Sequential Single-Cluster Auctions for Multi-Robot Task Allocation

Bradford Heap

A thesis in fulfilment of the requirements for the degree of

Doctor of Philosophy

UNSW
AUSTRALIA

School of Computer Science and Engineering
Faculty of Engineering

November 2013

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed

Date

Abstract

This thesis studies task allocation in multi-robot teams operating in dynamic environments. The *multi-robot task allocation* problem is a complex NP-Complete optimisation problem with globally optimal solutions often difficult to find. Because of this, the rapid generation of near optimal solutions to the problem that minimise task execution time and/or energy used by robots is highly desired. Our approach seeks to cluster together closely related tasks and then builds on existing distributed market-based auction architectures for distributing these sets of tasks among several autonomous robots.

Dynamic environments introduce many challenges that are not found in closed systems. For instance, it is common for additional tasks to be inserted into a system after an initial solution to the task allocation problem is determined. Additionally, it is highly likely in long-term autonomous systems that individual robots may suffer some form of failure. The ability to alter plans to react to these types of challenges in a dynamic environment is required for the completion of all tasks. In our approach we allow the repeated formation and auctioning of task clusters with varying tasks. This allows us to react to and change the task allocation among robots during execution.

Throughout this thesis we use empirical evaluation to study different approaches for forming clusters of tasks and the application of task clustering to distributed auctions for multi-robot task allocation problems. Our results show that allocating clusters of tasks to robots in solving these types of problems is a fast and effective method and produces near optimal solutions.

Related Publications

Heap, B., and Pagnucco, M.
Sequential Single-Cluster Auctions for Robot Task Allocation.
In Proceedings of the Australasian Joint Conference on Artificial Intelligence (AI 2011). LNCS, vol. 7106, pp. 412-421.

Heap, B., and Pagnucco, M.
Repeated Sequential Auctions with Dynamic Task Clusters.
In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2012).

Heap, B., and Pagnucco, M.
Analysis of Cluster Formation Techniques for Multi-robot Task Allocation using Sequential Single-Cluster Auctions.
In Proceedings of the Australasian Joint Conference on Artificial Intelligence (AI 2012). LNCS, vol. 7691, pp. 839-850.

Heap, B., and Pagnucco, M.
Repeated Sequential Single-Cluster Auctions with Dynamic Tasks for Multi-Robot Task Allocation with Pickup and Delivery.
In Proceedings of the German Conference on Multiagent System Technologies (MATES 2013). LNCS, vol. 8076, pp. 87-100

Heap, B.
Sequential Single-Cluster Auctions - Extended Abstract.
In Proceedings of the German Conference on Multiagent System Technologies (MATES 2013). LNCS, vol. 8076, pp. 408-411

Heap, B., and Pagnucco, M.
Repeated Auctions for Reallocation of Tasks with Pickup and Delivery Upon Robot Failure.
In Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2013). LNCS, vol. 8291, pp. 461-469

Acknowledgements

Although the cover of this thesis states only one author, I consider this work to be the culmination of years of effort that a wide variety of people have contributed to through their knowledge, advice and friendship they have given to me.

At The University of New South Wales, thanks goes first to my supervisor Maurice Pagnucco who has provided advice and direction over the previous three and a half years. Within the Artificial Intelligence Research Group thanks goes to the staff: Bernhard Hengst, David Rajaratnam, Michael Thielscher, Claude Sammut, Malcolm Ryan, Matthew McGill, Adam Milstein, and Abdallah Saffidine. To the fellow research students: Michael Gratton, Timothy Cerexhe, Nawid Jamali, Bhuman Soni, Oleg Sushkov, Adam Haber, Jennifer Buehler, Adrian Ratter, Timothy Wiley, Sean Harris, Theo Wadsley, Jayen Asher and Marc Chee; and to the visitors to the lab: Hannes Straß, Andreas Braun, Benjamin Andres, Christoph Schwering, Roman Feldhoff, Martin Liebenberg and others who have listened to countless presentations and provided feedback and advice throughout the previous few years, thank you.

Thanks also goes to the many individuals who have asked questions and provided feedback after presentations at a number of conferences. In particular, the feedback and advice provided by Wolfgang Renz and Thomas Preisler at the MATES 2013 conference was invaluable to improving some of the data analysis in Chapter Five.

At Massey University thanks go to my honours supervisor Ken Hawick who spurred me into research. Additional thanks goes to those in the Computer Science Department: Chris Scogings, Peter Kay, Ian Bond, Napoleon Reyes, and Martin Johnson. To the wider community of the Institute of Information and Mathematical Sciences and Massey University, thanks goes to the many teachers, general staff and fellow students who have inspired me and pushed me to always ask why. I am especially thankful for the advice, friendship and guidance of Andrea Davies and Nigel Green in their roles within the university.

Outside of my university studies, thanks goes to my high school science, maths and technology teachers, especially Mr Wright whose passion for science is a continuing inspiration. Additionally, to my friends from Elim, Edge and Vine churches who through getting me involved in sound, lighting and A/V productions spurned my interest in science, technology and engineering from a very young age. Of particular significance, is

the friendship of Shane Blackett who throughout my youth was always ready to answer every time I asked why.

To my family, my dad, my mum, and my sister thank you. For always pushing me to the next level. For always providing me with books and technology to keep learning. And for the support throughout the journey.

Finally, to anyone else reading this, and to anyone I've accidentally left off this list, thank you.

Image Credits

The photo of the two K10 Planetary Rovers (Fig. 1.2) is courtesy of NASA.
The photo of the UNSW CSE rUNSWift Robot Soccer Team (Fig. 1.1) is courtesy of Sean Harris.

Contents

List of Tables

List of Figures

Table of Symbols

r	a robot
R	a set of robots
m	the total number of robots
t	a task
T	a set of tasks
n	the total number of tasks
C	a cluster of tasks
o	the number of tasks in any single cluster
S	a set of all subsets of clusters
K	a set of clusters
k	the total number of clusters
l	the maximum number of tasks assigned per auction round
U	the set of unassigned clusters
λ	a cost
β	a bid
b_λ	the cost associated with an individual robot's bid
f	cost function for a set of robots
I	a set of individual local robot bids
B	the set of submitted robot bids
A_{ssi}	an SSI auction
N_{ssi}	the number of rounds in an SSI auction
A_{ssc}	an SSC auction
N_{ssc}	the number of rounds in an SSC auction
F	the set of completed tasks
V_C	a cluster's central vector
p	a single point location
c	a task collection point
d	a task delivery point

Chapter 1

Introduction

Efficient and effective teamwork among autonomous robots operating in dynamic environments is a complex and open problem in the field of artificial intelligence. Rapid technological developments are leading to autonomous robotic systems becoming more integrated into our everyday lives, performing important roles in domestic services [38, 65, 82], healthcare [77, 12] and transportation [36]. In a broader context autonomous robots are now actively being developed for use in harsh and complex environments, such as, underwater exploration [101], space exploration [15, 85] and disaster response [74, 108, 79, 88]. Many existing robotic systems are only effective at completing limited sets of pre-defined static tasks in isolation or working together on explicitly specific tasks in closed world systems. There are few robotic systems that are able to work in environments in which both the operating environment and tasks to be completed change frequently.

