12-2016

Multi-robot Task Allocation using Agglomerative Clustering

Maria Shoaib
mxs8942@rit.edu

Multi-robot Task Allocation using Agglomerative Clustering

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Zack Butler, Supervisor

Dr. Raymond Ptucha, Reader

Dr. Hans-Peter Bischof, Observer

Multi-robot Task Allocation using Agglomerative Clustering

by

Maria Shoaib, B.S.

THESIS

Presented to the Faculty of the Golisano College of Computer and Information Sciences
Rochester Institute of Technology
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Computer Science, Department of Computer Science

Rochester Institute of Technology

December 2016

Acknowledgments

I wish to thank many people who have helped and motivated me. First of all, my adviser Dr. Zack Butler for advising, helping, and guiding me throughout the process. The process itself made a significant impact on my graduate school experience. I sincerely benefited from his expertise in the area and deeply appreciate his help in writing several reports (Pre-proposal, Proposal, DARS paper and, this thesis report). In addition to this, his insightful comments on the reports helped me improve my writing style. I would like to thank the other members of my thesis committee, Dr. Hans-Peter Bischof, and Dr.Raymond Ptucha for their time and valuable feedback on all of the reports. I would also like to thank my friends and family for all the support, in particular my mother and best friend, Uzma and father, Shuaib, without whom I would not have made it through graduate school.

...

Abstract

Multi-robot Task Allocation using Agglomerative Clustering

Maria Shoaib, M.S.
Rochester Institute of Technology, 2016

Supervisor: Dr. Zack Butler

The main objective of this thesis is to solve the problem of balancing tasks in the Multi-robot Task Allocation problem domain. When allocating a large number of tasks to a multi-robot system, it is important to balance the load effectively across the robots in the system. In this thesis an algorithm is proposed in which tasks are allocated through clustering, investigating the effectiveness of agglomerative hierarchical clustering as compared to K-means clustering. Once the tasks are clustered, each agent claims a cluster through a greedy self-assignment. This thesis investigates the performance both when all tasks are known ahead of time as well as when new tasks are injected into the system periodically. To account for new tasks, both global re-clustering and greedy clustering methods are considered. Three metrics: 1) total travel

cost, 2) maximum distance traveled per robot, and 3) balancing cost index are used to compare the performance of the overall system in environments both with and without obstacles. The results collected from the experiments show that agglomerative hierarchical clustering is deterministic and better at minimizing the total travel cost, especially for large numbers of agents, whereas K-means works better to balance costs. In addition to this, the greedy approach for clustering new tasks works better for frequently appearing tasks than infrequent ones.

Table of Contents

List of Figures

Chapter 1

Introduction

The Multi-Robot Task Allocation (MRTA) problem is a challenging problem in the robotics domain with numerous practical applications. In the past, several techniques for task allocation have been proposed for numerous application domains such as multi-robot coverage [8], search and rescue [20], surveillance [22], and health care [5]. For example, in multi-robot search and rescue missions, the motive is to explore the surroundings and rescue survivors. Each of the sub-tasks constitutes of exploring a smaller area of the environment and finding survivors. The successful completion of each of these sub-tasks results in the completion of the entire mission. As the size of the survivors (available tasks) and robot teams increase, the process of allocation becomes more difficult [30]. In general, these algorithms operate by selecting robots to perform individual tasks as they arrive, and attempt to increase system performance and reliability in terms of total task execution time and robustness in cases of robot failures. Due to the nature of some complex tasks, these tasks may be even difficult for a single robot to complete in reasonable time. We are interested in reducing the total effort expended by the team and/or the final task completion time by utilizing many robots that are able to balance workload amongst them and work on tasks in parallel.

Many different types of algorithms have been proposed. The most common technique is using market-based methods to allocate tasks to robots. In market-based approaches, the environment is considered as a virtual economy and the robots are agents operating in it. Several pioneering auction algorithms like ALLIANCE, MURDOCH, RETSINA, CBBA, and open agent architecture are market-based [4, 9]. Some techniques like RETSINA and open agent architecture are centralized, working with a single broker [9]. Following the centralized methodologies, distributed methods like the consensus-based auction algorithm, CBAA were introduced. CBAA was intended to solve the MRTA problem of a fleet of autonomous mobile robots [4]. The advantage of decentralized techniques is their resilience. In cases of critical failures, if one of the robots break down, the other robots are capable of executing tasks independently or in teams with others [21]. As opposed to auctions, machine learning techniques have also been proposed. The algorithm makes individual task utility estimates and task assignments to form teams of agents. The planning process includes Markov Decision Processes and Reinforcement Learning based short term and long term planning, which helps in allocating tasks and determine the position of the individual robots. Other approaches include determining computational models and probabilistic inference to define relationships which depict coordination between teams and clustering tasks to balance load in the system. However, in the majority of the papers, less attention is paid to explicitly considering the balanced MRTA problem. In many of the past techniques, only a small number of tasks and robots are used, even in

simulation [7, 10, 31]. While some of the works, particularly the auction based techniques consider environments with obstacles, others like clustering methods do not consider the impact of obstacles on their algorithm, experiments, and results [7, 31]. Many multi-robot planning techniques use single item or multi-item auctions instead of combinatorial auctions because combinatorial auctions make bid evaluation, communication, and auction clearing intractable [9]. Due to this, their ability to provide a quick response to a large number new tasks entering the system is hindered. Although replanning is crucial in dynamic environments only few methods incorporate peer to peer trading of tasks [9].

In the proposed method, we will divide n tasks into k clusters, which will then be assigned to m number of agents. The method will allow for multi-robot missions that support run time allocation for both tasks that are known beforehand and new tasks entering the system, frequently and in frequently. The idea is to cluster similar tasks together based on a similarity metric where each agent decides and allocates a cluster of tasks to itself based on its belief of what is the most beneficial for itself. The metric is arbitrary and can vary for different kinds of problems. For example, for an exploration and mapping problem, where the tasks are well distributed in the geometrical space the euclidean metric would work much better as opposed to the cosine similarity metric. Also, the term beneficial can mean different things for different types of problems — for example, in systems with different types of tasks, the type itself can be included in the similarity computation. One study uses Bayesian

belief propagation to determine the most beneficial action for each agent in the system [25]. For this thesis, the cluster which minimizes the euclidean distance from the agent to the cluster centroid is considered most beneficial and is selected by the agent. Each agent communicates with a centralized agent, who will give advice on the basis of how beneficial it would be for the system to allocate a particular task to that agent. The central agent rectifies any overlaps between two or more agents. As there is no direct communication between agents, this method has less network usage than auction techniques. This makes the system scalable to larger numbers of agents. In the past, K-means has been used to form clusters of sub-tasks, coupling it with auction techniques [7]. The problem with K-means is that it is sensitive to outliers and it implicitly assumes that each cluster has an equal number of observations. Since the cluster size is pre-defined for K-means, it can result in over-fitting. We have used agglomerative hierarchical clustering as it results in more appropriate clustering for different data sets, thereby reducing the total travel cost. The following metrics are used to compare the performance of the overall system:

1. Total travel cost is the total path distance traveled by all the robots.

2. Maximum distance traveled per robot is the maximum distance traveled by a robot in one round of task assignment. This signifies the time taken to execute a round of assignment.

3. Balancing cost index is the measure of balancing the workload over all robots in the framework. The range of values for this index is (0,1] and

the more the value of this index is closer to 1, more balanced is the workload in the system (more detail in results chapter).

Furthermore, we have considered the impact of obstacles in an environment by using the floor plan maps. The rest of this report is organized as follows. After the introduction, Chapter 2 overviews auction based techniques, learning based methods, and inference and clustering techniques. Chapter 3 introduces the methodology and the framework of the system. Chapter 4 presents the experimental results, and Chapter 5 contains concluding remarks.

Chapter 2

Background

Numerous problems in the robotics domain require multiple robots to perform complex tasks for many practical problems, such as batch manufacturing systems, foraging, surveillance, and exploration. Due to the nature of these complex tasks, several methods focusing on the Multi-Robot Task Allocation, MRTA have been proposed. Based on the type of techniques, the methods can be further subdivided as: 1) auction based techniques, 2) learning based methods and, 3) inference and clustering techniques.

2.1 Auction based techniques

Several of the proposed methods in the past for distributed task allocation use variants of a market-based approach using auctions, as described in Dias's survey paper [6]. The motivation behind this method is that robots should be allocated tasks such that each task is performed using minimum resources. It uses the concept of divide and conquer, with the goal of a team being able to perform a large task by dividing it into smaller sub-tasks. The large task is decomposed and the sub-tasks are performed by the sub-teams of agents. Each of the sub-teams has limited resources in which these small

sub-tasks can be performed with a cost or an objective function, which is maximized e.g. profit or reward for a particular action or minimized e.g. time taken to move a box from one location to the other. In the process of an auction, an auctioneer initiates the auction for the assignment of every small task. It accepts bids from other members which are determined by calculating the cost function or the objective function for each robot or sub-team. The auctioneer then chooses to allocate the task to a team or robot by maximizing or minimizing this function. Auctions are very common in market-based approaches. Each of the teams or robots submits a bid and the winner is the one with the highest bid. Bids are submitted to the auctioneer who itself can be a sub-team or a separate agent dedicated to taking the role of an auctioneer. The following steps explain the different stages in the auctioning process [9]:

- A message for the introduction of the task is sent by an agent referred to as the auctioneer into the network. This is a new task that is to be performed.
- Each robot who has the resources receives this message and evaluates its own fitness and returns the bid in a message.
- After some time, the auctioneer determines the fittest through the received messages and notifies the winner and the losers.

The classical algorithm, MURDOCH was shown to be an approximation to the global optimum of resource usage and that this piece of work is

a contribution to the area of intentional co-operation based multi-agent distributed systems [9]. In MURDOCH, after some time each auctioneer determines whether the initial winner is able to perform the task or is it stuck at some point because of a failure. In the case of failure, it reassigns the task to someone else based on fitness. Such a monitoring mechanism provides a way to execute the task even in the case of robot failures. Before MURDOCH, RETSINA [28] and open agent architecture [19] were also proposed. Unlike MURDOCH, these approaches are centralized with one broker [9].

To summarize, an auctioneer initiates the auction for the assignment of every task. It accepts bids from agents that are computed by calculating the cost function or the objective function for each robot or sub-team. The auctioneer then chooses to allocate the task to a team or robot by maximizing or minimizing this function.

The most common type of auction is a single-item auction in which only one task is offered at a time. Other type of auctions are combinatorial or multi-item auctions in which a bundle of tasks is offered in every round. As compared to single item auctions, combinatorial auctions allow many tasks to be allocated in a single round. As combinatorial auctions make bid evaluation, communication, and auction clearing computationally complex, they are not used frequently [9].

