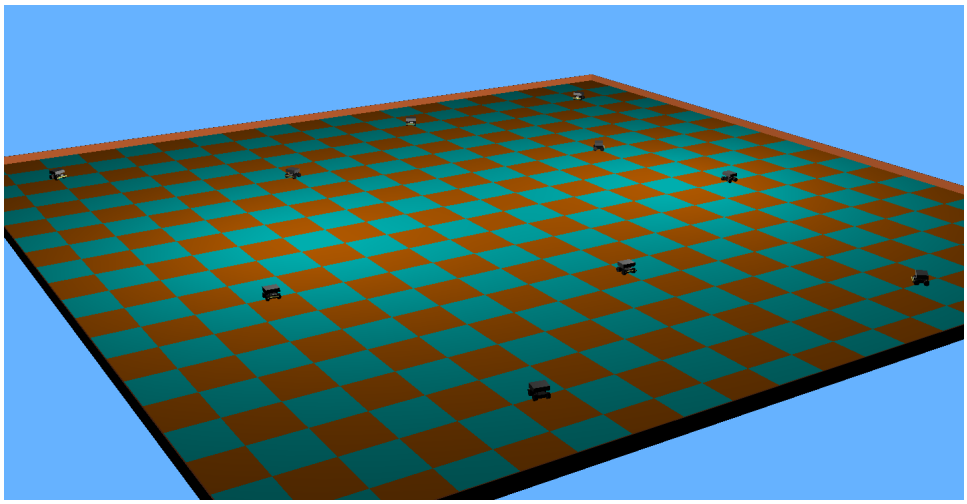


Figure 4.2: A screenshot of the arena with 10 corobots deployed

directives. In our system the robot's primary directive is to move towards a goal point generated by the task allocation system. However, other sensory data may override this edict by manipulating wheel speeds in order to avoid a detected obstacle, or, as a proactive measure, to prevent a collision with another robot in the vicinity.

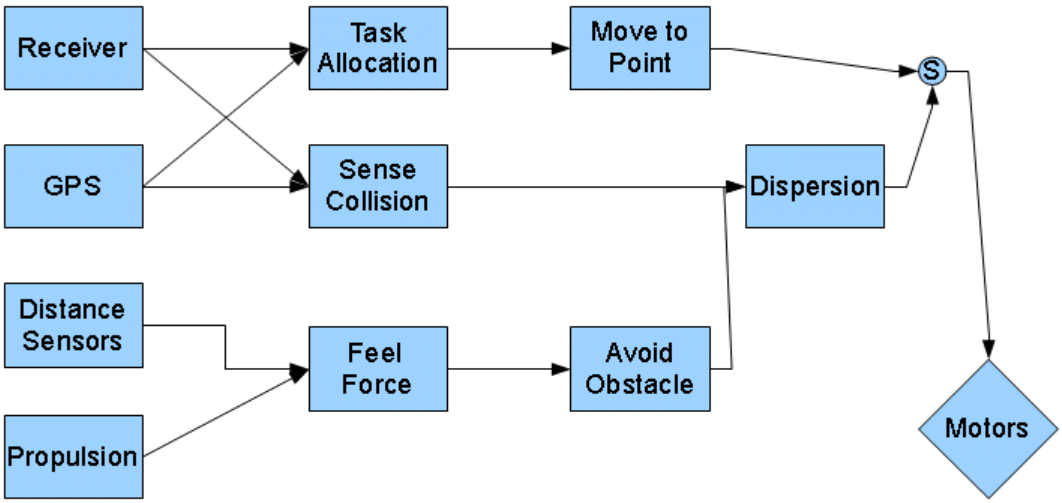


Figure 4.3: Generic subsumption architecture for a Corobot

Directional movement is split into two logical hemispheres (each of π radians) with respect to the current heading as it relates to the underlying grid's coordinate system.

The robot is guided by adjusting wheel speeds; a positive differential between the right wheels and the left veers the robot to the left and vice versa. The magnitude of this differential is a factor of the distance to the goal in conjunction with the measure of the angle separating the robot's heading from the straight line path to the goal coordinate. This factor is varied in such a way as to ensure a smooth transition to a direct bearing on the target.

The Pro version of Webots allows for a supervisor node which can control the simulation in code. We employ this option in order to execute batches of simulations without human intervention.

4.1.1 Collision Detection and Avoidance

Although the infrared sensors, coupled with a Braitenberg controller, work well in detecting stationary obstacles, they are ineffective at discovering moving objects in close proximity. Furthermore, as all robots share the same plane and visit identical task regions, collisions are an inevitable side effect of operation. To mitigate this side effect a collision prediction and avoidance system must be installed in the hierarchy that is the subsumption architecture. But, since this study is focused on task allocation a fairly primitive collision prediction algorithm was contrived, although, as I discuss in the future work section, I believe a neural net based solution could prove advantageous to the scenarios visited in our work.

Our collision prediction system relies on inter-robot communication for actuation. Intermittently, throughout the lifetime of a simulation, each robot broadcasts position and heading data on the inter-robot communications channel. Upon receiving these signals, the information garnered is utilized in determining the closest teammate. Once ascertained, an intersection point between the two trajectories is calculated, and the estimated time of arrival to this point is extrapolated using current velocity estimations. If the result of these calculations is within a certain threshold a collision

is deemed likely and evasive action is triggered.

4.2 Algorithm Comparison

For comparison purposes, we have implemented and simulated three other algorithms in addition to the spatial queue and transition matrix based solution we showcase in this document. Representing the optimal case, we have recast the Hungarian algorithm [21] to a dynamic setting, the result of which is an offline schedule tailored to each robot. These offline schedules are then read into and simulated on the Webots platform using the same environments as our online solution packages. Our first heuristic draws inspiration from the theory of vacancy chains and is in essence a distributed greedy approach to the task allocation problem. Next we adapted and implemented Luo's recent work on repeated auctions with task grouping constraints, also a decentralized approach. While this latter work expands the view of task allocation to that which must consider sets of tasks manifested in a time-extended manner, they regard each set as independent whereas we allow for interrelations between groups and recognize that a versatile solution may very well have to negotiate a setting that distends beyond the constraints put forth in their document.

4.2.1 The Hungarian Algorithm

As a baseline for comparing against our other methods, an optimal schedule was computed offline for every combination of tasks, robots, and environments and then read into a Webots project and simulated using the same motion controller code shared across all of our implementations. I used a public version of the matrix interpretation of the Hungarian algorithm in Java, modifying the source to accommodate idiosyncrasies present in a time-extended scenario that are unaccounted for in the basic Hungarian method. Specifically, the Hungarian method does not produce an

optimal selection when tasks and robots can both be off limits due to availability or completion status.

A complete schedule is built for every robot by simulating each environment file from start to finish. The beginning state consists of the robots in their starting positions along with a cost matrix composed of the Euclidean distances between every robot and every task. An initial allocation is provided by a call to the Hungarian method followed by an iteration, defined as the closure between the minimum robot-task pair. The distance traversed by this closure is subtracted off of the distances of any other active robot-task pairs. At this point, if there are any idle robots another call is made to the Hungarian function and the procedure loops until all objectives are met. During this process a schedule, including any idle states, is composed for all robots and written to a file upon termination, identified by the environment descriptor.

A separate Webots world, built for the purpose of executing these schedules, forces the robots to choose tasks in the order presented to them in their individual schedule, while the other rules of the simulation remain intact.

4.2.2 Decentralized Greedy

Our first heuristic organizes the worker robots in a distributed manner and allows them to greedily select tasks based on distance. During the initialization phase all participating robots receive a copy of the task list from the master robot, a role only present during this short time. The master robot reads the task data from disk and begins a call to action dialogue with any other robots present in the simulation. Those that respond according to protocol are counted in and will receive the task matrix wirelessly at the appropriate time. Once initialized each robot maintains all requisite data individually and is fully capable of acting on its own behalf, based on the data it collects.

All idle robots submit a bid to the collective by communicating the distance to the closest task from its current position. Once all bids are collected each robot can determine whether it is clear to execute its chosen task based on a set of precedence rules. These rules form a hierarchy by which the lowest bid is always allowed to proceed first. After the lowest bid has been recognized by an acknowledgment from the winning robot, the the robot with the second best bid is free to accept its task after first verifying that it was not allocated to the first robot. As this progression continues if a robot finds that its preferred task was already awarded it resubmits a bid: the distance to the next closest task. In this way either all robots are either allocated to a task or, if all tasks are awarded, the remaining robots must sit idle and await the completion of some task so that it may again submit a bid. This process is summarized in the algorithm 2.