Dynamic environments introduce many challenges that are not found in closed systems. These challenges include: updating beliefs and knowledge about operating environments; incorporating current sensor information with existing prior knowledge to reason about actions; and the ability to alter plans to react to changes in the environment. Addressing this required functionality is computationally complex and incorporating these requirements into robots operating in dynamic real world environments is difficult.

Where single robots may struggle to effectively operate and complete tasks in highly dynamic environments, teams of robots operating in these environments may be able to distribute the computational load and improve the overall performance of the system. Teams of robots are also able to amalgamate knowledge and beliefs about the world from individual robots spread throughout the operating environment [39]. This combined knowledge allows erroneous data from noisy or malfunctioning sensors influencing plans and actions to be addressed [108]. Additionally, teams of robots can complete tasks in parallel and reduce the overall time required to complete a set of tasks.

Allocating tasks to robots to exploit these advantages in *multi-robot systems* (MRS) is in itself a difficult problem. The *multi-robot task allocation* (MRTA) problem is a complex NP-Hard optimisation problem and, in many instances, globally optimal solutions are difficult to find [102]. Because of this, the rapid generation of near optimal solutions to the problem that minimise task execution time and/or energy used by robots are highly desired [63]. This thesis is concerned with task allocation in multi-robot teams operating in dynamic environments. Our approach seeks to cluster together closely related tasks and then builds on existing distributed market-based auction architectures for distributing these sets of tasks among autonomous robots.

1.1 Contributions

The key contributions of this thesis are:

- **Development of *sequential single-cluster auctions* (SSC) auctions**. Many existing approaches for solving MRTA problems either allocate tasks sequentially one at a time or alternatively, attempt to allocate all tasks at the same time. Allocating tasks one at a time often fails to consider synergies between tasks,[1] while approaches that allocate all tasks at once are computationally complex. Our approach provides a middle ground that sequentially allocates clusters of tasks with good synergies, is relatively computationally simple and therefore runs quickly.

- **Development of repeated SSC auctions with dynamic task clusters**. Many sub-optimal auction-like approaches for MRTA problems have non-reversible assignments of tasks to robots which can generate poor solutions. In our approach we allow the repeated formation of task clusters containing different tasks. This allows us to explore additional synergies between tasks and, through the reallocation of tasks in this process, the solution costs can be gradually improved.

- **Analysis of algorithms for task cluster formation**. There are a number of techniques for task cluster formation. Each of these techniques seeks to balance synergies between tasks and the time required to calculate and form clusters. As a result, when these clusters are used for task allocation, the cost of the MRTA solution is also affected. This thesis provides an analysis of these techniques and additionally explores the influence of using different inter-task distance metrics in cluster formation.

[1] Inter-task synergies arise in a situation where the cost for one robot to complete two or more tasks is cheaper than the combined cost of separate robots completing each task individually.

- **Approaches for handling online dynamic task insertion.** In dynamic environments, it is common for additional tasks to be inserted into a system after an initial solution to the MRTA problem is determined. Allocating these additional tasks to robots and reallocating tasks, if necessary, is important for maintaining low cost solutions. In this work we study the trade-off in solution quality between an individual robot replanning its task execution and a global reallocation of all tasks in the system.

- **Approaches for reallocation of tasks upon robot failure**. In dynamic environments, it is highly likely that over time robots will suffer some form of failure. Responding to and reallocating tasks upon robot failure is important in completing MRTA problems. In a similar vein to task insertion, determining how much of the tasks in a system to reallocate is an important trade-off for consideration in reacting to robot failure. In this thesis we compare partial reallocation of a failed robot's tasks to a global reallocation of all tasks in the system.

1.2 Example Applications

MRTA problems can arise in a variety of different domains and scenarios. Our approach for solving MRTA problems is designed to be adaptable to a wide variety of domains. In this section, we describe five active areas of robotic research and the application of MRTA in these domains.

1) **Service Oriented Robots.** Consider a team of autonomous mobile robots operating in an office-like environment. These robots may be required to make deliveries, clean up spillages, or act as tour guides to visitors. In this scenario there may be many tasks to complete and some of these tasks may have high inter-task synergies, e.g. multiple delivery tasks. As such, it may be desirable for one robot to be allocated the delivery tasks and another robot to be allocated the clean up tasks.

   This scenario can also be highly dynamic. For instance, it is highly unlikely that the tasks requiring clean up are all known prior to allocation and as a result during execution new clean up tasks may need to be assigned [82]. Additionally, some tasks may have higher priority than others. For example, tour guide tasks may have higher priorities than delivery tasks. However, visitors may arrive early or late and, in order to maintain efficient usage of the robots, task allocations and execution orders may need to be continually re-evaluated to ensure efficient resource usage across the robot team.

Figure 1.1: Robot Soccer. / Photo: Sean Harris.

2) **Role Allocation in Multi-Robot Teams.** In team based activities there are often a number of individual tasks that need to be completed to achieve an overall goal. Additionally, during execution the roles assigned to individual robots may need to be revised to address changes in the environment [35]. For instance, consider a team of robots playing soccer (Figure 1.1). There is one global shared goal among the team and additionally the tasks that each robot performs in their individual roles—attacker, defender, goal keeper—contributes to the achievement of the global goal [40]. Depending on the location of the ball and other robots, each individual robot's current task and role may change regularly during play. For example, when the opposing team has the ball more robots may be assigned to the defending role and subsequently move into positions to prevent the opposing team from scoring.

3) **Search and Rescue.** Consider a team of robots that can be deployed into a collapsed building. In situations where it may be too dangerous to send in human rescue workers, robots can search through the debris to locate victims who require rescue. This information can then be used by human first responders to prioritise their rescue efforts. In this scenario the team of robots can rapidly distribute themselves throughout an area and additionally communicate with each other to share what areas of the building have been explored. Further extensions of this problem include systems with heterogeneous robots, in which, certain robots may have special abilities, such as, being able to fit through small and narrow voids or operate specialist cutting equipment.