2.2 Learning based methods

Learning based approaches for task allocation have also been proposed [26]. This research work describes the problem as sequential decision making in a partially observable environment, and as such can use Markov Decision Processes and reinforcement learning to perform task allocation in a team of robots. It combines short-term planning and a value function together to generate a dynamic scheduler for multi-robot task allocation. It terms it as partially observable as the tasks will be entering the system randomly and thus, while planning for a set of tasks one cannot be aware of what set of tasks will be entering the system later on. As the agents in the system are not aware of the complete system, there is a notion of belief space, which is the probability distribution representing the current state of the system given the observation history. Since the state space can be high dimensional and technically belief space is infinite, there is a need for approximate methods. The work also discusses the concept of short term and long term planning. Short term planning assumes that the short-term future is predictable and it is then treated as static planning. However in the long term, everything is unpredictable and requires a method that caters tasks entering the system randomly. It mentions about the proposed hybrid planning and learning system in which in the short term planning, it ignores new tasks entering the system up to a particular threshold of time.

A concurrent learning approach has also been studied [21]. In this technique, each agent is responsible for its own learning. This is very powerful

if the problem can be decomposed into sub-problems independently. Based on how accurately the agents have learned, they get a reward, split equally amongst all the learners. This work mentions two issues of using machine learning techniques on multiple agents to solve problems. Firstly, multiple agents cause the search space to be large because of the many ways in which these agents can interact with one another. Even the smallest change to the environment of one agent can impact the system as a whole. Secondly, multi-agent learning can involve a single learner or multiple learners. Since these agents are not completely independent of one another they actually have to learn things in context to one another.

2.3 Inference and clustering techniques

In auction-based methods, messages are always exchanged explicitly. Although this is rarely a problem and significant methods have been introduced that cater to the communication failure, implicit coordination can also be used. For example, in the work of Stulp et al. [27] a computational model is developed for coordination between agents, models of teammate behavior are then computed and used for implicit coordination on real robots. Some algorithms use Bayesian probabilistic inference to determine the belief of a robot in a task allocation problem [11, 16, 25]. The method proposed by Schwertfeger uses the chain of sight problem domain to demonstrate this approach [25]. In the chain of sight problem, robots form a chain between a particular start and goal location. Instead of using a centralized or a distributed

auction/market-based methodology, this approach allows robots to infer task assignments locally through Bayesian belief propagation. The algorithm converts belief to task assignments where each robot decides its own task based on the belief of others and itself. Each robot computes a probability distribution using the Bayesian belief propagation algorithm over all the possible sub-tasks in the system.

In the past, the idea of clustering similar tasks in order to balance the distribution of load in a multi-agent system has been proposed [10, 12]. In Elango's work, K-means has been used to form clusters of sub-tasks, coupling it with auction techniques [7]. However in this approach, a combinatorial scheme of bidding has been used. In cases of a large number of incoming tasks and many robots, combinatorial auctions are known to make bid evaluation, communication, and auction clearing intractable [9]. Also, the presence of obstacles has not been considered in experiments.

Clustering has also been used to solve the problem of coverage with multiple robots [8]. This approach has considered the impact of obstacles in their experiments but have only used K-means for clustering. For the performance evaluation of these methods, only a small number of tasks, robots, and cluster combinations are used even when working in simulation [7, 8, 10].

After reviewing the research performed in the past, it is evident that there have been tremendous advances in solving the MRTA problem. In this area, although mostly research aims in developing systems that will be multi-purpose, many of the papers propose algorithms to solve problems in a specific

problem domain. Also, in the majority of the papers, very little progress is made in performing complex heterogeneous tasks in a tightly coupled scenario and balancing tasks across all robots in a system. While some of the works, particularly the auction based techniques consider environments with obstacles, others like clustering work with an ideal environment of an empty world.The problem with coupling auctions with clustering techniques is that a combinatorial scheme of bidding has been used. In cases of a large number of incoming tasks and many robots, combinatorial auctions make bid evaluation, communication, and auction clearing intractable. In many of the methods, only a small number of tasks, robots, and clusters are used even when working in simulation.

Chapter 3

Methodology

The proposed algorithm consists of three major components. Fig. 3.1 shows a high-level architecture of the methodology. First, every task which has entered the system is clustered on the basis of their similarity. Tasks can be known beforehand (e.g. for this thesis, the VRP traveling salesman dataset is used [2]) or can enter the system periodically or irregularly (e.g. if large batches of tasks enter from time to time). Second, the mapping of each cluster to an agent is determined via self-assignment. Each agent sends a message to the centralized agent containing the selected cluster's ID and the associated cost. The centralized agent detects overlaps in individual assignments, rectifies them, and determines the final assignment. The agents then execute their assigned tasks. The following sections explain each component in detail.

3.1 Clustering

Clustering can be broadly defined as dividing a set of objects in such a way that objects in the same cluster are more similar to each other as compared to those in other clusters. The similarity metric is defined according to the nature of the problem being addressed. For spatially-related tasks, the

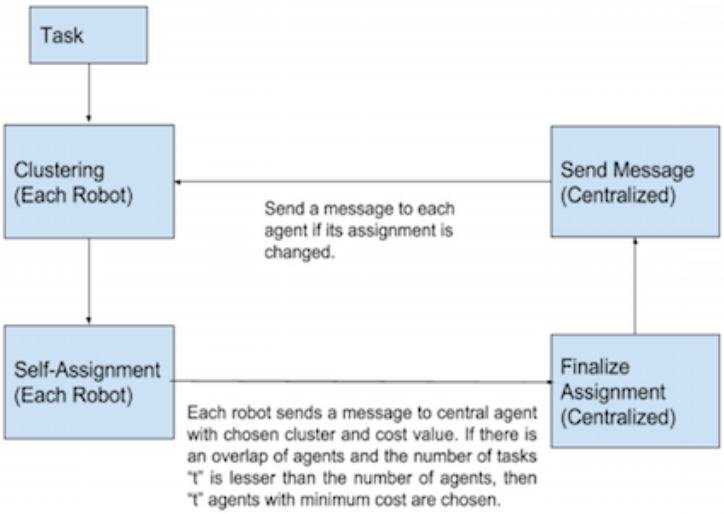

Figure 3.1: High Level Architecture.

Euclidean distance or L_2 norm is used frequently as a similarity measure. The cosine similarity, which measures similarity between two vectors based on the cosine of the angle between them, can also be used to determine points which are similar with respect to orientation in the geometrical space. In order to balance workload in the system, we will use the clustering technique with the L_2 norm to group the n tasks into k clusters where each cluster of task/tasks will be assigned to an agent. We use K-means and agglomerative hierarchical clustering techniques and compare the performance of the overall task allocation system. Eq. 3.1 describes the similarity relationship.

$$\mid d \mid = (x1, x2) \tag{3.1}$$

where *x1* and *x2* are two tasks and d is the result of the measure of similarity between these two tasks. In specific scenarios, *x1* and *x2* may represent location of boxes (box pushing problem) or the areas to be explored (exploration problem).

K-means is a widely used method for clustering. In the past, K-means has been used to form clusters of tasks, coupling it with auction techniques [7]. However, it makes some assumptions that can make the balancing of tasks inefficient. In particular, it implicitly assumes that the variance of the distribution of each attribute is spherical and all attributes have the same variance and that the prior probability for all k clusters are the same, i.e. each cluster has roughly an equal number of observations. In addition, the number of clusters must be given ahead of time.

The method introduced in this thesis, proposes using agglomerative hierarchical clustering to divide the n sub-tasks into k clusters. It is possible that a task (e.g. consider a searching task where the similarity metric is the Euclidean distance between the agent and the goal location) will be similar to other tasks but might be misclassified and assigned to a different cluster because the distribution in the geometrical space might not be spherical. K-means technique can be very sensitive to outliers. Due to this, when this cluster will be assigned to a agent it is possible that these tasks might not be similar e.g. considering the euclidean distance metric, it might be a task which requires the team to go to a farther location than a closer one. This will result in a higher total travel cost. Since agglomerative hierarchical clustering

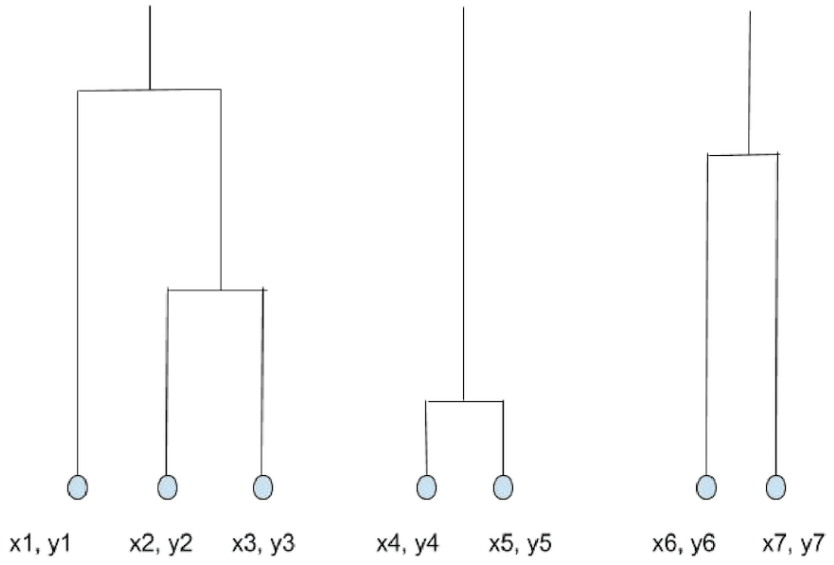

Figure 3.2: Agglomerative Hierarchical Clustering.

does not have such limitations when compared with K-means, it will result in more accurate clustering, thereby reducing the total travel cost.

An example of using hierarchical agglomerative clustering is shown in Fig. 3.2. Consider a box pushing task where the user asks to push boxes located at multiple locations. Each point is a target location in two-dimensional space specified as (x,y) coordinates. The similarity measure can be visualized as a dendrogram shown in Fig. 3.2 where each merge is represented by a horizontal line. The height of the horizontal line denotes the similarity of the two locations that were merged, with the higher value being less similar. Moving from bottom to top, each location is treated as a single cluster initially. The dendrogram shows all the merges resulting in the location clustering.

Since agglomerative clustering does not use a fixed number of clusters, we need to specify the point for ending the clustering process. We use a threshold mechanism based on the average of pair-wise distances within a cluster. Initially each point is in its own cluster and we iteratively merge the two closest clusters until all the points are merged into a single cluster. From this hierarchical structure, we form flat clusters such that the original data points in each flat cluster have no greater pairwise distance than d. Eq. 3.2 explains about the calculation of the threshold.