Algorithm 2 Decentralized greedy task allocation using auctions

```
1: Input: T = { τ_i : 1 ≤ i ≤ n }
2: while an incomplete task exists do
3:     Set bid as the distance to the closest available task
4:     Set task to the task number of the closest task
5:     if an available task was found then
6:         sendBroadcast(BID, bid)
7:     else
8:         wait until task-complete signal received
9:     end if
10:    wait until competing bids received
11:    if my bid is the global minimum then
12:        sendBroadcast(ACCEPT, task)
13:        Execute task
14:    else
15:        wait for ACCEPT signal
16:    end if
17: end while
```

4.2.3 Repeated Auctions

In [18], the authors perform a competitive analysis of a repeated auction algorithm and treat groups of tasks in a time-extended manner, with the condition that all tasks in one group must be satisfied before approaching the next wave of tasks a method for modeling dependencies between tasks. We have adapted the repeated auction algorithm in this same vein relieving some of the constraints so as to adequately model the situation we are investigating. In our version, as tasks become available new auctions begin immediately in order to allocate free tasks to idle robots.

The repeated auction algorithm works as follows: each robot determines the utility value of every task by subtracting their private value of the task from the current maximum price of that task (set to zero initially). The private value is simply the inverse Euclidean distance between the robot's position and the coordinates representing that task. A bid is then formed by finding the difference of the two highest utilities, adding a small offset, and adding this result to the current price. As this is a distributed model each robot must track these prices individually via broadcast communication. If outbid, this process is repeated until the maximum utility value is negative, at which point the robot has nothing to gain and will remain idle and wait for the next auction. When a stable allocation develops, the auction terminates and the winning robots begin executing their tasks. Reference Algorithm 3 for pseudocode of the technique.

4.3 Experimental Setup

Simulations for all four approaches were centered around three Webots worlds containing either 5, 10, 15, or 20 Corobot robots. In each world the robots are statically deployed, meaning that they begin every simulation at exactly the same location in space. This allows us to compare the performance of the different algorithms in a systematic

Algorithm 3 Repeated Greedy Auctions for Online MRTA [18]

Input: $T = \{ \tau_i : 1 \leq i \leq n \}$
while an incomplete task exists **do**
    for all unoccupied tasks **do**
        Set *value* $\leftarrow$ the inverse distance from my position and the task
        Set *utility* $\leftarrow$ the difference between *value* and the task's current price
        Store task IDs and utilities for the highest and second highest utility values
    end for
    if an available task was found **then**
        bid $\leftarrow$ price + (highest - second + ϵ) $\triangleright$ ϵ is a small constant to ensure the bid price increases
        sendBroadcast(BID, highestID, bid)
    else
        wait until task-complete signal received
    end if
    wait until competing bids received
    if I am the high bidder **then**
        sendBroadcast(ACCEPT, highestID)
        Increment price by: highest - second + ϵ
        Execute *highestID*
    end if
end while

manner. The task allocation algorithms operate on environments containing 6, 12, 18, or 24 tasks read from disk at the start of execution. Ten environments each were generated for all four tiers of tasks, and these same 40 environment files were used for every combination of algorithm and Webot world.

Each environment definition consists of one record for every task in its tier level, where a record contains the x and z coordinates of the task epicenter and the number of required inspections to complete the task. Task coordinates were randomized to the hundredths place with two guidelines: first, that tasks must be at least one meter from the walls bounding the physical limits of the arena, and second, that tasks are a minimum of two meters apart. The required number of inspections to complete a task vary between 3 and 5 for all tiers.

An optional feature, and one that may be investigated in future work, is to assign a weight or priority to each task and to incorporate this additional variable into the task allocation calculations, but for the sake of these proceedings the priority is assumed to be identical for every task under consideration.

4.4 Results

A multitude of data was collected from individual log files written upon controller termination; each containing measurements and sensor data compiled over the course of each run. The list of metrics includes total time in simulation, total time idle during simulation, distance traveled in meters, estimated battery life (experimental), bytes sent and received, messages sent and received, and tasks completed. The data was extracted from each robot, for each of the 10 environments, over all combinations of tasks and robots and then averaged first at the environment level and then again at the robot-task pair level resulting in 16 robot-task combinations for each of the four methods studied.

One of the key metrics that must be evaluated in the domain in question is the total time required to complete all inspections, or completion time; this value is defined as the seconds elapsed when the final inspection is completed and the last remaining robot terminates its controller. This value is simply the maximum simulation time of all robots in an environment. When comparing the four methods side by side there are two viewpoints to consider: the first is to leave the number of robots fixed and look at the results as the task load increases, the second is to fix the number of tasks and vary the count of robots available to perform the work. Figure 4.4 charts completion times for each of the four methods with robot levels fixed and task loads variable.

Although our spatial queuing approach does not stand out on its own, it does match the results of repeated auctions and they both showcase the versatility to perform efficiently in all the scenarios presented; whereas the Hungarian and decentralized greedy approaches both exhibit weaknesses when tested, the repeated auction and spatial queuing methods seem to have none.

Studying these results illuminates a couple of interesting conclusions. The first being that the decentralized greedy approach is very inefficient when the number of tasks greatly outweigh the count of robots in simulation. In Figure 4.4a, where the robot count is 5, the decentralized greedy approach gets progressively worse as more tasks are introduced to the system. As the task to robot ratio grows, a greedy approach becomes ineffective because it cannot cope with the number of possible trajectories available for completing all outstanding task requirements; by definition it can only superficially select the best allocation for the immediate iteration and the constraints therein. It is only by adding more robots to the simulation that this approach mirrors the efficiency displayed by the repeated auction and spatial queuing methods. By Figure 4.4d, when the robot count is 20, there is little statistical difference between these three methods. However, the opposite can

be said for the Hungarian method - at first it is producing results very similar to the repeated auction and spatial queuing algorithms, but as additional robots are introduced it performs poorly. This can be attributed to the fact that the schedule is scripted offline and therefore the algorithm has little ability to adapt to change materializing dynamically during run time (additional robots complicate the simulation by increasing the probability of collision avoidance events).

The inefficiency of the decentralized greedy system with five Corobots can be seen clearly in Figure 4.5a, which graphs the competitive ratio of completion time, not against the other methods, but against itself with a baseline task load of six. With five available robots the decentralized greedy algorithm nearly triples in running time with a task load of 24, whereas the other three methods accomplish the same task load at a multiplier of only slightly over 1.5 times the baseline of 6 tasks.

As is to be expected in Figure 4.4, overall completion time increases as the task load increases; similarly, the opposite is expected when the number of tasks is fixed, but additional robots are added to the system. This is precisely what Figure 4.6 conveys via charts of completion time from the perspective of task load. Generally, what can be observed from these results is that the jump from 5 to 10 robots offers the most improvement in efficiency, after which the interference overhead introduced by additional workers limits the gains or even decreases the system efficiency. In Figure 4.6a (6 tasks), an average decrease in execution time of 35% is visible when jumping from 5 to 10 robots, however 15 and 20 robots offer almost no advantage and growth is flat (plus or minus 4%). At 12 tasks, Figure 4.6b, efficiency gains are visible with 10 and 15 robots. The average decrease in execution time nearly doubles from 15% with 10 robots to 29% with 15; an additional decrease of 5% accompanies an increase to 20 robots. Conversely with task loads of 18 (Figure 4.6c) and 24 (4.6d) growth is flat or negative when 15 or 20 robots are present. With 18 tasks completion time decreases by 25% on average when jumping from 5 to 10 robots; 15 robots offers

no advantage and 20 robots actually performs worse by 4%. A task load of 24 is completed 24% faster by 10 robots than 5; 15 robots complete this same task load 2% slower than with 10 and 20 robots slip another 4% as compared to 15 robots. A possible explanation for this anomaly is ideal localization of tasks with respect to robot deployment points; it should be noted that task groups are not super or sub sets of one another - each grouping in unique and independent.

It should also be noted that both our spatial queuing algorithm and repeated auctions, show their worth by outperforming the greedy heuristic in scenarios that require multi-stage path planning in order to visit all the tasks. This happens when the ratio of robots to tasks is less than one. As the ratio of robots to tasks approaches 1 and beyond, the greedy algorithm closes the gap and produces similar results. At a ratio of 1 or more, every task can be assigned to the closest robot and there is less efficiency to be gained through path planning. Examples of the benefits of path planning are visible in Figures 4.6b, 12 tasks 5 robots (23%), 4.6c, 18 tasks 5 and 10 robots (31% and 14% respectively), and 4.6d, 24 tasks 5, 10 and 15 robots (37%, 38%, and 26% respectively) by calculating the average improvement in completion time using the greedy results as a baseline. The Hungarian algorithm, as mentioned before, cannot respond to change and therefore performs poorly in simulation.