Figure 1.2: NASA's Prototype K10 Planetary Exploration Rovers. / Photo: NASA.

4) Space Exploration. Cooperation between robotic teams is highly desired in exploration of remote worlds, such as, the Moon and Mars with harsh environments. For instance, consider the exploration of an unknown area in a remote world with a team of robots (Figure 1.2). Because of the delay and difficulties in communication between earth and remote planets, it is not feasible for the robots to be continuously controlled from Earth [85].

As an example, assume a satellite map has been provided to each of the robots and the robots are able to localise themselves within the map. Within the map a number of points of interest have been determined, however, they are far apart and the time required for a single robot to travel between all these points is high. A team of robots can allocate these points of interest and travel to these locations in parallel with other robots reducing the time required to explore the area [122]. Additionally, this distributed exploration task allows additional features of the remote world, such as hostile weather, to be experienced by the robots in unison allowing additional scientific data to be collected.

5) Transport Logistics. MRS are well situated to solving problems in transport logistics [18]. Consider a set of trucks transporting goods from warehouses to customers. Each truck is able to hold a certain capacity of goods and is able to collect from warehouses and deliver goods to customers in any order. Minimising the transportation time and fuel consumed maximises the profit. During execution, additional jobs may be inserted into the environment and trucks may face delays from traffic congestion [36]. This problem is very difficult to solve optimally as handling the dynamic tasks and adhering to each truck's capacity constraint is complex.

1.3 Thesis Outline

The remainder of this thesis is structured as follows. In the next chapter, we formally outline the task allocation problem and discuss related work in relation to interaction between robots, approaches to task allocation in multi-robot systems, and handling task reallocation in dynamic environments.

In Chapter 3 we describe SSC auctions and show how bidding for and awarding clusters of tasks can lead to lower cost task allocations than single item auctions. We then implement repeated SSC auctions with dynamic task clusters, in which on the completion of each individual task, robots create new clusters of uncompleted tasks that are then auctioned to reallocate tasks to incrementally revise and improve the MRTA solution.

We analyse four different cluster formation algorithms and their influence on the solution cost in SSC auctions in Chapter 4. Additionally, we compare metrics for measuring the distance between tasks during cluster formation and the impact of this on cluster formation time and the MRTA solution cost. Finally, we consider the priority allocation of clusters with many tasks before smaller clusters in the trade-off between minimising individual robot costs and minimising the global team cost.

Chapter 5 explores extensions to the standard MRTA problem. In particular, we consider tasks that require collection and delivery in dynamic environments. In these dynamic environments, reallocation of tasks during execution is required as additional tasks may arrive after the initial task allocation and robots may also fail while executing tasks.

Finally, Chapter 6 summaries the key contributions of this thesis and suggests areas of future research.

Chapter 2

Related Work

In this chapter we present an overview of related literature on task allocation techniques in systems with *distributed artificial intelligence* (DAI). We begin with high-level descriptions of the classifications of different systems with DAI and describe the variety of inter-agent interaction approaches in these systems (Section 2.1). We then formally describe the MRTA problem and some common extensions (Section 2.2). In Section 2.3 we discuss market-based approaches, such as bargaining markets and auctions, for solving MRTA problems. This section includes a discussion of approaches for reducing the total team costs through the reallocation of tasks in a running system and clustering sets of related tasks. Our final consideration is task allocation approaches in dynamic environments where tasks may be added to a running system or robots breakdown (Section 2.5) and tasks with collection and delivery locations (Section 2.6).

2.1 Distributed Artificial Intelligence

Distributed intelligence is defined by Parker as groups of agents *"working together to reason, plan, solve problems, think abstractly, comprehend ideas and language, and learn"* [83]. An *agent* is defined by Russell and Norvig as *"anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators"* [92]. In this definition an agent may be any form of intelligent being, including humans, complex computer systems, and autonomous robots. In this thesis we are interested in the study of autonomous artificial agents and robots. In particular, systems devoid of any human control, where the agents in the system may interact with humans, however, their choice of actions to take is entirely self-controlled and independent.

In its simplest form a *Multi-Agent System* (MAS) is a collection of many agents. Within this system these agents can interact with each other either to cooperate or compete to achieve individual or collective tasks [26]. Interaction is typically facilitated by communication through the exchanging of messages across a network [127] but other

methods such as passive observation and reaction to the behaviour of other agents allows interaction without explicit communication.

A MRS is built on the ideas incorporated in MAS. Farinelli, Iocchi, and Nardi define a MRS as *"a set of robots operating in the same environment"* [34]. They argue that a MRS is a lot more complex than a MAS as the use of robots introduces issues not found in MAS, such as, uncertainty and incompleteness of information acquired from the environment, errors in communication between robots, and hardware failures. One can consider each robot in a MRS having its own internal MAS (e.g. an agent for navigation, an agent for communication, etc.) which acts as a single combined entity interacting with the environment.

For example, consider a team of robots delivering mail in an office-like environment. The robots in this system will interact with each other to negotiate who delivers which piece of mail, but there may be no centralised controlling person or computer dictating the decisions and controls of the robots. To facilitate negotiation among robots each robot may value the cost of delivering each piece of mail differently from other robots. This difference in valuations allows robots to negotiate to complete mail deliveries that maximises individual values and/or swap deliveries of lesser value with those of greater value among other robots.

2.1.1 Classifications

Doran *et al.* presents Franklin's typology of cooperation in MAS (Figure 2.1) [26]. At the root level a MAS is classified into two branches:

- **Independent** *"if each agent pursues its own agenda independently of others"*; and
- **Cooperative** if the agenda of agents include *"cooperating with other agents in some way"*.

On the independent branch two leaves are specified:

- **Discrete** *"if the agendas of the agents bear no relation to one another"*; and
- **Emergent cooperation** where *"from an observer's viewpoint, the agents appear to be working together, but from the agent's viewpoint they are not"*.

For instance, emergent cooperation is seen in Dagaeff, Chantemargue, and Hirsbrunner where agents are programmed to be deliberately antagonistic towards other agents however cooperation emerges [17] and in Iba emergent cooperation appears in the evolution of agent behaviour using genetic algorithms in a MAS [49].