Given the pair-wise distances between points, we can select the measure for intercluster similarity, also known as the linkage criterion. Since the tasks in the benchmark datasets and synthetic datasets are well distributed in space, we use the euclidean metric for the distances and mean linkage as the intercluster similarity measure. Mean-linkage is a trade-off between complete-linkage, which is sensitive to outliers and single-linkage clustering, which can potentially form clusters which do not satisfy the natural notion of clusters forming spherical distributions. In each iteration, the pair of clusters with the highest cohesion are merged together. However, it is important to note that the selection of a similarity metric and linkage criterion will influence the shape of the clusters, and must be selected according to the nature of the problem. When comparing K-means to agglomerative clustering, since the number of clusters for K-means must be pre-defined, it is set to be equal to the number of agents in the experiments. In order to be fair in comparison to agglomerative clustering, for the experiments with fixed tasks, K-means with

threshold is implemented. In this case, the value of k is determined by using the threshold from Eq. 3.2. The initial value of k is set to 1 and is increased iteratively until it converges to a value, such that in each cluster the distance of every observation to the cluster centroid is within the value of the threshold.

When new tasks enter the system while tasks are already being executed, two different approaches; greedy and reclustering can be used. In the greedy approach, the new task(s) are added to whichever cluster is closest based on the cluster mean, while in re-clustering, new clusters are created from all current and new tasks. For this thesis, both techniques are implemented and the results are compared with each other. Python's Scikit learn clustering functions are used for K-means and agglomerative clustering [1].

There are two important things to consider with respect to the implementation:

1. The number of clusters is determined using a threshold mechanism for agglomerative technique by computing the average of pair-wise distances for n tasks.

$$d_{average} = \frac{\sum_{i=1}^{p} p_i}{p} \tag{3.2}$$

   where d is the threshold, and p is the number of pairs. To be fair in comparison to agglomerative clustering, the number of clusters for K-means is set to be equal to the number of agents in the experiments.

2. The number of clusters can plausibly change if new tasks are entering

the system. We have used greedy and re-clustering techniques in the experiments and have compared the results.

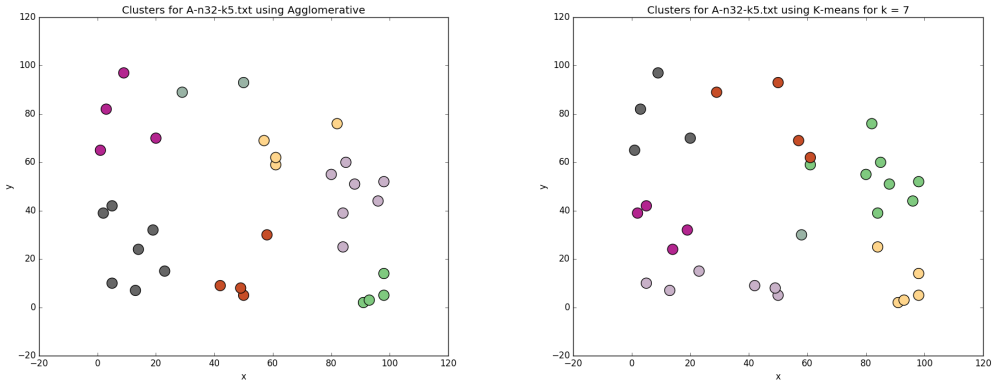

Figure 3.3: Clusters for A-n32-k5 dataset: (left) Clusters using agglomerative hierarchical clustering (right) Clusters using K-means with k=7.

Fig. 3.3 shows the clusters generated from agglomerative hierarchical and K-means clustering for one of the benchmark datasets. On the left side of Fig. 3.3, we can see that the agglomerative clustering produces 7 clusters. On the right side is one instance of K-means clustering — here we have specified 7 clusters as a comparison to the agglomerative approach, but any number can be computed. Also, since K-means involves an initial random seed of cluster means, these clusters will not always be repeated for this data set. Not only must the number of clusters be decided in advance, K-means can result in classifying data points in counter-intuitive ways. In this particular case, we can see that the points (61,59) and (61, 62), shown in green and red respectively, have been separated by K-means in two different clusters. Similarly, the point (58,30) has been put in a cluster by itself. The proposed

method uses agglomerative hierarchical clustering to divide the n sub-tasks. Agglomerative clustering is a bottom-up approach in which each data point starts as an individual cluster and pairs of clusters are merged on the basis of a similarity metric and linkage criterion. Since K-means uses the cluster mean to decide how to divide the tasks, it can be very sensitive to outliers, and may put tasks in a cluster where they are not particularly similar (i.e. nearby) to the others, whereas agglomerative clustering should not suffer from the same issues. However, since agglomerative clustering does not use a fixed number of clusters, we need to specify at what point the clustering process stops.

3.2 Self-Assignment

Once the clusters have been determined, each agent should assign tasks to itself. Here we use a greedy approach where each agent will determine which cluster is most beneficial for it by minimizing the cost of executing that task. The cost function depends on the nature of the problem e.g. for a box pushing problem it can be the robot travel cost i.e. the distance between the robot and the goal location.

For this thesis, once the clusters are made, each robot chooses a cluster based on the minimal euclidean distance to the cluster center. The algorithm consists of the following steps:

1. The robot greedily finds the closest cluster, the one that has the minimum travel cost. It assigns itself to that cluster and sends a message

containing the cluster ID and travel cost to a centralized agent.

2. The centralized agent checks for overlap of robots and clusters. If there is an overlap of agents, it updates the assignment of each overlapping agent by finding the next closest cluster and the associated travel cost. Also, if the number of clusters is less than the number of agents, then agents with minimum cost are chosen.

3. The centralized agent sends an acknowledgment message to the robots with the final assignment. These agents can start executing the assigned tasks.

4. The agents receive the final assignment in the acknowledgment message and start executing the tasks based on the assignment. The order of executing tasks is computed. After finishing the tasks in the calculated order, each agent sends an acknowledgment message to the centralized agent.

In this work, both techniques are tested with same initial robot locations and tasks. In testing, as the algorithm exchanged messages and agent locations are updated, the overlaps were seen to be reduced. SimPy, a process-based discrete-event simulation framework, is used to exchange messages between agents. Its event dispatcher, based on Python's generator functions, provides for asynchronous networking between agents.

3.3 Task execution

After each assignment, it is required to determine the following for performing task execution by each robot in the system:

1. Order of execution: The order of execution is the order in which the assigned tasks are completed by each robot.

2. Individual path traversal: The individual path traversal is the path taken by each robot to execute the tasks assigned.

We can observe that each round of assignments is equivalent to a Traveling Salesman Problem (TSP). Every robot's current position is the starting location and the assigned tasks are analogous to the locations to be visited. The order of execution is determined by solving the TSP for each robot's starting position and the assigned tasks. The dynamic programming approach has been used for implementing this solution, which is known to be faster than the brute force solution to this NP-complete problem [3]. This solution has a worst case $\mathcal{O}(n^2.2^n)$ running time. Since the value of n is a subset of the number of actual tasks in the system, for the analysis of this thesis the value of n is always small. The path traversal for each ordered set of tasks (ordered by TSP) is then calculated through the A* algorithm, which is widely known to produce an efficiently traversable path between multiple points, known as nodes in the geometrical space [14]. Eq. 3.3 is the heuristic function that has been used for pathfinding between two subsequent nodes.

$$h = |x1 - x2| + |y1 - y2| \tag{3.3}$$

where the coordinates for individual task locations (nodes) are (*x1,y1*) and (*x2,y2*).

Fig. 3.4 shows a floor plan and some traversed paths for the A-n-32 dataset for 3 agents and 32 tasks from one of our experiments. Each pixel in the image is considered as a node and space is discretized to contain 8 possible actions in order to move to any other node in space where a 1-pixel move is equivalent to 0.317 meters in length.

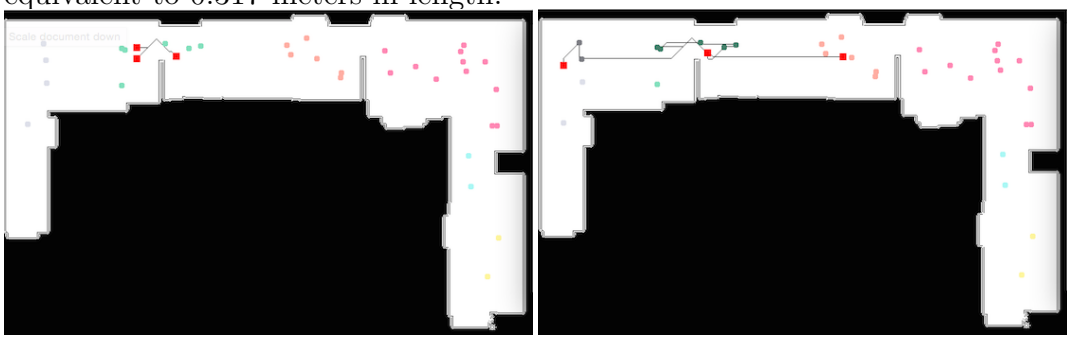

Figure 3.4: Path traversal with A* for 32 tasks and 3 agents: (left) Initial cluster configuration (right) Path traversal.

Chapter 4

Results and Discussion

To evaluate the performance of the proposed method, both clustering techniques are implemented in a simple simulation. Different environments and task distributions were studied. First, the experiments are performed with the benchmark VRP dataset, which contains examples with 32, 45, 55, 69, and 80 tasks in an environment without obstacles. For an environment with obstacles we used the floor plan in Fig. 3.4 with tasks from a uniform random distribution. The metrics are calculated from an average of ten random sets to remove any bias due to the type of distribution. In each case, both clustering techniques are tested and the number of agents are varied. To evaluate the resulting task assignments, total travel cost, maximum distance traveled per robot and balancing cost index are calculated. The total travel cost is the total path distance traveled by all the robots. The balancing cost index is the measure of balancing the load over all robots in the framework, calculated as:

$$b = \frac{\sum_{i=1}^{r} c_i / m}{r} \tag{4.1}$$

where b is the balancing cost index, c_i is the travel cost for robot i, m is the maximum travel cost across all robots, and r is the number of robots. The range of values for b is (0,1] and the closer the value of this index is to 1,

the more balanced is the load in the system. A value of 1 signifies that all the robots in the system have the same travel cost corresponding to the tasks assigned to them.