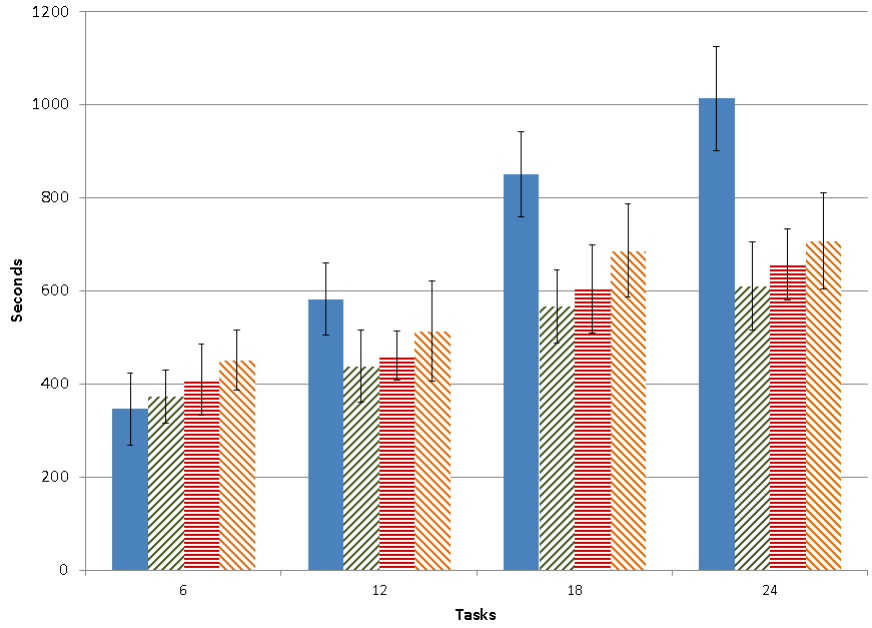
The competitive ratios for completion time of 10, 15 and 20 robots (as compared to 5 for each method) are charted in Figure 4.7. These show that, except for 6 tasks where the number of robots greatly outnumbers the tasks, the decentralized greedy heuristic improves the most with supplemental members (an average overall gain in efficiency of 35%) while at the same time the Hungarian method improves the least (15%).

Average distance traveled per robot is another key metric when analyzing overall system fitness. Minimizing the distance traveled in turn decreases the usage of a robot's power source, which is a valuable commodity in most real life environments.

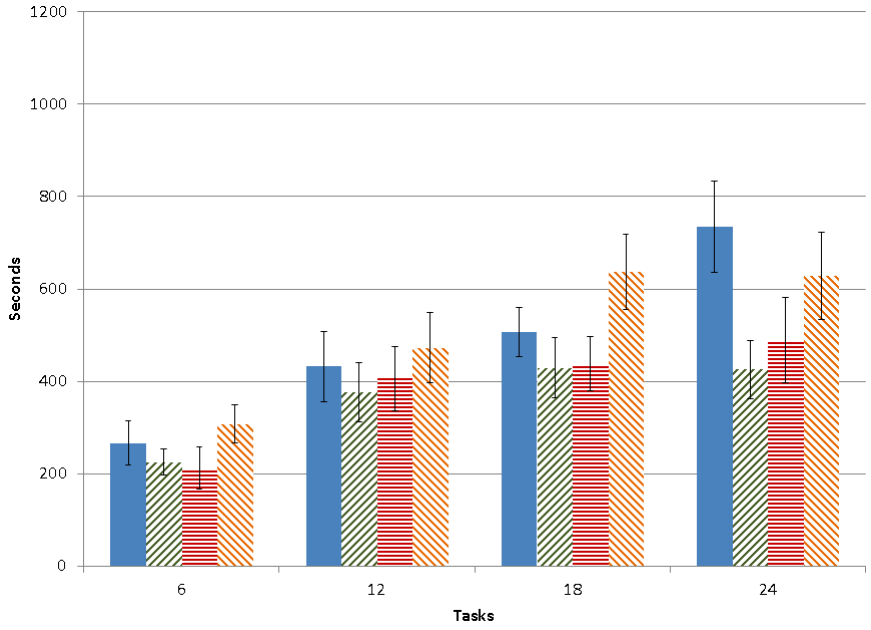
Figure 4.9 charts the average total distance traveled per robot, with 5, 10, 15 and 20 robots, as the task load increases from 6 to 24. The average total distance traveled per robot is directly proportional to the number of tasks and inversely proportional to the number of robots, or in other words distance increases with task load and decreases with the number or robot candidates. The data shows quite clearly that the two heuristics and the Hungarian optimized schedule perform comparably in all scenarios (overall average distance of 50 meters for the repeated auction method and 53 meters for both spatial queuing and Hungarian); the greedy algorithm, however, operates less efficiently as the task load is increased, culminating in a gap of approximately 60 meters in the worst case (5 robots, 24 tasks), its overall average distance traveled was 65 meters. As we witnessed with completion time, the greedy algorithm does improve with the addition of worker robots, and this can be seen in Figure 4.9b where its performance is in line barring a slight jump in the 24 tasks case.

Whereas completion time refers to the average total time to complete all inspections, average simulation time refers to the time in simulation for each individual robot. Since the robots in the Hungarian implementation are following a schedule they can terminate immediately upon completing their assigned tasks - they do not need to remain idle in preparation for performing more work on demand. Therefore, the average simulation time for the Hungarian implementation can be used as a benchmark for comparing the three heuristic algorithms. Their average simulation times will be some multiple of this benchmark - a competitive ratio. Figure 4.10 graphs the evolution of this metric for all sixteen combinations of robots and tasks. Again we see that the decentralized greedy tack cannot compete in many situations that benefit from path planning (its overall average competitive ratio is 1.47 whereas repeated auction's and spatial queuing's are 1.25 and 1.27 respectively), but that it does fall more in line as the robot to task ratio borders on one (1.45 versus 1.38 and 1.37). Also visible in the progression are the large spikes at 15-6 and 20-6 (1.86 and

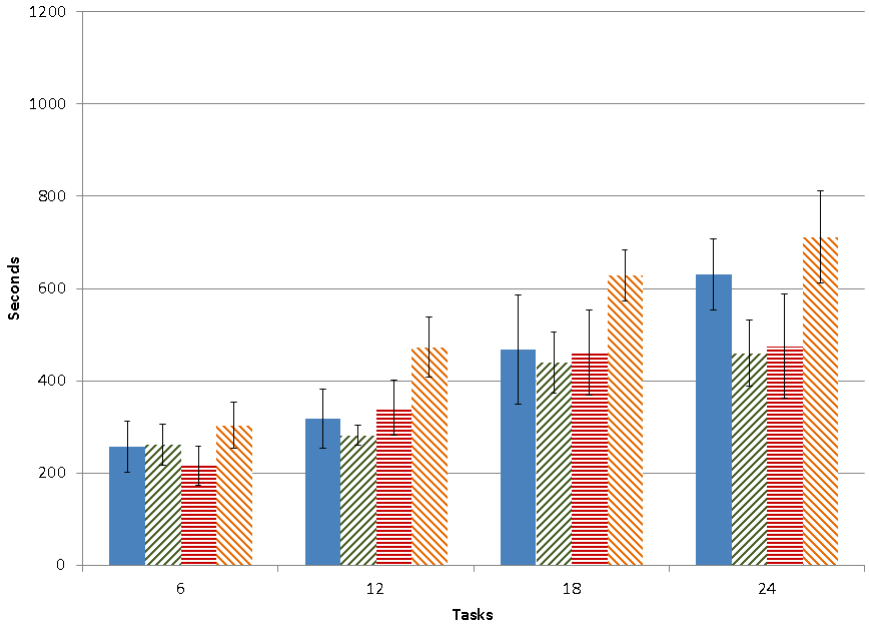
1.97 compared to an average of 1.24 for all other scenarios). This can be explained by the fact that many robots sit idle never winning any tasks while these same robots terminate immediately in the Hungarian case because their schedule is empty. Apart from these two spikes, spatial queuing and repeated auctions maintain a competitive ratio just above the benchmark; that is until the 20 robot cases at which point the ratio noticeably expands (an average of 1.57 in 20 robot configurations as compared to a composite average of 1.25 for 5, 10 and 15 robot environments).