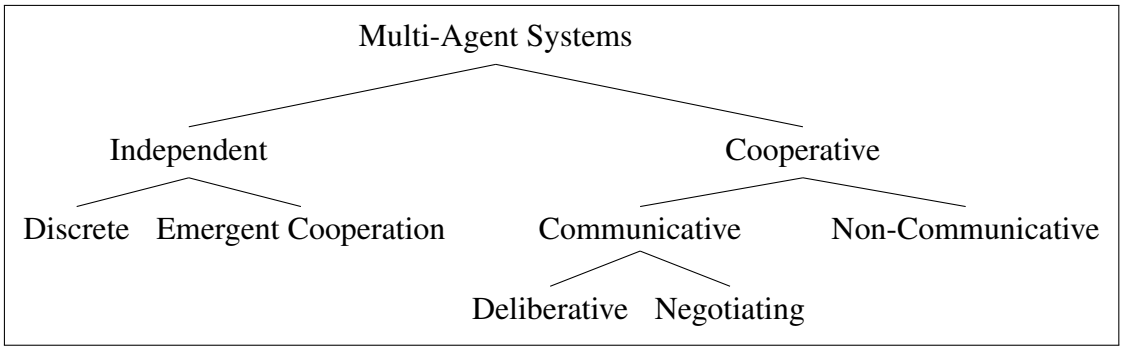


Figure 2.1: Franklin's Cooperation Typology [26].

Within the cooperative systems branch, MAS are divided into two further subclassifications. In *communicative* systems, agents cooperate via intentional messages being sent and received. This communication can take one of two forms, *deliberative* or *negotiating*. Deliberative systems see agents jointly plan any actions they take, which may or may not entail cooperative joint actions. Negotiating systems also involve communication in planning but agents can bid or compete against each other to achieve the best outcomes for themselves.

In *non-communicative cooperative* systems, agents cooperate through observing and reacting to the behaviour of others. This behaviour differs from that of independent emergent cooperation as agents have the ability to anticipate and assess the actions of other agents. Typically, this ability to anticipate and assess operates on the principle of *stigmergy* where changes in the environment provides information indirectly to other agents. An algorithm commonly used in these systems is *Ant Colony Optimisation* (ACO) [27].

Farinelli, Iocchi, and Nardi present a taxonomy of MRS (Figure 2.2) [34]. Their taxonomy follows a similar pattern of thought to that of Franklin's cooperative branch in classifying MAS. However, unlike Franklin's focus on communication, Farinelli, Iocchi, and Nardi's primary focus is on different forms of coordination among MRS. The authors present two different types of coordination: *strong coordination* represents MRS that rely on a coordination or communication protocol, whereas *weak coordination* represents those systems that do not operate via a protocol (e.g. ACO).

At the organisation level a distinction is made between the autonomy each agent has in the system's decision process. A *centralised* system has a leader who is solely responsible for organising the entire team of robots. This centralised decision process can be classified as *strong* where one leader has complete command during the entire mission or *weak* where a number of different leaders may have command during different periods of the mission. In a *distributed* system agents are completely autonomous in their decision making and no sole team leader exists.

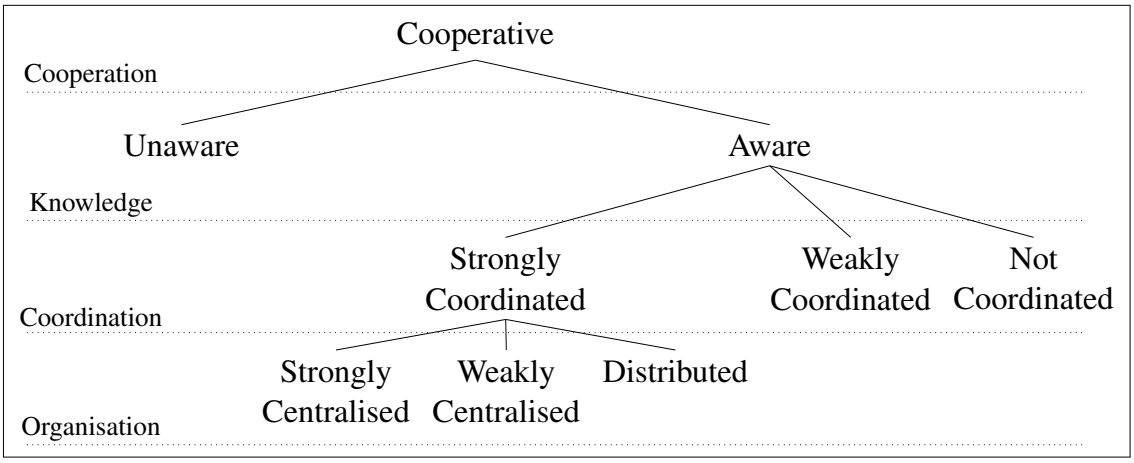


Figure 2.2: Farinelli, Iocchi, and Nardi's MRS Taxonomy [34].

When considering MAS and MRS the classification of the complete system is important in determining the behaviour of individual agents. The classification of the system typically determines the form of interaction individual agents have in their communication and cooperation with other agents. This is explored in detail in the following section.

2.1.2 Agent Interaction

Positive interaction between agents in both MAS and MRS plays an important role in determining the overall success of a system. Parker defines four common forms of interaction: *collective*, *cooperative*, *collaborative*, and *coordinative* (Table 2.1) [83]. The choice of the best type of interaction depends on the particular problem being solved. Most problems consist of a set of tasks to be completed. Tasks can be of two forms: *individual* and *shared*. Individual tasks can be *homogeneous* in that all agents perform the same actions, or *heterogeneous* in which each agent has a different criteria to satisfy.

When discussing interaction it is best to consider why interaction is required in the first place. It is possible to operate a MAS/MRS without any interaction between agents, however, as the system becomes more complex, the lack of interaction between agents can lead to bottlenecks, collisions and other problems as agents fail to account for each other. For example, consider a group of people scattered throughout a park, suddenly it begins to rain and everyone individually decides to seek shelter in the same location [107]. In this scenario the agents all have the same task, but there is no communication between the agents in deciding the actions to take. Now, let's consider that the shelter doesn't have enough space to keep everyone dry, however, there is a second shelter that is empty. Those agents that are unable to fit under the first shelter now have to go to the second shelter. In this example, if at the beginning the agents had communicated with each other, the group could have collectively divided into two with half going to each

Table 2.1: Parker's Interaction Categorisation [83].