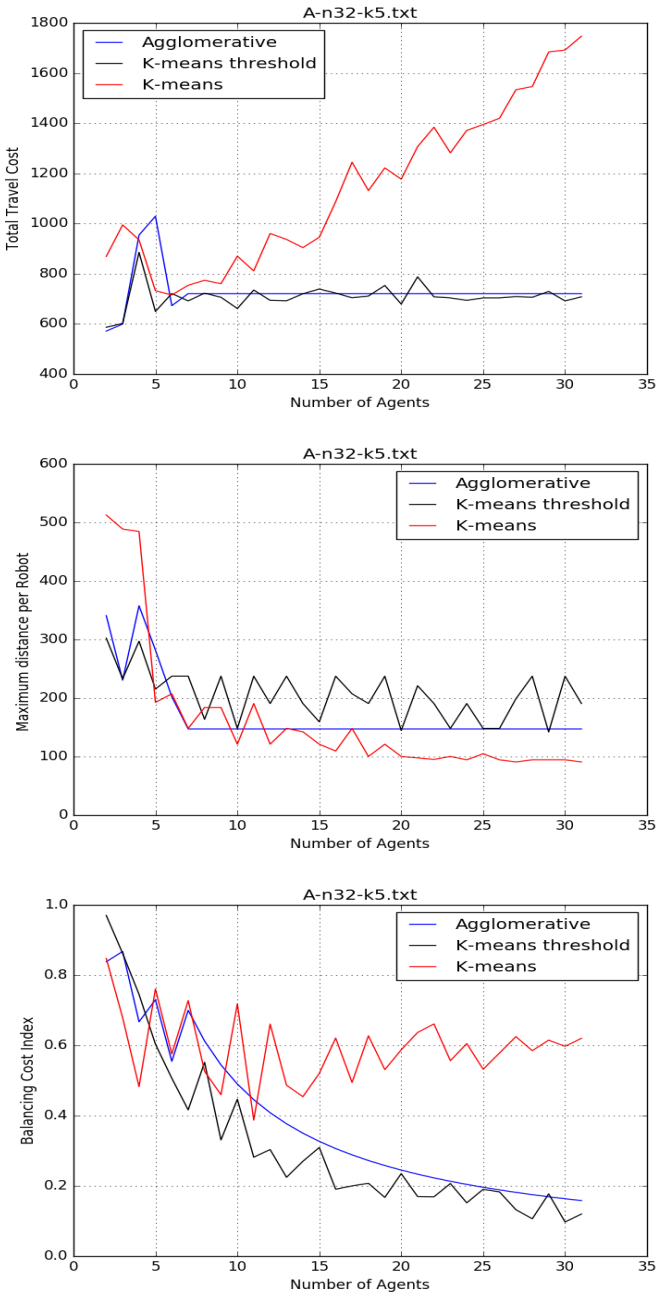

Figure 4.1: Results for VRP data set A-n32-k5 as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

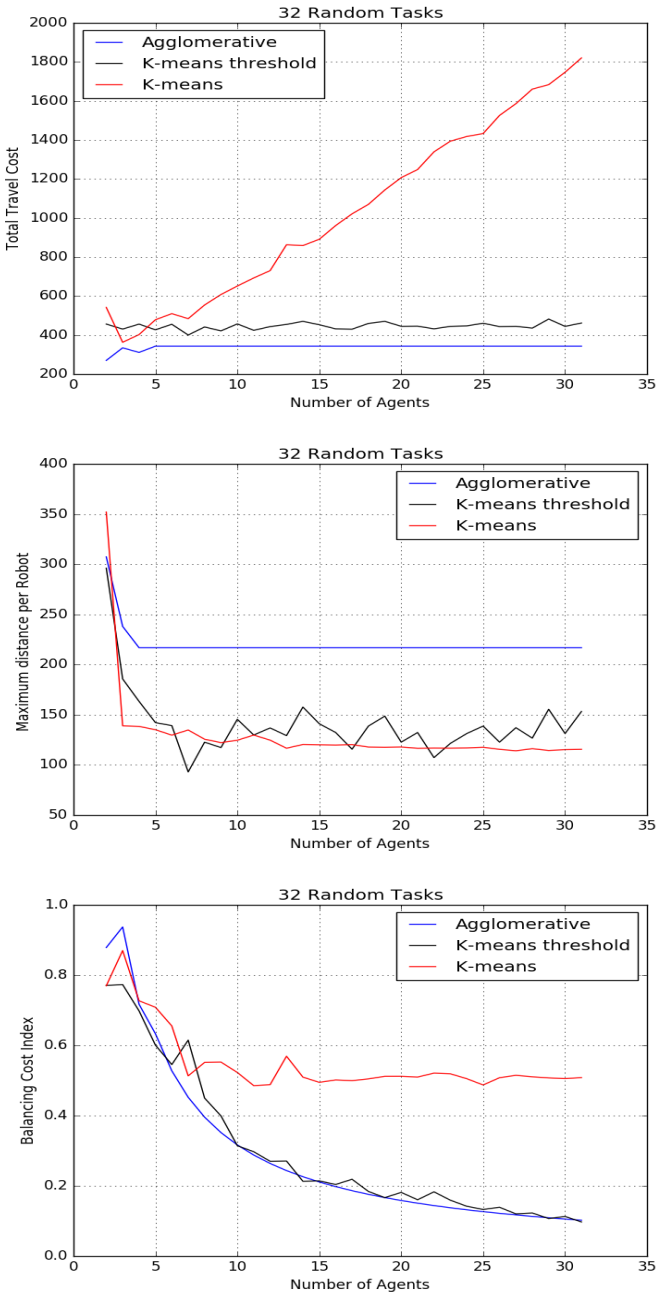

Figure 4.2: Results for 32 randomly generated tasks as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

The results for one representative test of the VRP tasks is shown in Fig. 4.1. For the first case of K-means, the number of clusters is set to the number of agents, so as the number of agents increases, at least one task will be assigned to each robot. As expected, the total cost of the system then increases since all robots must travel some distance to a task, even when two robots are going to very nearby tasks. To be fair, for the second case of K-means (K-means threshold), the threshold (which is calculated using the average pairwise distance) is used to determine the value of the number of clusters. The value of k is increased (starting from 1) iteratively until the resulting clusters are within the same diameter as the threshold. The value of k after the convergence is used.

As we increase the number of agents, the total travel cost for the system under agglomerative clustering becomes constant while the travel cost for K-means increases. This is the point where the number of agents is equal to the number of clusters created using the agglomerative clustering threshold. Since each cluster is assigned to an agent, no matter how many agents are available, excess robots are not utilized and the total cost does not change. One can observe from the plots for the maximum distance traveled per robot and the balancing cost index, that the higher value for a balancing cost index corresponds to a lower value for maximum distance traveled per robot (essentially conveying the same information). We can see that the maximum distance traveled for any robot (and thus the overall completion time for the set) is slightly lower for K-means, due to the increased utilization of the robots.

However, when comparing K-means threshold and agglomerative, a very slight difference is seen between the two methods when comparing the three evaluation metrics. A major benefit of using agglomerative clustering is the stability due to it's deterministic behavior when using the threshold on the hierarchical tree structure. K-means threshold, due to its stochastic nature results in different values of travel cost even when the same value of k is used. For other tests in the VRP dataset (shown in the Appendix), different numbers of clusters were created by agglomerative clustering but the same patterns held in all cases. We also found generally similar results for random tasks in an environment with obstacles as shown in Fig. 4.2, though in this case the agglomerative clustering was generally more efficient, presumably due to the more uniform distribution of tasks being suited to this type of clustering compared to the VRP tasks. In this case agglomerative selects a lower value for the number of clusters than K-means threshold, thereby reducing the total travel cost.

On the other hand, if we are instead concerned with balancing the load across the agents, the balancing cost index is seen to decrease more sharply towards 0 (i.e. less load balance across the robots) for agglomerative clustering as the number of agents increase. This reflects that the number of robots used is capped by the number of clusters found by this technique, so that each robot travels longer distances (more time) as opposed to K-means which uses all robots so each one may travel shorter distances. The main advantage of the agglomerative technique is its ability to determine an appropriate cluster

size for individual datasets based on the thresholds.

In the remaining experiments, we investigated the performance of the system when new tasks are added dynamically over time. We have tested both the greedy and reclustering techniques for incorporating new tasks over the same sets of tasks. In the first of these experiments, we regularly introduce tasks by giving one new task as each task is completed, for a total number of n times where n is the initial number of tasks. Results of one representative experiment are shown in Fig. 4.3.

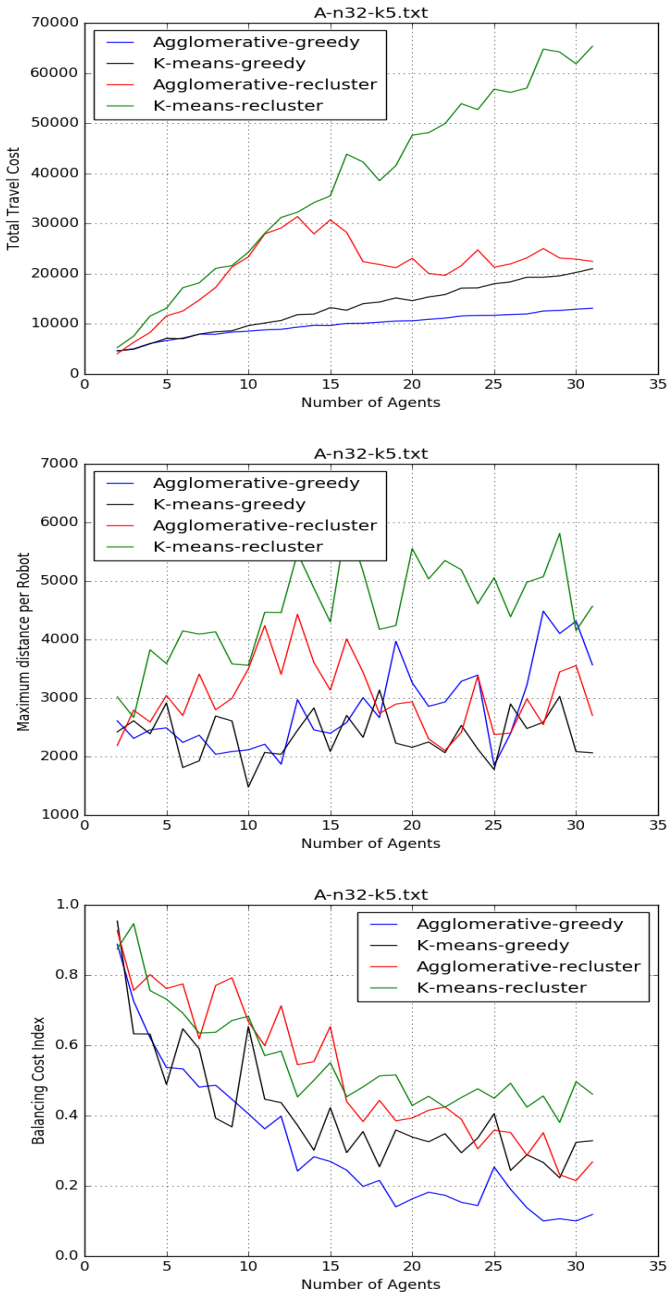

Figure 4.3: Results for VRP data set A-n32-k5 with tasks entering frequently, as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