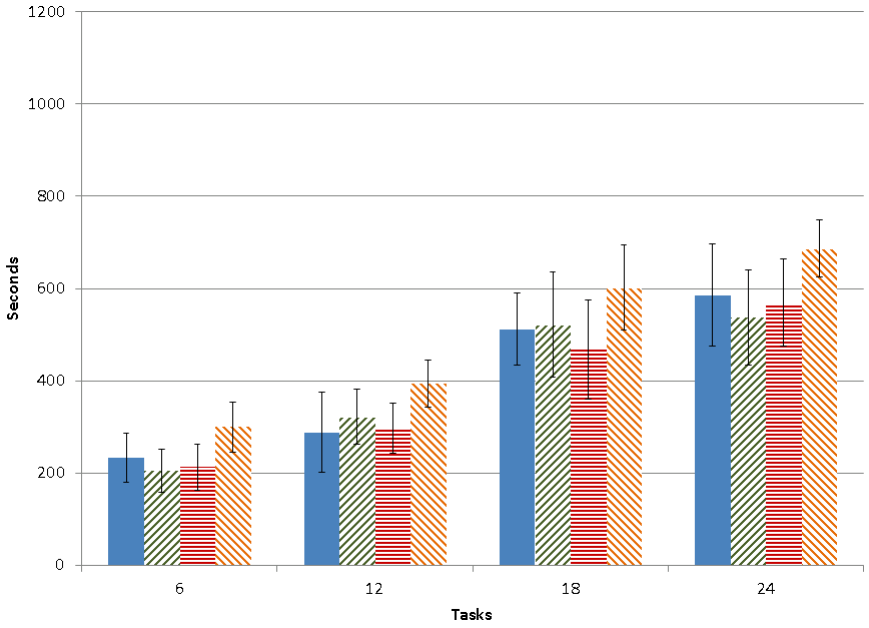
(a) 5 robots



(b) 10 robots

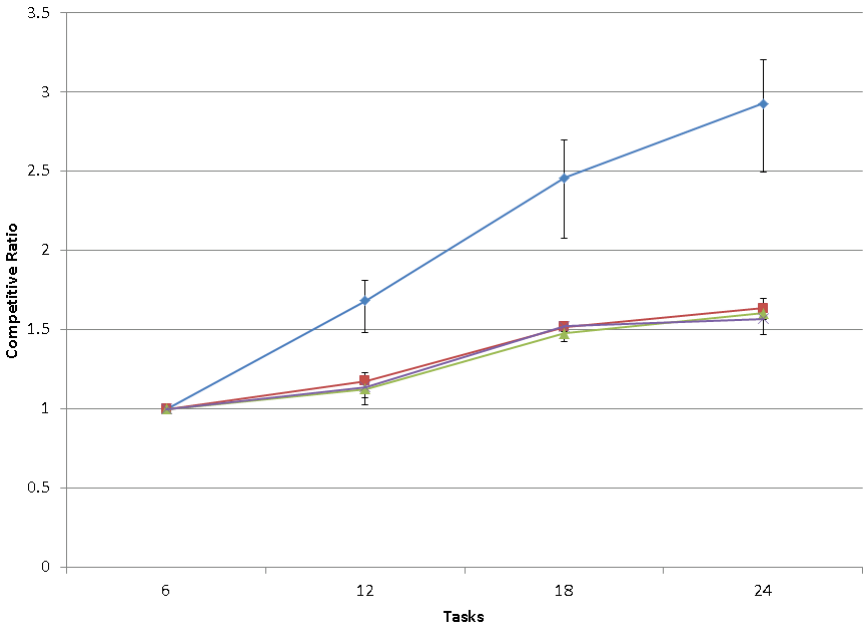


(c) 15 robots

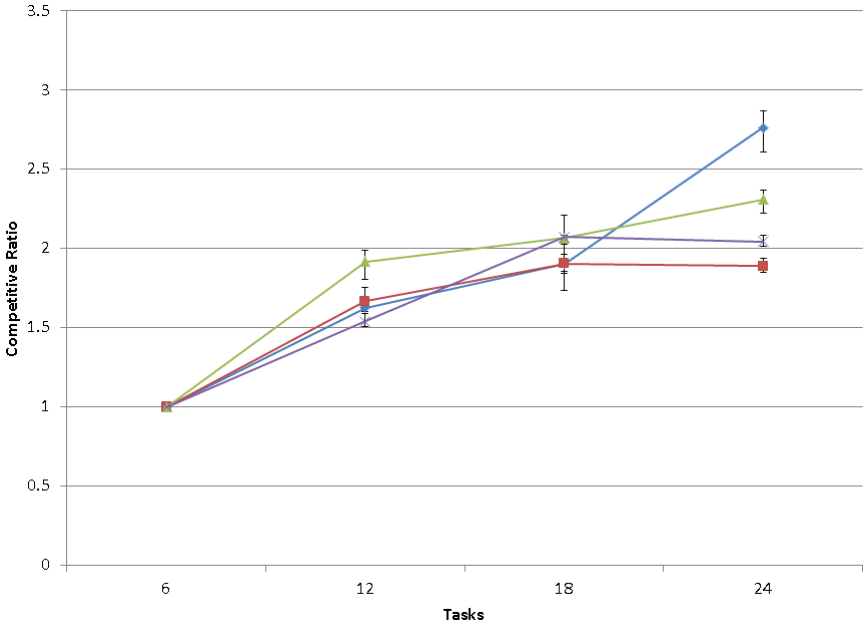


(d) 20 robots

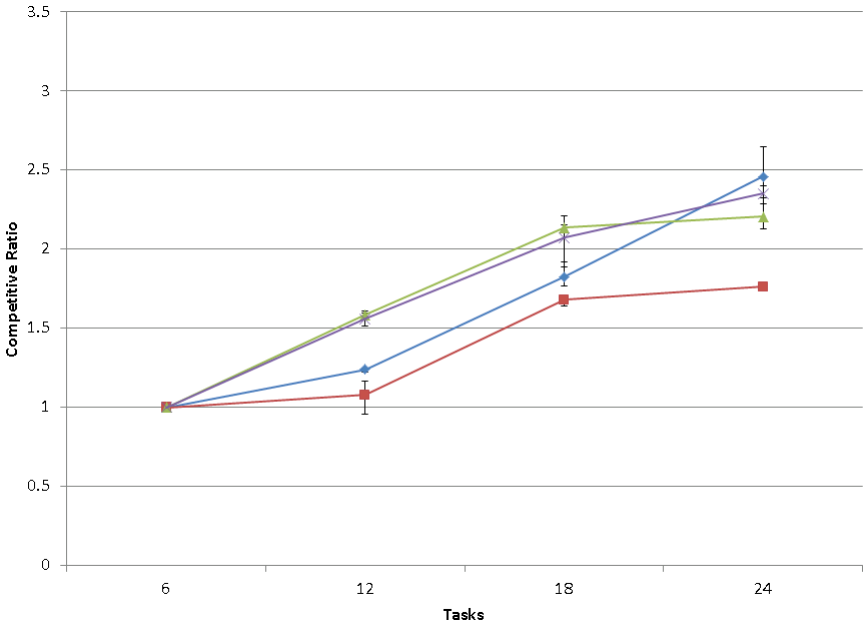
Figure 4.4: Completion times with robot levels fixed for 6, 12, 18, and 24 tasks