Interaction	Awareness	Goal Type	Mutually Beneficial Actions
Collective	not aware	shared	yes
Cooperative	aware	shared	yes
Collaborative	aware	individual	yes
Coordinative	aware	individual	no

shelter. Jennings uses this example in the discussion of models of individual behaviour that include communication and coordination among other individuals [51].

Collective Interaction

Returning to Parker's forms of interaction, collective interaction is defined as where agents are not aware of other agents on the team, although they share tasks, and their actions are beneficial to their teammates. An example of this would be a team of robotic excavators digging a hole. The action of one agent removing dirt from the hole would be beneficial to all the other agents in the team. Systems using collective interactions often follow models that are biologically inspired, such as, flocking and herding. The most widely studied form of collective interaction is *swarming*.

Swarming is a type of MAS that shows emergent cooperation. In a swarm, all agents share the same task and act independently of each other. Despite this they often appear to be working cooperatively together. Trianni *et al.* bridge individual and collective interaction in their research which focuses on autonomous robots deciding if they should act as an individual or use the information that they have gathered from the environment to act with other robots as a robotic swarm [123]. In their research three robots are placed in a circle with two tasks, only one of which can be achieved: 1) individually leave the boundaries of the circle through a defined exit, or 2) group together as a swarm in the middle of the circle. The completion of either task is considered a successful achievement of the team objective and the best agents are those that complete either task in the shortest period of time. To improve the agents, the authors use genetic algorithms to evolve new individual agent controllers which seek to predict the actions of other agents.

Cooperative Interaction

Parker's second type of interaction is called *cooperative interaction* where agents are aware of other agents, they share tasks, and their actions are beneficial to their teammates. This type of interaction can take two forms: *joint task coordination* or *shared information coordination*[1] [110]. In joint task coordination agents work together to achieve a task that

[1] Smith and Davis [110] call this task-sharing and result-sharing respectively.

an individual agent could not achieve by itself, such as, box pushing [43] or coordinated robotic construction [117].

Shared information coordination involves agents acting independently but sharing information. One of the major challenges in shared information coordination is determining the correct amount of information to share. If too little information is shared between agents the system may fail or cease to perform adequately, conversely, if too much information is shared the system may be swamped in the communication process and the task is not achieved. In the domain of robotic search and rescue, Settembre *et al*. outline cooperative situation assessment *"that balances the use of communication bandwidth with the need for good situation assessment"* [108]. The basis of their approach is for each robot to form a plan of action, once they have formed a plan they randomly send this plan to another robot in the system. If this robot agrees with the plan they then randomly forward it on to another robot. This process continues until a fixed number of robots agree to the same plan and subsequently the plan is executed. If the receiving robot does not agree with the plan proposed, it will send back additional data supporting its reasons why it does not agree to the proposed plan. The original robot then decides to revise its plan based on this new data or find a different robot that will support the original plan.

There are two primary benefits of this approach to distributed planning. The first is low levels of communication overhead as there is only one-on-one communication between agents. The second is that by reasoning across multiple robots any errors in a single robot's sensors are filtered out. The authors point out that one limitation in their approach is when the environment in which the robots are operating is sparse, the state information and plans of each robot can be quite contradictory which degrades the performance of the algorithm.

Collaborative interaction

Collaborative interaction (also called *coalition formation*) is defined by Parker as *"agents have individual tasks, they are aware of their teammates, and their actions do help advance the tasks of others"*. An example of this sort of interaction in a MRS is within the domain of robot soccer (Figure 1.1). The overall objective of a team will be to win the game and within the team each robot will have individual tasks: the attacker to score, the defender to take the ball off the opposing team's attacker, and the goalkeeper to block any shots on goal. When each agent achieves their individual task it helps to contribute to the achievement of the overall team objective.

Vig and Adams explore coalition formation in a MRS with three scenarios in which teams of robots need to work together to achieve a task: *coordinated box-pushing*, *clean-up*, and *sentry duty* [125]. They argue that a discrepancy exists between coalition forma-

tion in MAS and its application to the MRS domain. Coalition formation in MAS assumes that all agents are available and communication is always possible. However, in a MRS it is possible that robots will be far enough apart that communication can not occur across the entire team but instead only subsets.

For the box-pushing task, two robots are required to push each large box. From a team of 12 robots, two boxes needed to be moved and two sets of two robots succeeded in working together to move both boxes. The clean-up task requires a group of robots clearing an area of small boxes in which a single robot would individually move one box at a time. The robots were able to successfully work together to clear the area of boxes. For the third task, sentry-duty, robots are required to navigate to a position and collaborate information to perform motion detection. Like the previous two tasks this was performed successfully. For a final task a team of robots was given all three tasks to perform simultaneously, this required sub-teams of robots within the MRS to form coalitions to achieve one of the tasks, and in this scenario, three sub-teams completed all three tasks. The authors highlight that as more tasks are allocated and fewer robots remain, the number of messages exchanged to form a coalition decreases. This suggests that the algorithm used in the experiment should scale well to larger systems.

Coordinative Interaction

Parker's final type of interaction is *coordinative*. In this interaction agents are aware of each other, but they do not share a common objective, and their actions are not helpful to other team members. For instance, this form of interaction is seen in systems where agents operate in an enclosed environment and need to move from one area to another in the shortest distance and shortest time. In this scenario it is in an agent's best interests to communicate with other agents to avoid a collision, but it would not be in its interest to give-way to many agents. This form of interaction typically sees agents acting against each other, which is in contrast to the previous three types of interaction.

2.2 Multi-Robot Task Allocation

We now consider the allocation of tasks to robots in cooperative and coordinated multi-robot systems. MRTA in its simplest form is the process of assigning individual tasks to a set of robots [14]. It is important to have well defined and efficient processes for distributing these tasks that aids in the achievement of individual and shared goals.

Within a large MRS the achievement of the global objective of the system may be dependent on the successful completion of a number of sub-tasks. The requirements and dependencies of these tasks can vary widely. The simplest tasks are *independent*, meaning

that they can be completed by one or more robots without any prior task needing to be achieved first. In other situations tasks are *dependent*, meaning prerequisite tasks must be successfully completed before the current task can begin. For example, a robot cooking a microwave meal must first complete the task of removing outside packaging from the meal before placing it in the microwave. In some situations a group of tasks need to be sequentially achieved together. Take, for example, a robot cooking pancakes. The robot must pour the batter into a pan, place the pan over heat, flip the pancake, place over heat again, and finally, remove the pancake from the pan. The robot cannot break this sequence and complete other tasks in the interim, if it did, the pancakes may burn.