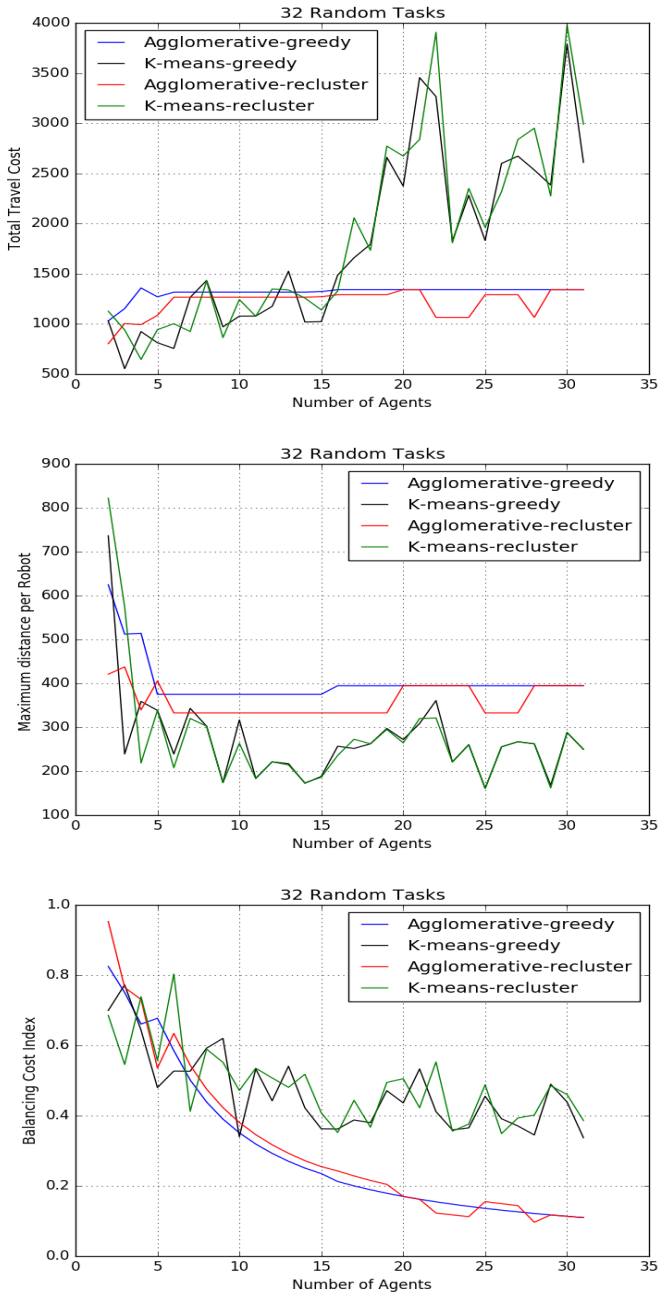

Figure 4.4: Results for 32 randomly generated tasks with tasks entering frequently, as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

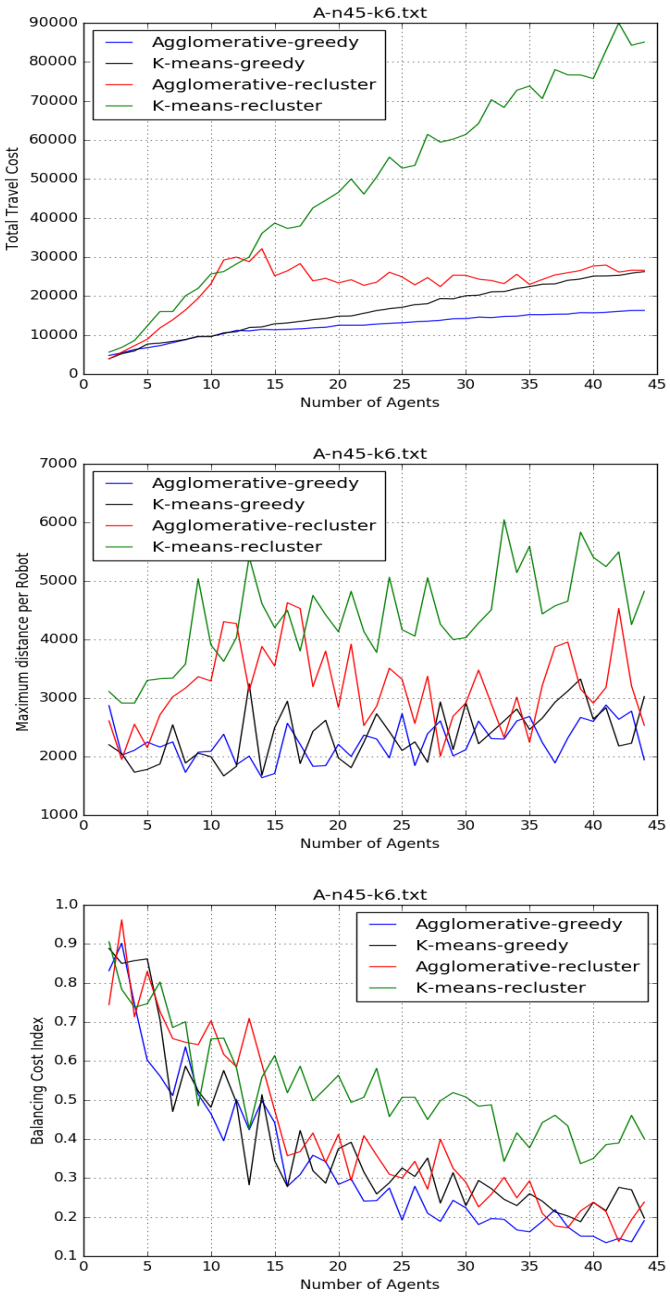

Figure 4.5: Results for VRP data set A-n45-k6 with tasks entering frequently, as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

Similar to the results for fixed tasks, for frequently entering dynamic tasks, agglomerative hierarchical clustering has a lower total travel cost. The results also show that when tasks are added over time, the greedy clustering method works better than global reclustering for both the K-means and agglomerative approaches. The impression from these experiments is that the greedy approach works better because it takes into account not only the original distribution of tasks in space but also retains the current robot assignments. However, in the presence of obstacles, as shown in Fig. 4.6, there is little difference in the performance of both clustering methods.

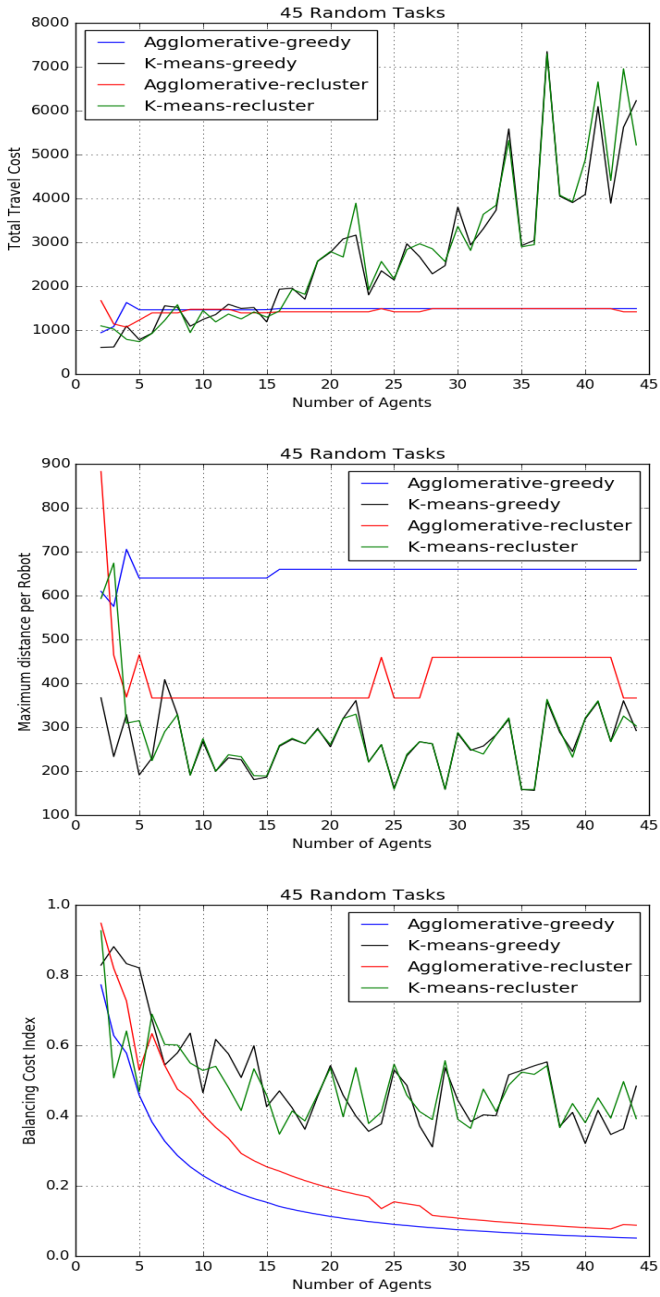

Figure 4.6: Results for 45 randomly generated tasks with tasks entering frequently, as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

In the final set of experiments, we are interested in the performance of the system when a large number of tasks are added infrequently. To test this case, we set up an experiment in which n tasks are given initially, and once the number of clusters is less than or equal to the number of agents, an additional n tasks are added. Results for random tasks in the environment with obstacles are shown in Fig.4.8.