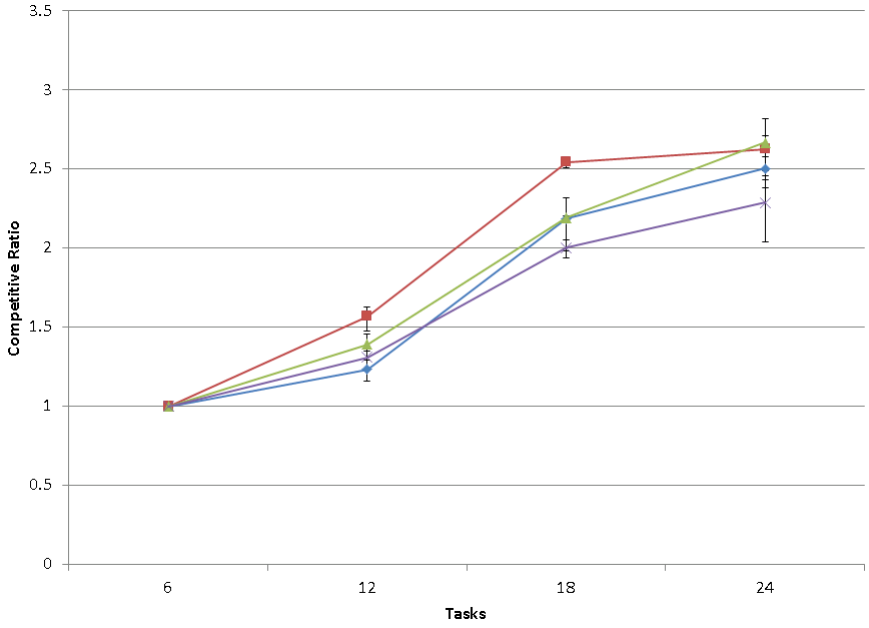
(a) 5 robots



(b) 10 robots



(c) 15 robots



(d) 20 robots

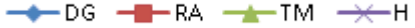
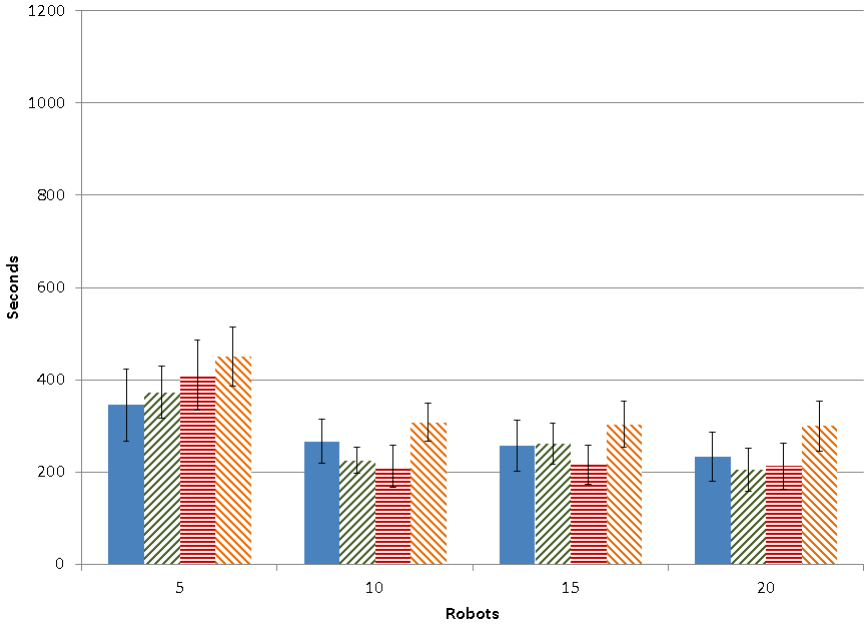
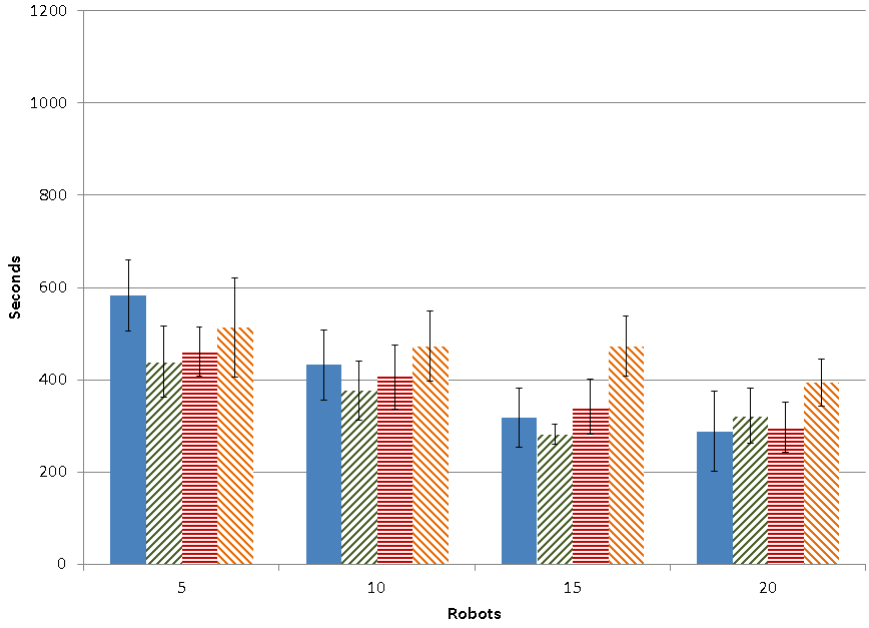


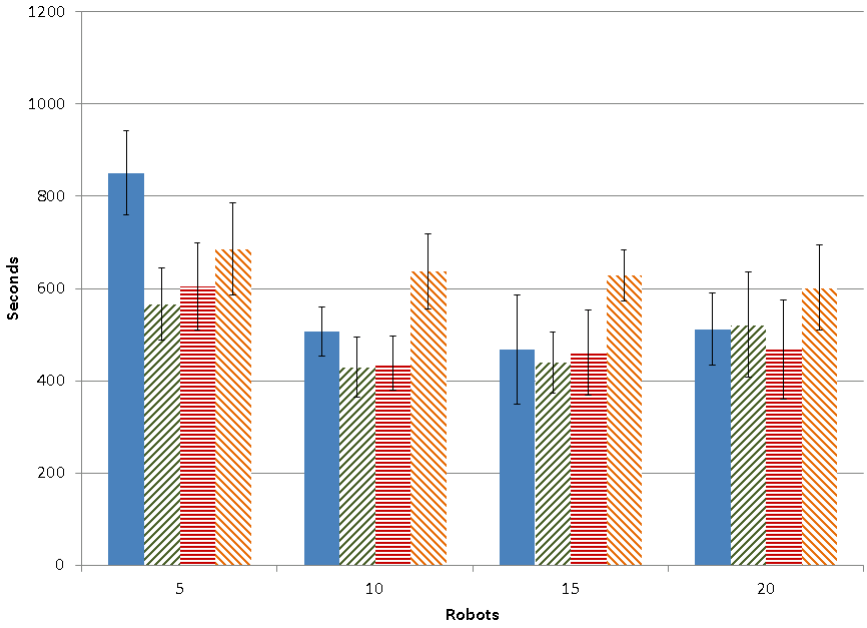
Figure 4.5: Competitive ratio of completion times for 12, 18, and 24 tasks as compared to 6 tasks