2.2.1 Problem Formalisation

We formalise the definition of the standard MRTA problem with independent tasks in a manner that is similar to Sandholm [95] and Koenig *et al.* [57]. Given a set of robots $R = \{r_1, \ldots, r_m\}$ and a set of tasks $T = \{t_1, \ldots, t_n\}$. A partial solution to the MRTA problem is given by any tuple $\langle T_{r_1}, \ldots, T_{r_m} \rangle$ of pairwise disjoint bundles of tasks:

$$T_{r_i} \subseteq T \text{ with } T_{r_i} \cap T_{r_{i'}} = \emptyset, i \neq i', \forall i = 1, \ldots, m$$

This means that robot $r_i \in R$ performs the tasks T_{r_i} and no task is assigned to more than one robot. To determine a complete solution we need to find a partial solution where every task is assigned to exactly one robot:

$$\langle T_{r_1} \ldots T_{r_m} \rangle \text{ with } \cup_{r_i \in R} T_{r_i} = T$$

The standard testbed of the MRTA problem is multi-robot routing [24]. The tasks represent locations to visit. We assume robots have perfect localisation and can calculate the cost λ to travel between locations. The travel cost between any two locations are symmetric:

$$\lambda(t, t') = \lambda(t', t)$$

and equal across all robots. The robot cost $\lambda_{r_i}(T_{r_i})$ is the minimum cost for an individual robot r_i to visit all locations T_{r_i} assigned to it. The specific ordering of tasks resulting in the minimum cost for each robot is called its task execution plan. There can be synergies between sets of tasks, such that:

$$\lambda_{r_i}(T_{r_i}) + \lambda_{r_i}(T_{r'_i}) \neq \lambda_{r_i}(T_{r_i} \cup T_{r'_i})$$

A positive synergy is when the combined cost for a robot to complete two tasks is lower than the individual costs for the robot to complete each task:

$$\lambda_{r_i}(T_{r_i} \cup T_{r'_i}) < \lambda_{r_i}(T_{r_i}) + \lambda_{r_i}(T_{r'_i})$$

Approximation Methods and Heuristics

Calculating the optimal minimum individual robot cost and path to visit all assigned tasks is NP-hard [63, 122]. In many situations this is infeasible, as an alternative, approximation methods and heuristics are commonly used to calculate near optimal costs. Some heuristics take advantage of inter-task synergies allowing robots to factor in the cost of completing any current allocations when determining the assignments of additional tasks. Commonly used heuristics for solving MRTA problems in related literature include:

- *Cheapest-insertion heuristic* which inserts an additional task to be completed into an existing task execution plan, at the point in the plan where, the cost to complete the task plan increases by the smallest amount and no other reordering of tasks in the plan occurs. This approach is simple and is widely used [62, 63, 122, 56, 57, 58, 120].

- *2-opt improvement rule* which iteratively reverses the execution order between two tasks if this reduces the cost to complete the overall task execution plan. This approach is often combined with the cheapest insertion heuristic as a means of reducing costs after the insertion of a new task [139, 122, 57, 58, 120].

- *Rapidly-exploring random trees* which find the shortest paths between key points in a map allowing for quick cost calculations when path planning [80, 81].

- *Minimum spanning trees* which connect all robots and tasks and when converted from a tree to a path provides bounds on the path distance [62].

- *D* lite* which attempts to find the minimum cost to travel between locations. As full knowledge of an environment or costs between locations may not be known additional replanning occurs as more data becomes available [8].

Extensions

There are many extensions to the standard MRTA problem. Common extensions include:

- *Variable task costs* where the cost to travel to or execute a task changes during task execution [122, 56, 30].

- *Capacity constraints* where robots are load balanced and are allocated at most a fixed number of tasks [57, 58, 136].
- *Task dependence and precedences* where robots can only complete certain tasks after some other tasks are completed [45, 10, 43, 80].
- *Time limited tasks* where robots must complete tasks within fixed time windows [46, 88].
- *Task preferences* where robots have a preferences for certain tasks.
- *Task/Resource requirements* where robots need to have particular abilities or resources to complete tasks [43, 120].
- *Complex tasks* where tasks can be decomposed into simpler tasks that all need to be achieved, however, task decomposition may differ according to the capabilities of the robot that the complex task is assigned to [139] or tasks in which multiple robots are required to form coalitions to complete the task [88, 137].

2.2.2 Team Objectives

Generating a valid solution to the standard MRTA problem is not difficult. For instance, a simple approach is to assign each task in turn to a randomly selected robot. However, this approach gives no guarantees on the execution time, energy or resources used in completing the assigned tasks. Subsequently, the application of *team objectives* arises to provide additional guidance in the search for solutions to the task allocation that meet certain criteria. For instance, some common desires of a multi-robot system are minimising time spent in execution of tasks, minimising energy or fuel consumed, and/or even distribution of tasks across all robots.

Lagoudakis *et al.* discusses team objectives in detail and their application to MRTA [63]. In this work they express the generic team objective function as:

$$\min f(b_\lambda(r_1, T_{r_1}), \ldots, b_\lambda(r_m, T_{r_m}))$$

where b_λ is the cost or bid calculation function for each individual robot to complete all allocated tasks and f is the cost function of the set of all robots. In this thesis we use two commonly used team objectives [20, 21, 8, 62, 63, 122, 102, 56, 138, 57, 75, 58, 120, 81, 137, 106]:

MiniMax $\min \max_{r_i \in R} \lambda_{r_i}(T_{r_i})$ that is to minimise the maximum distance any individual robot travels.

MiniSum $\min \sum_{r_i \in R} \lambda_{r_i}(T_{r_i})$ that is to minimise the sum of the paths of all robots in visiting all their assigned locations.

The application of these two team objectives to solving the MRTA problem can generate vastly different allocations of tasks to robots. The MiniMax team objective can be considered as the Min-Max Vehicle Routing Problem or the Makespan problem [122] and the MiniSum team objective can be considered as a multi-robot version of the Travelling Salesperson Problem [64, 62, 122].