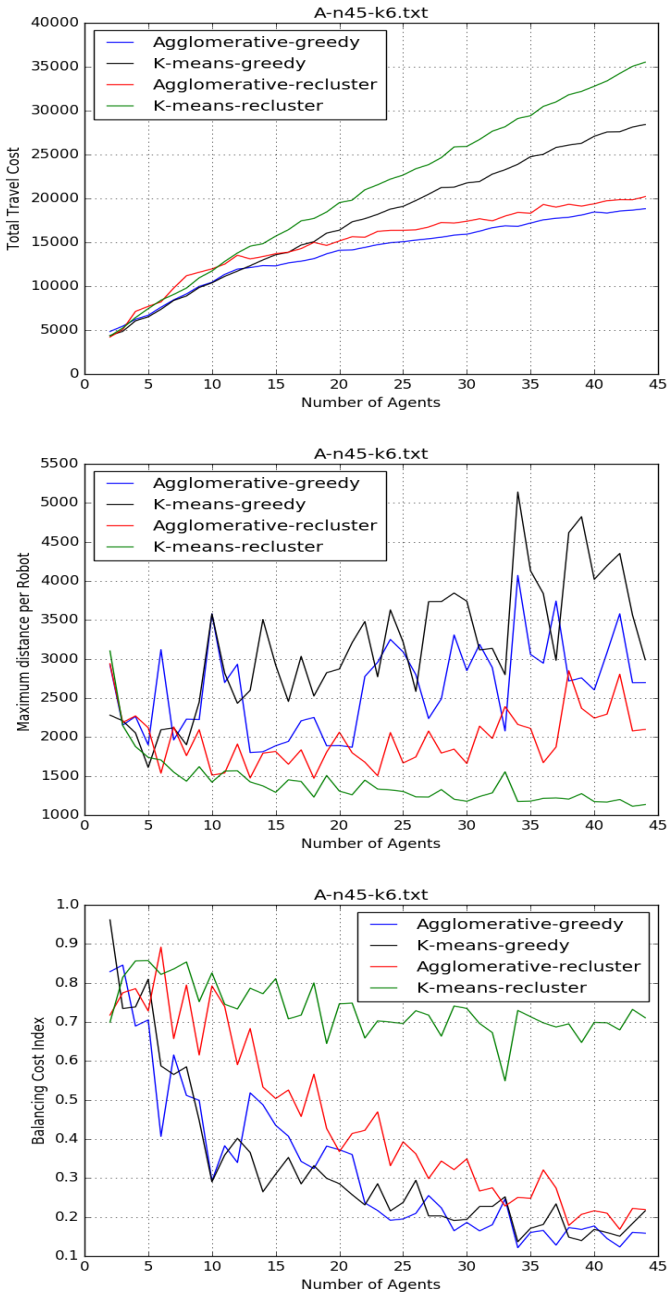

Figure 4.7: Results for VRP data set A-n45-k6 with tasks entering infrequently, as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

Similar to other experiments, agglomerative has a lower cost than K-means, and the greedy approach to adding tasks gives better results than global reclustering. This may seem counterintuitive, in that for a large number of infrequent tasks, reclustering should work better since it will be able to choose an appropriate cluster size for the new distribution. However, reclustering with new tasks can create additional clusters, so instead of robots adding a few nearby tasks, a different idle robot is assigned to these tasks from farther away, resulting in a higher total travel cost.

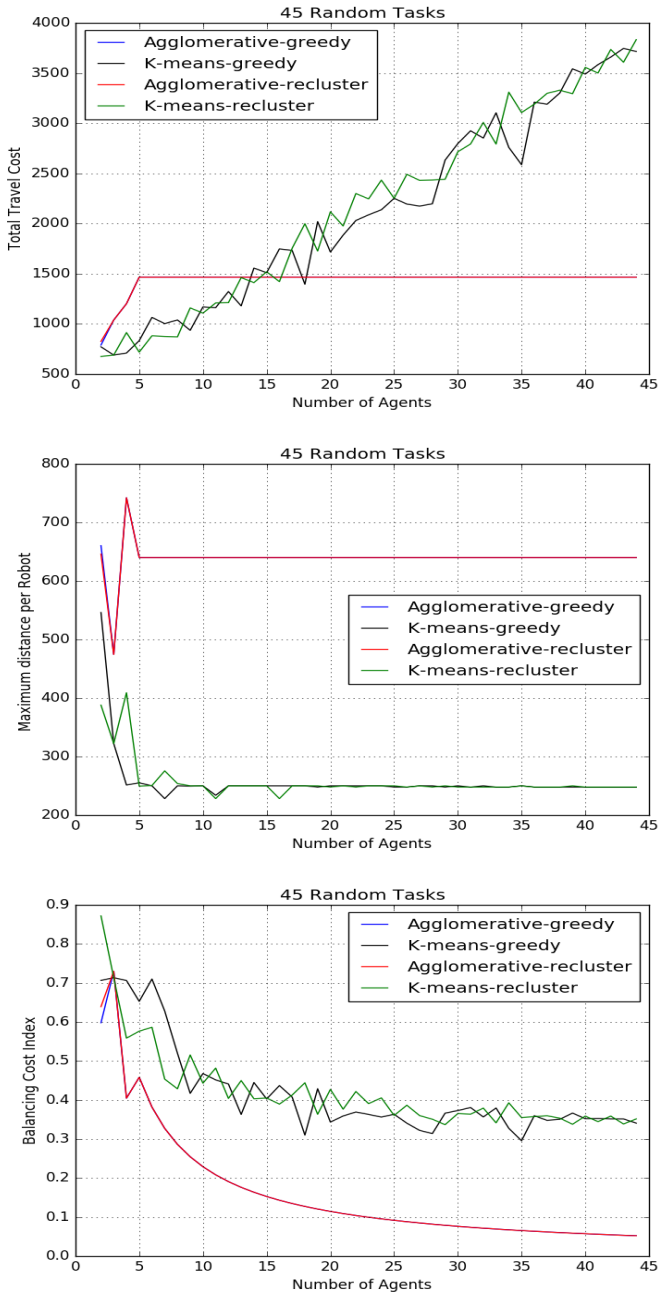

Figure 4.8: Results for 45 randomly generated tasks with tasks entering infrequently, as a function of the number of agents present: (top) Total travel cost (middle) Maximum distance per robot (bottom) Balancing cost index.

We can draw a number of conclusions from the conducted experiments. Firstly, the travel cost by using agglomerative clustering does not vary beyond the point where the number of agents becomes equal to the number of clusters. This is because the number of clusters produced is independent of the number of agents and no matter how many agents we increase, they are not going to be utilized. In K-means, adding more number of agents means utilizing them which will increase the travel cost. However, K-means will result in lower completion time (seen from the maximum distance per robot plots) and will also balance tasks in a better way (as observed from higher values for balancing cost index) as compared to agglomerative clustering.

Secondly, in K-means, when we increase the number of agents (increasing the cluster size), the K-means clustering approach starts to look for patterns that dont actually exist in the underlying data, causing overfitting.

Thirdly, when new tasks are added frequently, one at a time, the greedy approach works better. The greedy approach maintains the current robot to task assignments while assigning new tasks to existing clusters, inherently incorporating the current robot positions with respect to the assigned tasks. Since the added tasks correspond to a uniform random distribution between the minimum and maximum value of existing tasks, it is beneficial to cluster them greedily by assigning the new appearing task to the cluster with the nearest cluster mean.

Lastly, although it makes intuitive sense that when a large number of tasks are added all at once, reclustering should work better, the experiments

depict an opposite behavior. The reason for this pattern is that reclustering does not take into account the current robot positions requiring the robots to travel larger distances, eventually resulting in a higher travel cost.

Chapter 5

Conclusion and Future Work

This thesis proposes a clustering and greedy assignment based method for solving the balanced MRTA problem. In the process of doing so, the agglomerative hierarchical and K-means clustering techniques are compared to obtain the best cluster size and develop a greedy task assignment strategy.

The experimental results demonstrate the effectiveness and efficiency of the proposed method, with agglomerative clustering performing better than K-means in terms of reducing the total travel cost. The agglomerative hierarchical clustering can not only reduce the total travel cost but can also select an appropriate cluster size even in cases of a large number of initial tasks, new dynamic tasks, and presence of obstacles. However, in the case where there are a large number of agents, agglomerative clustering produces assignments with fewer clusters and therefore higher maximum cost per robot (and higher execution time) than K-means (seen from the maximum distance per robot plots). In cases of new tasks, the greedy approach results in lower travel cost by inherently incorporating the current robot positions with respect to the assigned tasks. Also, it is evident from the higher values for balancing cost index that K-means will balance tasks in a better way as compared to agglomerative

clustering.

Some of the conducted experiments (fixed tasks) use K-means threshold method to determine the number of clusters by an iterative search of increasing k until the resulting clusters are consistent with the threshold. Other conducted experiments use the number of clusters for K-means to be the same as the number of agents. In the future, other experiments can be conducted using this method for frequent and infrequent dynamic tasks. Also, for cluster selection in agglomerative, an iterative search of varying the threshold can be used. This will result in a direct comparison between the balancing cost index for agglomerative hierarchical and K-means clustering methods. For example, it is no surprise that the balancing cost index is always higher for K-means considering that we keep more robots busy by creating as many clusters as agents.

In the method proposed in this thesis, the assignment of agents to clusters is a greedy self-assignment process which may not be optimal. In the future, the cluster to agent assignment can be made optimal by using polynomial time combinatorial optimization algorithms. One of such algorithm is the Hungarian method in which the problem is to find the lowest-cost way to assign jobs to resources and is able to find the optimal solution in polynomial time[17].

The proposed method can be extended to account for robot failures and perform clustering considering obstructed paths. The obstructed distances will incorporate real-world dynamic obstacle constraints and improve clustering

quality. Future work will focus on a more distributed approach to cluster assignment as well as providing automated techniques to tune the system behavior toward optimizing total system effort versus overall execution time.