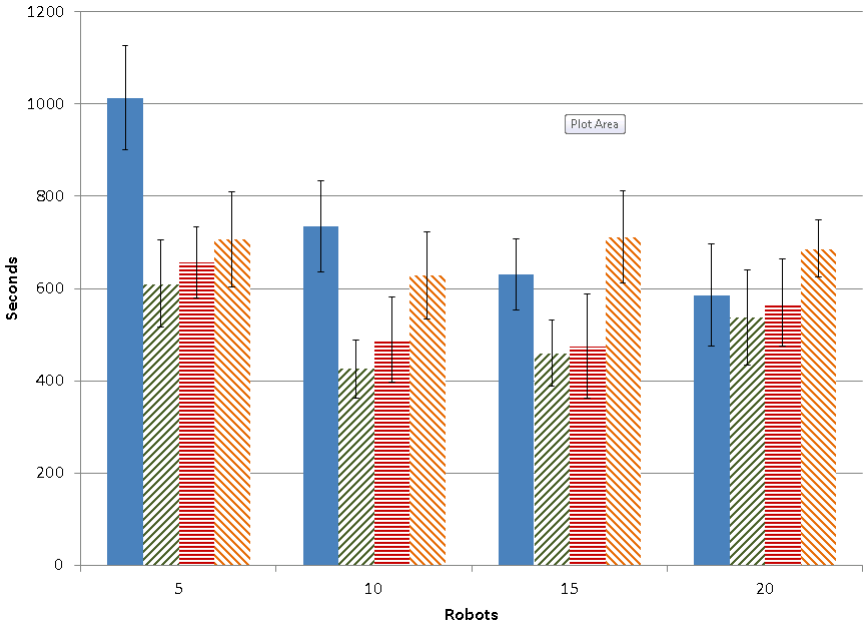
(a) 6 tasks



(b) 12 tasks

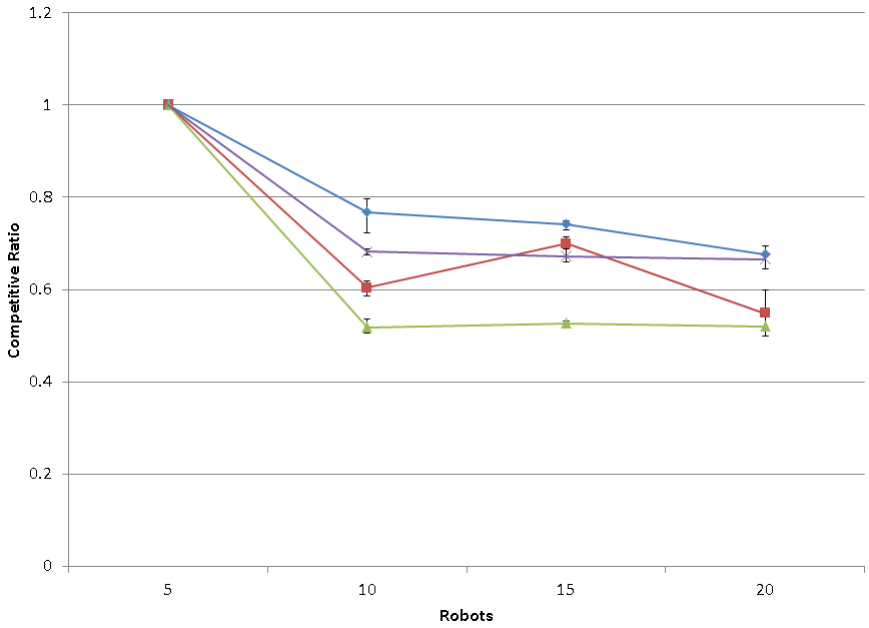


(c) 18 tasks

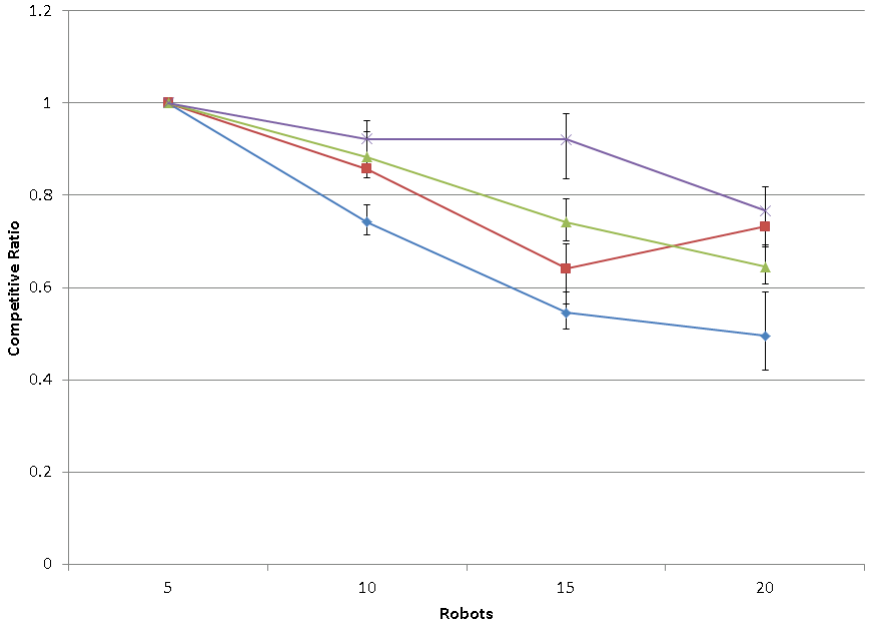


(d) 24 tasks

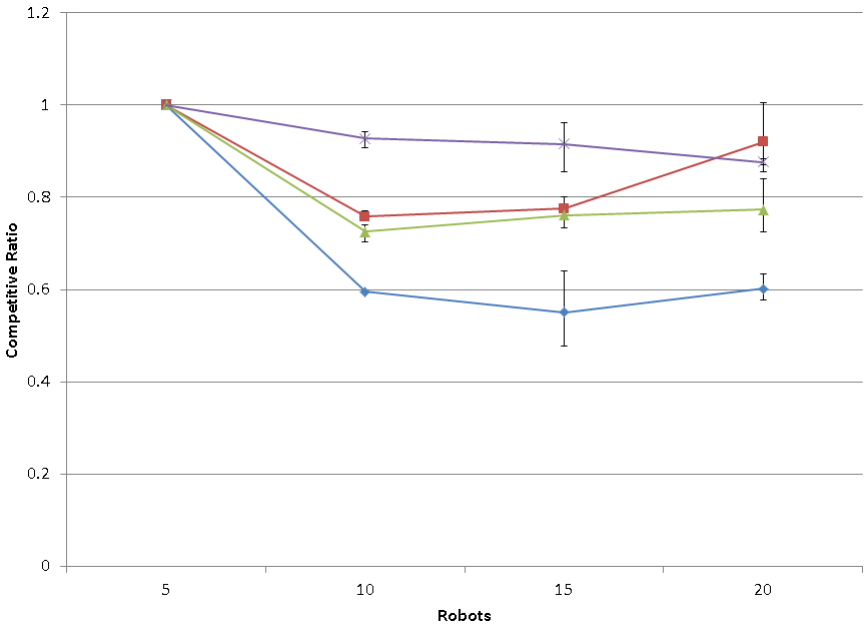
Figure 4.6: Completion times with task load fixed for 5, 10, 15, and 20 robots