2.2.3 Properties

We now consider some properties of the number of tasks and robots in MRTA problems:

Theorem 2.1. *As the number of robots $|R|$ increases, the new optimal solution cost $\lambda(T)_{new}$ will not be greater than the previous optimal solution cost $\lambda(T)_{prev}$.*

Proof. The optimal solution cost will not increase because the previous allocation and solution cost will remain valid as we can assign no tasks to the new robot $r_{\text{new}} \leftarrow \emptyset$. □

The optimal solution cost may decrease, and we show this using the example given in Figure 2.3. In this example we assume that the first allocation is done with only one robot $R = \{r_1\}$. Using the MiniSum team objective the minimal cost to complete all tasks $T = \{t_1, \ldots, t_4\}$ is $\lambda(T)_{\text{prev}} = 14$. If we now increase the number of robots so that $R = \{r_1, r_2\}$, the optimal allocation is $r_1 \leftarrow \{t_2, t_4\}$ and $r_2 \leftarrow \{t_1, t_3\}$ and the minimal combined cost is $\lambda(T)_{\text{new}} = 11$.

Contrary, as the number of robots decreases the optimal solution cost will not decrease. The optimal solution cost will not increase if the robot removed has no tasks assigned $r_{\text{removed}} = \emptyset$. However, if the removed robot had tasks assigned $r_{\text{removed}} \neq \emptyset$ these tasks will need to be assigned to other robots to ensure a new valid solution to the MRTA problem. The reassignment of these tasks may cause the optimal solution costs to increase, as fewer robots are required to do more tasks. Decreasing the number of robots is therefore equivalent to increasing the number of tasks to complete.

The following is a straightforward consequence of the assumption that path costs are non-negative and robot costs are monotonic. Any additional task allocations require the modification of at least one robot's task execution plan. Given we are minimising costs across all robots the optimal cost of the previous allocation plus any additional tasks cannot be less than the cost of the previous allocation:

t_1				r_1			t_2
t_3				r_2			t_4

Figure 2.3: Exploration Task 3 from Koenig *et al.* [56] (distance between cells is 1 unit, robots move horizontally and vertically).

Theorem 2.2. *Let T_{prev}, T_{new} be two sets of tasks and $\lambda(T_{prev}), \lambda(T_{new})$ their respective optimal solution costs. If $T_{prev} \subseteq T_{new}$, then $\lambda(T_{prev}) \leq \lambda(T_{new})$.*

Proof. Suppose for contradiction that $\lambda(T_{\text{new}}) < \lambda(T_{\text{prev}})$. Since path costs are non-negative, the solution represented by $\lambda(T_{\text{new}})$ would be a valid solution for T_{prev}. Hence $\lambda(T_{\text{prev}})$ is not an optimal solution cost for T_{prev}. Contradiction. □

2.2.4 Distributed Approaches to Multi-Robot Task Allocation

Lagoudakis *et al.* show that finding an optimal solution to the MRTA problem for the MiniMax and MiniSum team objectives is NP-hard [63]. Therefore, much research focus considers efficient approximation algorithms for solving MRTA problems with large numbers of tasks and robots. Solutions to MRTA problems can be found using centralised methods, such as, mixed integer programming [62, 57, 30] or graph partitioning [111, 67]. However, in all but the simplest problems, centralised methods are not efficient for MRTA [20, 15]. In particular, they introduce a single point of failure into the system, where *"if the central controller fails, so does the entire robot team"* [122]. In addition, the information required by the centralised controller in decision making introduces large, and generally impractical, communication overheads into the system. This problem is particularly important in dynamic environments, where delays in decision making can lead to robots in dangerous situations being unable to react swiftly [22]. On the contrary, fully distributed systems with no communication between robots are generally fault tolerant and highly reactive to changes in the environment. However, relying solely on local information and planning without any coordination among robots leads to highly inefficient solutions to MRTA problems [21].

In between fully centralised and fully non-communicative individual approaches for MRTA are approaches such as token-passing [105] and markets [24, 131]. Dias and Stentz provide a comparative study of centralised and distributed approaches to solving MRTA problems [22]. In this work a centralised system with a depth-first search (e.g. Figure 2.4) is used to produce an optimal task allocation for all robots is compared to a distributed

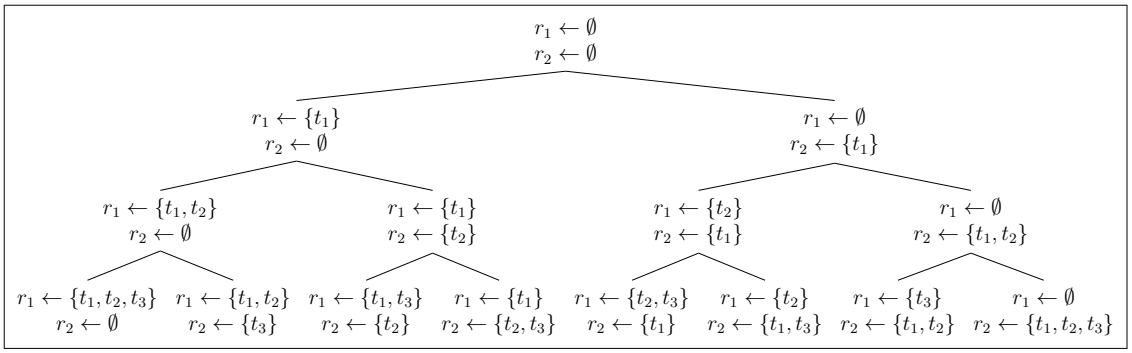


Figure 2.4: Centralised Search Tree for MRTA solutions with two robots and three tasks.

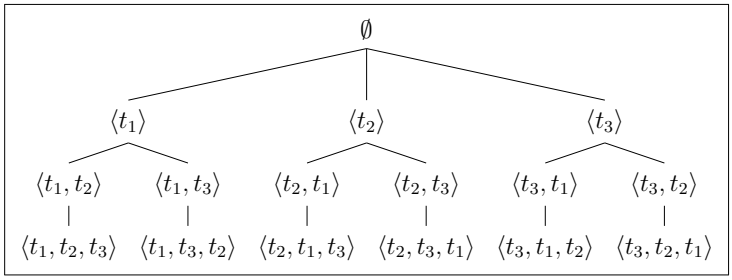


Figure 2.5: Search Tree for Task Execution Paths with three tasks.

system where each individual robot also performs a depth-first search (e.g. Figure 2.5) to find an optimal solution path to individually complete all tasks. In the distributed system when any robot completes any task, they communicate this to all other robots, and on notification of this task completion, all individual robots replan paths to complete all remaining tasks.

In addition to this, Dias and Stentz describe a market-based approach to MRTA problems. In the market-based approach robots initially negotiate for tasks they wish to execute and, during execution, robots are able to trade tasks one-for-one with other robots. This approach allows robots to complete the tasks they desire and offload the tasks they do not wish to complete. In their experiments they compare the solution quality and time required for each of the task allocation approaches with the fully distributed solution performing the worst. The market-based solution was shown to generate solutions that were close to the optimal in the smallest time of the three approaches.