Chapter 6

Appendix

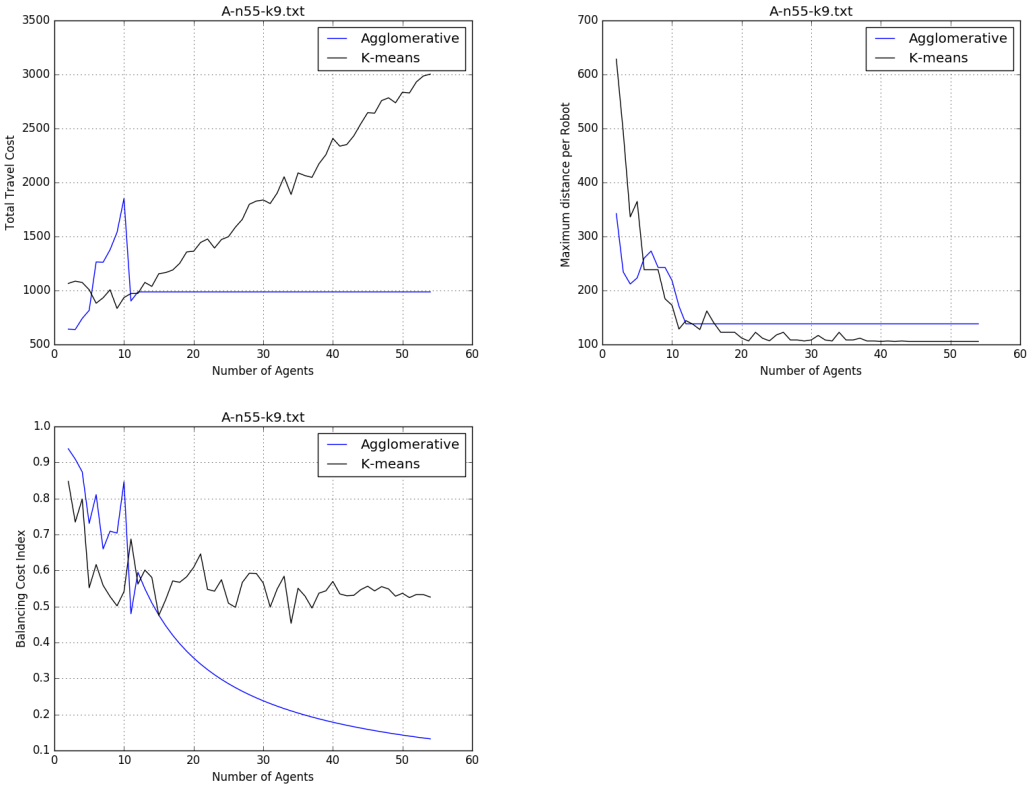

Figure 6.1: Results for VRP data set A-n55-k9 as a function of the number of agents present: (top left) Total travel cost (top right) Maximum distance per robot (bottom left) Balancing cost index.

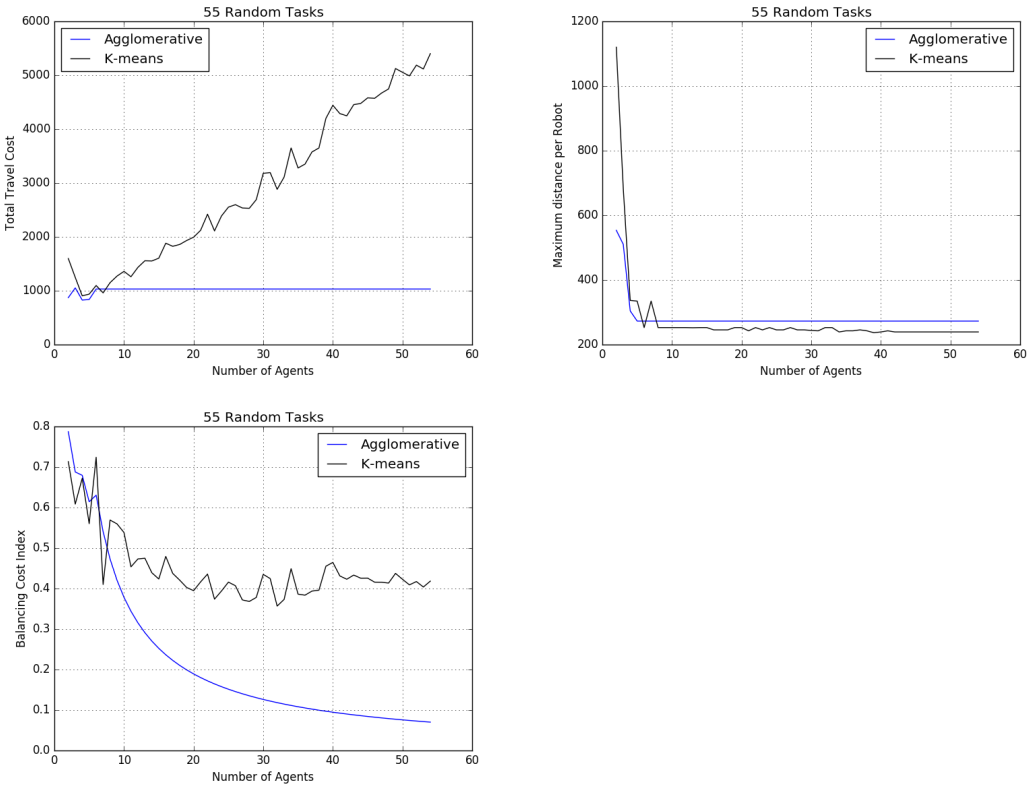

Figure 6.2: Results for 55 randomly generated tasks as a function of the number of agents present: (top left) Total travel cost (top right) Maximum distance per robot (bottom left) Balancing cost index.

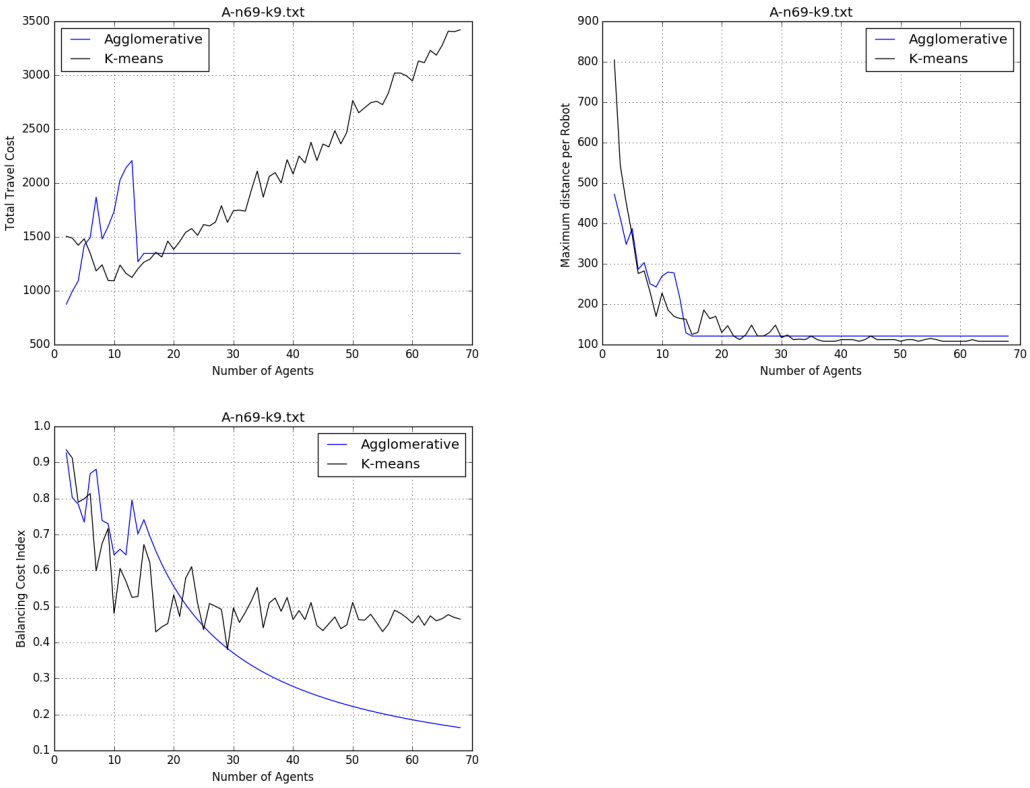

Figure 6.3: Results for VRP data set A-n69-k9 as a function of the number of agents present: (top left) Total travel cost (top right) Maximum distance per robot (bottom left) Balancing cost index.

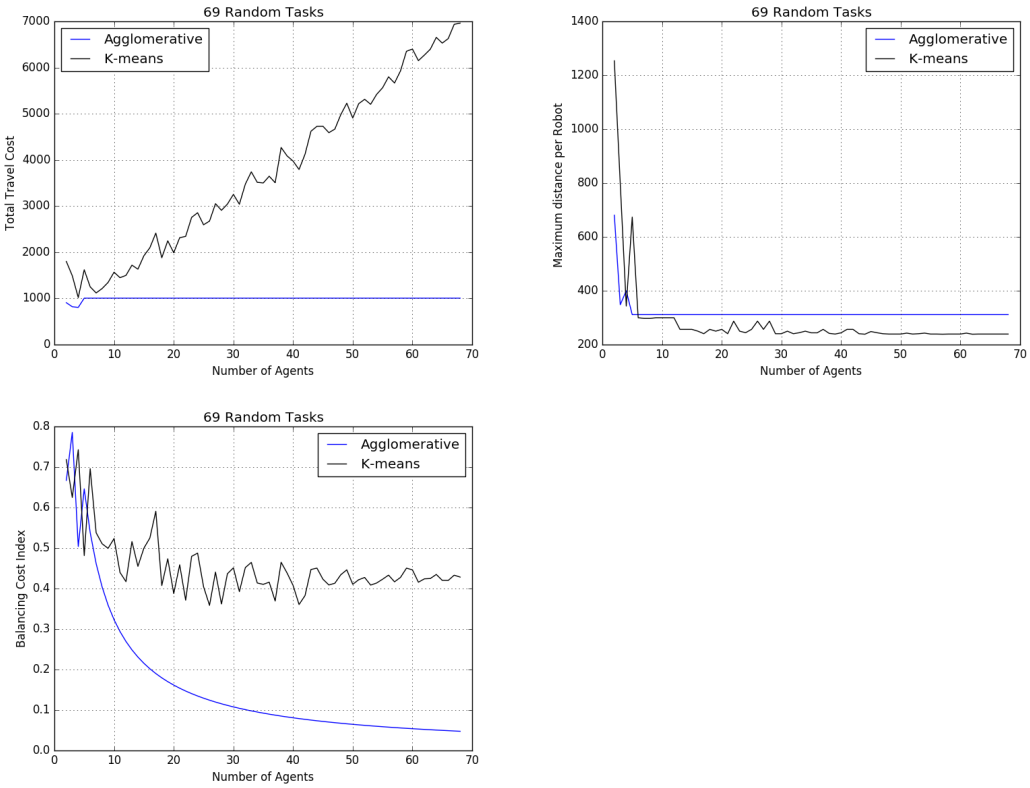

Figure 6.4: Results for 69 randomly generated tasks as a function of the number of agents present: (top left) Total travel cost (top right) Maximum distance per robot (bottom left) Balancing cost index.