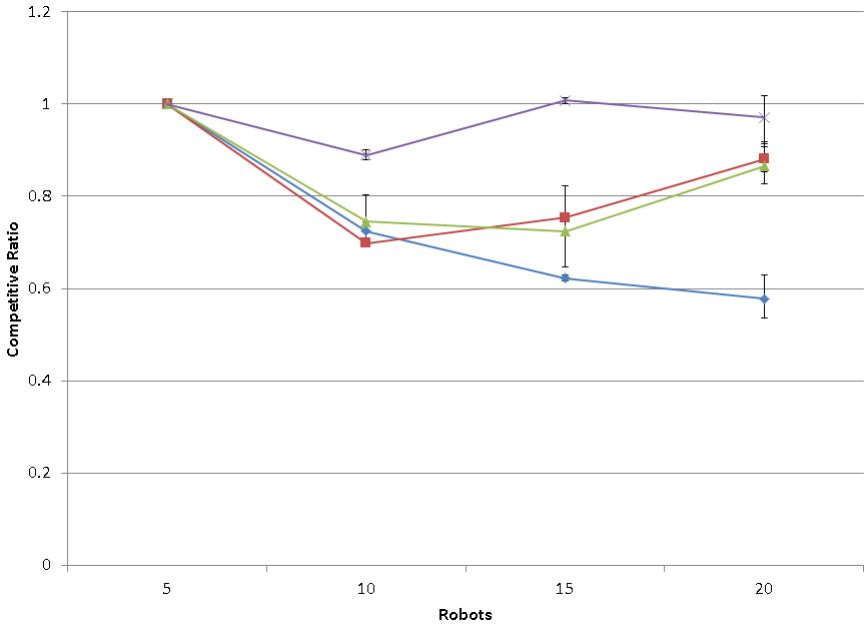
(a) 6 tasks



(b) 12 tasks

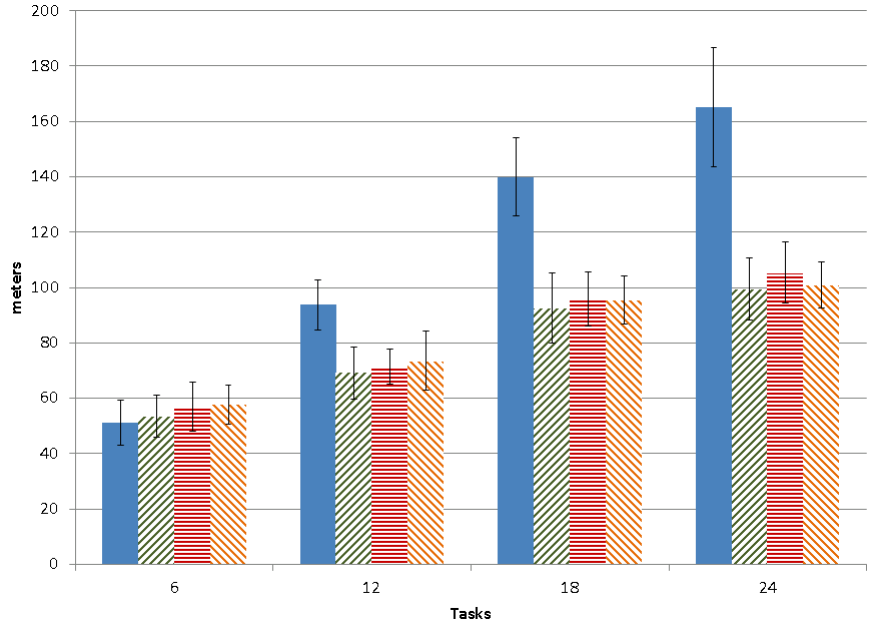


(c) 18 tasks

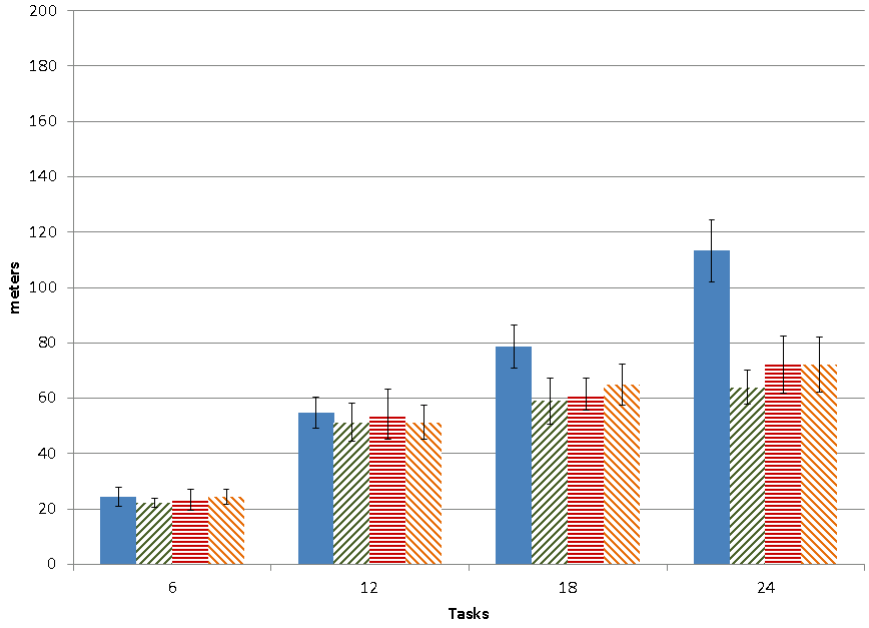


(d) 24 tasks

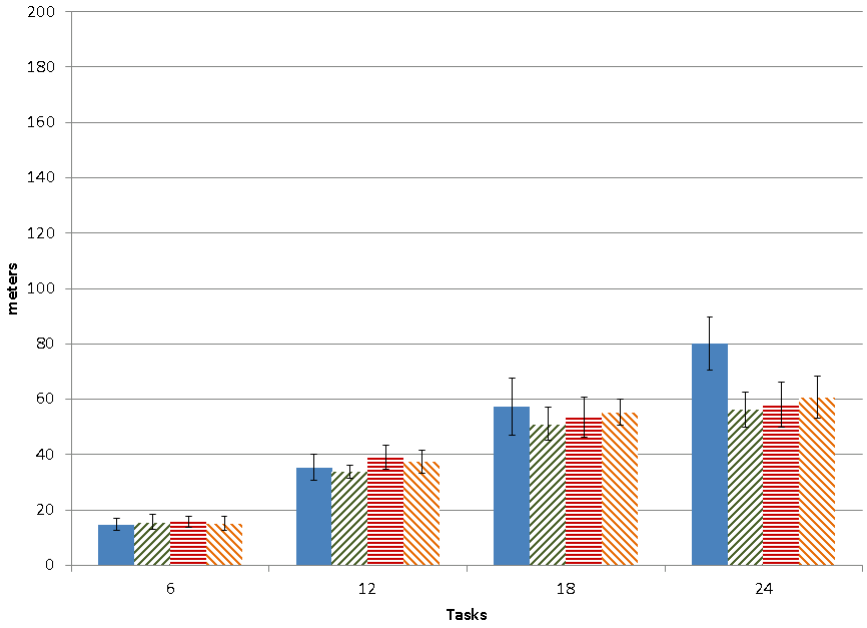
Figure 4.7: Ratio of completion time as robots are added to the simulation



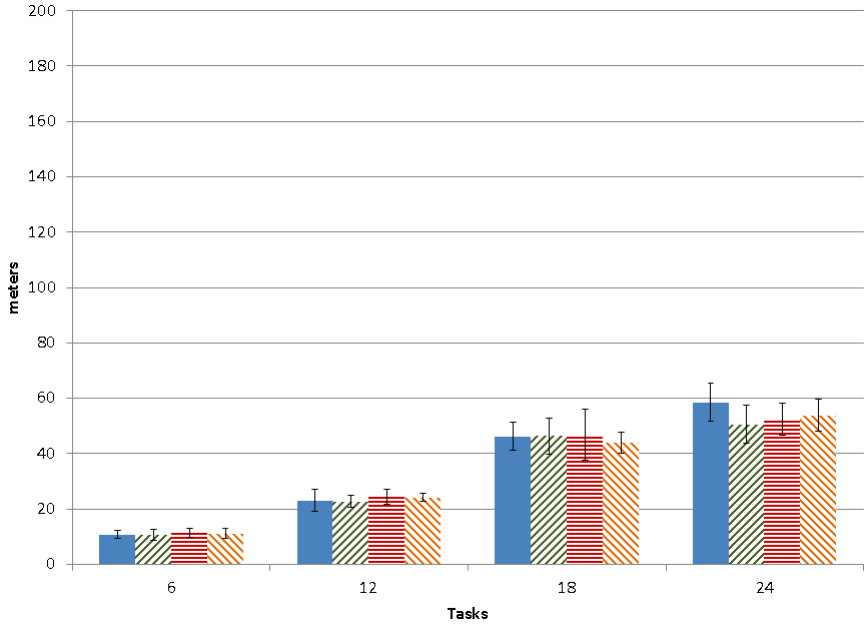
(a) 5 robots



(b) 10 robots



(a) 15 robots



(b) 20 robots

Figure 4.9: Overall distance traveled in meters

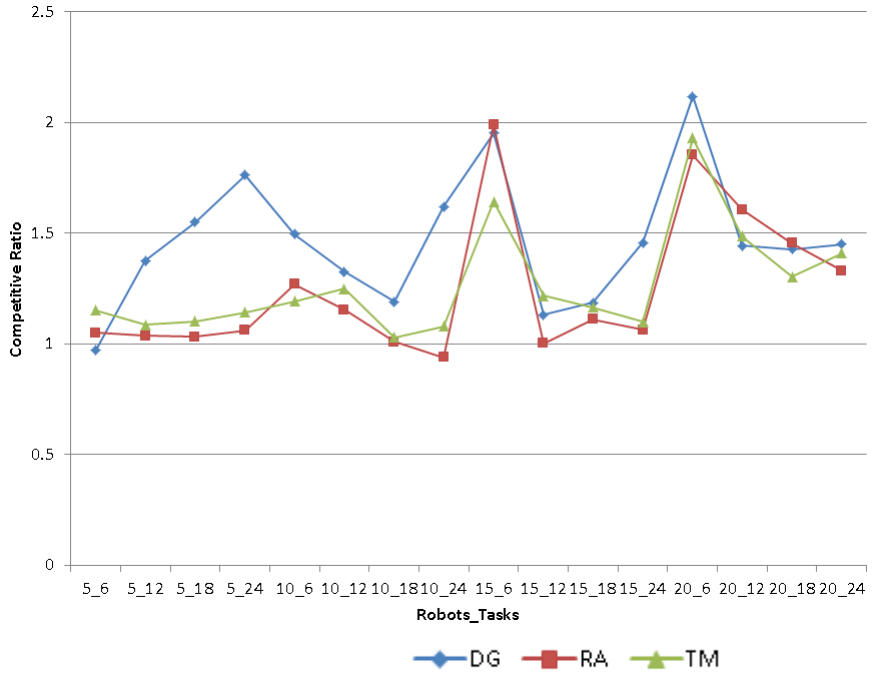


Figure 4.10: Competitive ratio of simulation times using the Hungarian method as the baseline

Chapter 5

Conclusions and Future Work

5.1 Conclusions

We proposed a spatial queuing algorithm with integrated look-ahead as a solution for multi-robot task allocation in the context of automated landmine detection. Our completed system was simulated on accurate models of the Corobot robot using the Webots simulator for many combinations of robot teams, task loads, and task environments. For comparison purposes we also implemented controllers for a decentralized greedy heuristic, a repeated auctions approach, and an optimized offline schedule based on the Hungarian algorithm.