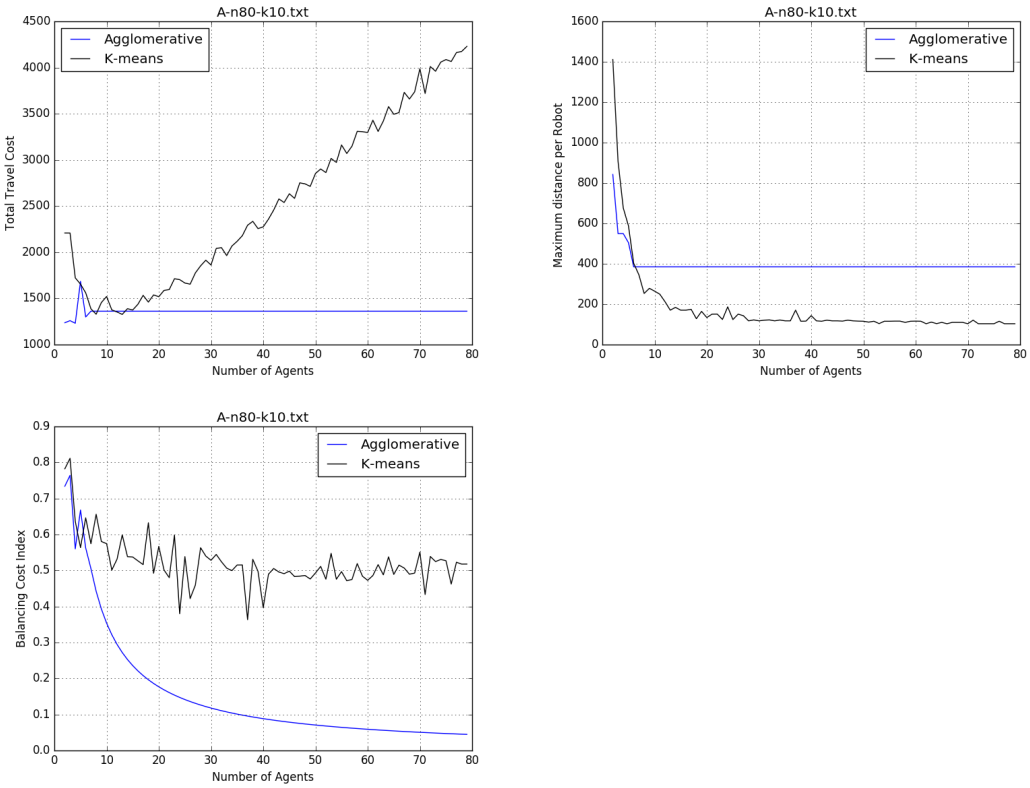

Figure 6.5: Results for VRP data set A-n80-k10 as a function of the number of agents present: (top left) Total travel cost (top right) Maximum distance per robot (bottom left) Balancing cost index.

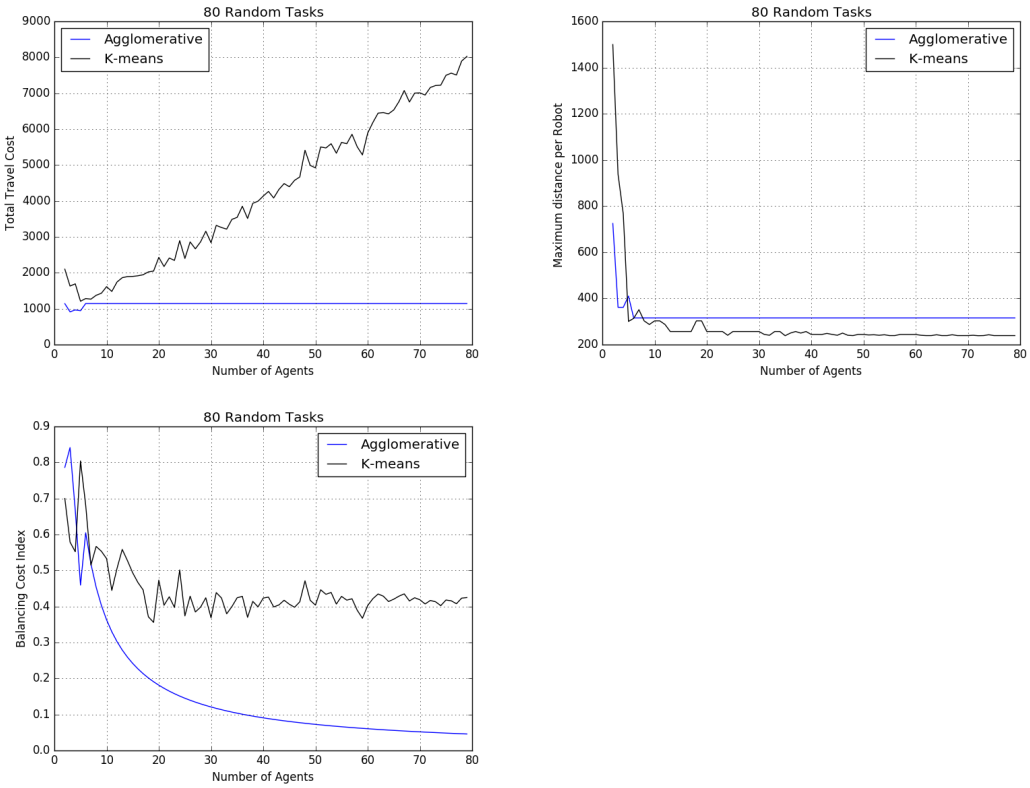

Figure 6.6: Results for 80 randomly generated tasks as a function of the number of agents present: (top left) Total travel cost (top right) Maximum distance per robot (bottom left) Balancing cost index.

Bibliography

[1] scikit-learn 0.17 : Python package index.

[2] Vehicle routing data sets, http://www.vrp-rep.org.

[3] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, January 1962.

[4] H. L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, Aug 2009.

[5] Gautham P. Das, Thomas M. McGinnity, Sonya A. Coleman, and Laxmidhar Behera. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *Journal of Intelligent & Robotic Systems*, 80(1):33–58, 2015.

[6] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, July 2006.

[7] Murugappan Elango, Subramanian Nachiappan, and Manoj Kumar Tiwari. Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms. *Expert Syst. Appl.*, 38(6):6486–6491, June 2011.

[8] A. Gautam, J. K. Murthy, G. Kumar, S. P. A. Ram, B. Jha, and S. Mohan. Cluster, allocate, cover: An efficient approach for multi-robot coverage. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 197–203, Oct 2015.

[9] B. P. Gerkey and M. J. Mataric. Sold!: auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, Oct 2002.

[10] Bradford Heap and Maurice Pagnucco. Analysis of cluster formation techniques for multi-robot task allocation using sequential single-cluster auctions. In *AI 2012: Advances in Artificial Intelligence - 25th Australasian Joint Conference, Sydney, Australia, December 4-7, 2012. Proceedings*, pages 839–850, 2012.

[11] Bradford Heap and Maurice Pagnucco. Minimising undesired task costs in multi-robot task allocation problems with in-schedule dependencies. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 2542–2548. AAAI Press, 2014.

[12] Bradford Gregory John Heap and Maurice Pagnucco. Repeated sequential auctions with dynamic task clusters. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.*, 2012.

[13] Matthew Hoeing, Prithviraj Dasgupta, Plamen Petrov, and Stephen O'Hara. Auction-based multi-robot task allocation in comstar. In *Proceedings of*

the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07, pages 280:1–280:8, New York, NY, USA, 2007. ACM.

[14] Huang Jin, Wu Wei, and Ling Ziyan. Multi-agent pathfinding system implemented on xna. In *Proceedings of the 2012 Fourth International Conference on Computational Intelligence and Communication Networks*, CICN '12, pages 651–655, Washington, DC, USA, 2012. IEEE Computer Society.

[15] E. G. Jones, B. Browning, M. B. Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 570–575, May 2006.

[16] Aaron Khoo and Ian Douglas Horswill. Grounding inference in distributed multi-robot environments. *Robotics and Autonomous Systems*, 43(2-3):121–132, 2003.

[17] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[18] L. Luo, N. Chakraborty, and K. Sycara. Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks. *IEEE Transactions on Robotics*, 31(1):19–30, Feb 2015.

[19] David L. Martin, Adam J. Cheyer, and Douglas B. Moran. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2):91–128, 1999.

[20] K. Nagatani, Y. Okada, N. Tokunaga, K. Yoshida, S. Kiribayashi, K. Ohno, E. Takeuchi, S. Tadokoro, H. Akiyama, I. Noda, T. Yoshida, and E. Koyanagi. Multi-robot exploration for search and rescue missions: A report of map building in robocuprescue 2009. In *2009 IEEE International Workshop on Safety, Security Rescue Robotics (SSRR 2009)*, pages 1–6, Nov 2009.

[21] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, November 2005.

[22] A. Pustowka and E. F. Caicedo. Market-based task allocation in a multi-robot surveillance system. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, pages 185–189, Oct 2012.

[23] Eric Schneider, Ofear Balas, A. Tuna Ozgelen, Elizabeth I. Sklar, and Simon Parsons. An empirical evaluation of auction-based task allocation in multi-robot teams. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 1443–1444, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[24] Eric Schneider, Elizabeth I. Sklar, Simon Parsons, and A. Tuna Özgelen. *Auction-Based Task Allocation for Multi-robot Teams in Dynamic Environments*, pages 246–257. Springer International Publishing, Cham, 2015.

[25] J. N. Schwertfeger and O. C. Jenkins. Multi-robot belief propagation for distributed robot allocation. In *2007 IEEE 6th International Conference on Development and Learning*, pages 193–198, July 2007.

[26] Malcolm J. A. Strens and Neil Windelinckx. Combining planning with reinforcement learning for multi-robot task allocation. In Daniel Kudenko, Dimitar Kazakov, and Eduardo Alonso, editors, *Adaptive Agents and Multi-Agent Systems*, volume 3394 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2005.

[27] F. Stulp, M. Isik, and M. Beetz. Implicit coordination in robotic teams using learned prediction models. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1330–1335, May 2006.

[28] Katia Sycara, Massimo Paolucci, Martin Van Velsen, and Joseph Giampapa. The retsina mas infrastructure. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):29–48, July 2003.

[29] G. Thomas and A. B. Williams. Sequential auctions for heterogeneous task allocation in multiagent routing domains. In *Systems, Man and*

Cybernetics, 2009. SMC 2009. IEEE International Conference on, pages 1995–2000, Oct 2009.

[30] P. Ulam, Y. Endo, A. Wagner, and R. Arkin. Integrated mission specification and task allocation for robot teams - design and implementation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4428–4435, April 2007.

[31] Kai Zhang, Emmanuel G. Collins, Jr., and Dongqing Shi. Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Trans. Auton. Adapt. Syst.*, 7(2):21:1–21:22, July 2012.

[32] Y. Zhang, L. E. Parker, and S. Kambhampati. Coalition coordination for tightly coupled multirobot tasks with sensor constraints. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1090–1097, May 2014.