Our results show that an offline schedule, while optimal in theory, does not necessarily translate to a real world scenario where small disturbances in the environment can completely change the entire schedule of events that take place. In addition, the results clearly indicate that a greedy heuristic is inefficient in situations that require managing a small team of robots with a large task load; only when the the team size approaches the task load does the greedy algorithm match the performance of repeated auctions or our spatial queuing theory.

Our spatial queuing method performed well in all scenarios and right in line with

the efficiency of a repeated auctions implementation. Analysis of the results also aids in selecting the appropriate team size for a physical environment, if modeled in enough detail. In our basic representation of a bounded region the ideal team size appears to be 10, after that the complications and overhead of another five robots begin to erode the overall system performance. This study is by no means an exhaustive search into this query, but it does indicate that there would be advantages to be gained by doing so for a specific implementation in a real world project.

5.2 Future Work

There are many directions this work can be taken in and improved upon. An obvious one, after perhaps expanding the simulation parameters, is to run the finished product on actual hardware in a controlled physical environment. There are many assumptions in a simulated work space that need tweaking in a live one; communication reliability being a substantial one as this is a requisite subsystem in a distributed, collaborative effort. It is also worth noting that we have assumed accurate localization of the robots based on a simulated GPS receiver. This is paramount to an effective solution and incorrect localization would result in poor performance in actual hardware. Also related to hardware is the possibility of a heterogeneous team instead of the homogeneous group assumed in this study. It may be beneficial to include robots with varying skill sets and sensor capabilities, especially when the primary goal is to detect buried objects. As such, the confidence levels of an inspection by a particular robot may need to be graded, therefore changing the foundation of the algorithms to include a type of priority based on these confidence levels. Lower confidence levels may require supplemental inspections, possibly by specific sensor types, and higher confidence levels may change the priority of a task so that it attracts team members and focuses work on higher probability regions.

Others ideas that have merit for further investigation are to introduce a minimum number of tasks to allocate at one time, at least within some limited window in time, and also exploring the option of augmenting the transition matrix used in the spatial queue formation with data discovered during run time. With the parameters as they are, often only one task is allocated at a time (except for the first allocation and as the simulation nears completion). Yet, task completion events often occur within seconds of each other, therefore more adept allocations may be possible by allowing for small delays with the hopes of reaping a return on the investment in time. In regards to the idea of an augmented transition matrix, the priorities mentioned above could be integrated into the matrix by weighting the combination of distance and priority and recalculating affected regions of the matrix with this new information.

One final area of focus for future work, though not directly associated with task allocation, is a more sophisticated path planning for obstacle avoidance. The current method is to predict straight line trajectories based on position information broadcast periodically throughout the life span of the controller. I propose that a neural network based solution, in conjunction with a proper training period, could arm each robot with the a coefficient set tuned to react directly to position signal messages. The neural net would have input nodes correlating to the velocity and heading of both the subject and a possible target along with the distance between them. These nodes would feed forward to an intermediate layer followed by an output layer of two nodes whose values would be interpreted as multipliers for effecting the left and right wheel speeds. In addition, more sophisticated path planning like [23] could be used to address the inter-robot collision avoidance problem.

In this thesis, we have proposed a spatial queuing approach to MRTA; one that incorporates a measure of look-ahead constituting a primitive form of path planning. This feature, along with the decentralized aspect of the system, allows our approach to efficiently respond to a dynamic, unknown environment; to adapt and control team

resources within the environment and within the context of the goals appointed to the collective.

Bibliography

[1] B. Gerkey, M. Mataric, "A Formal Analysis and Taxonomy of Task Allocation in Multi Robot Systems," Intl. Journal of Robotics Research 23(9), 2004, pp. 939-954.

[2] B. Gerkey, M. Mataric, "Sold!: Auction Methods for Multirobot Coordination," IEEE Transactions on Robotics and Automation 18(5), 2002, pp. 758-768.

[3] E. Gil Jones, B. Browning, M. Dias, B. Argall, M. Veloso, A. Stentz, "Dynamically Formed Heterogeneous Robot Teams Performing Tightly-Coordinated Tasks," Proc. International Conference on Robotics and Automation, 2006, pp. 570-575.

[4] E. Gil Jones, M. Dias, A. Stentz, "Learning-enhanced Market-based Task Allocation for Oversubscribed Domains," Proc. Intelligent Robots and Systems, 2007, pp. 2308-2313.

[5] E. Gil Jones, M. Dias, A. Stentz, "Time-extended Multi-robot Coordination for Domains with Intra-path Constraints," Autonomous Robots 30(1), 2011, pp. 41-56.

[6] X. Li, D. Sun, J. Yang, "Networked Architecture for Multi-Robot Task Reallocation in Dynamic Environment", Proc. Robotics and Biomimetics, 2009, pp. 33-38.

[7] L. Liu, D. Shell, "Multi-Level Partitioning and Distribution of the Assignment Problem for Large-Scale Multi-Robot Task Allocation," Proc. Robotics: Science and Systems, 2011, pp. 26-33.

[8] L. Liu, D. Shell, "Assessing Optimal Assignment Under Uncertainty: An Interval-based Approach," Intl. Journal of Robotics Research 30(7), pp. 936-953.

[9] K. Seow, N. Dang, D. Lee, "A Collaborative Multiagent Taxi-Dispatch System," IEEE Transactions on Automation Science and Engineering, 7(3), 2010, pp. 607-616.

[10] M. Pavone, S. Smith, F. Bullo, E. Frazzoli, "Dynamic Multi-Vehicle Routing with Multiple Classes of Demands," Proc. American Control Conference, 2009, pp. 604-609.

[11] G. Celik, E. Modiano, "Dynamic Vehicle Routing for Data Gathering in Wireless Networks," Proc. Conference on Decision and Control, 2010, pp. 2372-2377.

[12] P. Dasgupta, "Multi-Robot Task Allocation for Performing Cooperative Foraging Tasks in an Initially Unknown Environment," Innovations in Defense Support Systems, vol. 338, 2011, pp. 5-20.

[13] L. Liu, D. Shell, "Tunable Routing Solutions for Multi-Robot Navigation via the Assignment Problem: A 3D Representation of the Matching Graph," Proc. International Conference on Robotics and Automation, 2010, pp. 4800-4805.

[14] L. Liu, D. Shell, "A Distributable and Computation-flexible Assignment Algorithm: From Local Task Swapping to Global Optimality," Proc. Robotics: Science and Systems, 2012, pp. 33-41.

[15] K. Zhang, E. Collins, A. Barbu, "An Efficient Stochastic Clustering Auction for Heterogeneous Robot Teams," Proc. International Conference on Robotics and Automation, 2012, pp. 4806-4813.

[16] S. Lim, D. Rus, "Stochastic Motion Planning with Path Constraints and Application to Optimal Agent, Resource, and Route Planning," Proc. International Conference on Robotics and Automation, 2012, pp. 4814-4821.

[17] I. Sucan, L. Kavraki, "Accounting for Uncertainty in Simultaneous Task and Motion Planning Using Task Motion Multigraphs," Proc. International Conference on Robotics and Automation, 2012, pp. 4822-4828.

[18] L. Luo, "Competitive Analysis of Repeated Greedy Auction Algorithm for Online Multi-Robot Task Assignment," Proc. International Conference on Robotics and Automation, 2012, pp. 4792-4799.

[19] Murphy, R., *Introduction to AI Robotics*. Cambridge, MA: The MIT Press, 2000.

[20] Braitenberg, V., *Vehicles: Experiments in synthetic psychology.* Cambridge, MA: MIT Press, 1984.

[21] Kuhn, H., *The Hungarian Method for the Assignment Problem.* Naval Research Logistics Quarterly 2, 1955, pp. 83-97

[22] D. Althoff, J. Kuffner, D. Wollherr, M. Buss, "Safety Assessment of Robot Trajectories for Navigation in Uncertain Dynamic Environments," Autonomous Robots, vol. 32, 2012, pp. 285-302.

[23] V. Desaraju, J. How, "Decentralized Path Planning for MultiAgent Teams with Complex Constraints," Autonomous Robots, vol. 32, 2012, pp. 385-403.