2.3 Markets

A common distributed approach for the allocation of tasks in multi-robot systems is through the use of a market. Dias *et al.* [24] lists the standard features of a market environment as:

- A global problem with tasks that are achievable by individual robots or robot sub-

teams (e.g. MRTA).

- A team objective function which gives preference to a particular solution to the global problem.
- Robots have a cost function which ranks and selects individual tasks relative to the team objective.
- A mapping between the team objective and individual tasks such that the completion of tasks aids the advancement of the overall problem.
- A mechanism for the distribution and redistribution of tasks among robots. This mechanism accepts offers from agents for tasks and subsequently allocates the tasks in a manner that achieves the team objective.

2.3.1 Contract Net Protocol

The *Contract Net Protocol* (CNP) forms the foundation of many distributed market systems. Originally designed for distributed computing, this protocol is based on the concept of a distributed system in which there is no central control or data storage. Agents are independent and loosely coupled, meaning, they *"spend most of their time in computation rather than communication"* [109, p. 1104].

The basic workflow of the CNP is as follows: an agent has a task that needs to be completed. The agent sends a task announcement message to other (one, a subset, or all) agents describing the task and requesting bids for the completion of the task. Other agents respond to the request by calculating bids for the task and submitting this to the original agent. The original contracting agent then evaluates these bids and assigns the task to the agent it determines to be the winner.

There are a number of direct implementations of the CNP. For instance, Sandholm provides an implementation of the CNP for vehicle routing that is based on marginal cost calculations [94]. In this work CNP is extended to allow clusters of tasks to be bid on and assigned. Botelho and Alami apply the CNP to a simulated MRS domain [10], robots are assigned tasks one at a time, they are able to indicate tasks they wish to complete in the future and other robots are able to offer counter offers for these tasks. Gerkey and Matarić apply CNP to a physical MRS [43]. In this work robots subscribe to receive task allocation messages according to their abilities and resources available to complete tasks. When a new task is inserted into the system it is announced to robots who have subscribed to the resource channels required for the task. Robots then calculate and submit a bid for the task and the auctioneer announces the winner. Furthermore, if the winner fails to complete the task in an agreed time-frame the auctioneer can reassign the task. Finally,

Fischer *et al.* [37] develop the *extended contract net protocol* (ECNP) for situations where the tasks in a contract may exceed the capacity of a single bidder. The ECNP has been tested in a transportation domain with time windows and is further discussed in Section 2.6.

2.3.2 Bargaining Markets

In bargaining markets robots exchange tasks through direct negotiation. Each robot may consider the value of certain sets of tasks higher than other robots. In situations where robots have existing allocations of tasks this discrepancy in the valuation of tasks allows for the trading of the tasks between robots. To successfully trade, robots negotiate to form agreements on the exchange of tasks. Wooldridge [127] outlines three different forms of negotiation:

- *One-to-one negotiation*. One robot negotiates over the exchange of tasks with only one other robot.
- *Many-to-one negotiation*. One robot offers a task and negotiates with many other robots to determine the best deal.
- *Many-to-many negotiation*. Robots negotiate with many other robots in parallel. This approach to trading allows robots to be negotiating on a number of different deals at the same time.

Golfarelli, Maio and Rizzi adapt the CNP for task swapping with many-to-one negotiation [45, 44]. In this approach a robot's bid becomes a task t_{out} which the robot r_i is willing to exchange for the task t_{in} that the auctioneer is offering. A task swap is only agreed to if both the auctioneer's and the bidder's cost is improved. They highlight that when robots calculate individual bid with marginal costs, that is, the difference in cost between completing their current task allocation and committing to an additional task:

$$b_{\lambda_{\text{marginal}}} = \lambda_{r_i}(T_{r_i} \cup \{t_{\text{in}}\}) - \lambda_{r_i}(T_{r_i})$$

May result in different robot costs and task allocations than bidding with task exchanges, that is the difference in cost between completing their current task allocation and committing to an additional task in lieu of a currently allocated task:

$$b_{\lambda_{\text{exchange}}} = \lambda_{r_i}(\{T_{r_i} \backslash \{t_{\text{out}}\}\} \cup \{t_{\text{in}}\}) - \lambda_{r_i}(T_{r_i})$$

Task exchanges can overcome some local minima that marginal cost bidding is prone to becoming stuck in. However, single task exchanges can still become stuck in local

t_1		r_1		r_2		t_2	r_3		t_3

Figure 2.6: Simple task allocation problem with three robots and three tasks (distance between cells is 1 unit).

Table 2.2: Task allocations for MRTA problem given in Figure 2.6.

Robot	Initial Task Assignment	Initial Cost	Optimal Task Assignment	Optimal Cost
r_1	t_3	7	t_1	2
r_2	t_1	4	t_2	2
r_3	t_2	1	t_3	2

minima [95]. In further work, Goldarelli and Rizzi [46] extend task swaps to clusters of tasks $C \subset T$ and $C_{\text{out}} \subseteq T_{r_i}$, such that:

$$b_{\lambda_{\text{cluster}}} = \lambda_{r_i}(\{T_{r_i} \backslash C_{\text{out}}\} \cup C_{\text{in}}) - \lambda_{r_i}(T_{r_i})$$

where the number of tasks in the clusters exchanged may not be equal:

$$|C_{\text{out}}| \neq |C_{\text{in}}|$$

Sandholm explores one-to-one negotiation with *S-contracts* and many-to-many negotiation with *M-contracts* [95]. An S-contract involves two robots agreeing to swap equal numbers of tasks so that their individual task completion costs decrease. An M-contract involves three or more robots negotiating a set of task exchanges. Any contract is only valid if all parties agree to the task exchanges. However, when robots act with self-interest to minimise their individual cost a globally optimal solution to the MRTA problem may not be achieved (Sandholm's propositions 5 and 7 [95]).

To demonstrate this consider the grid world environment of Figure 2.6. An initial allocation of tasks to robots is given in Table 2.2. Task exchanges with S-contracts will never achieve the optimal solution to this problem. This is because robot r_3 will never agree to any task exchange as its local cost will always increase. An M-contract to achieve the optimal solution in this example is robot r_1 gives task t_3 to robot r_3, robot r_2 gives task t_1 to robot r_1, and robot r_3 gives task t_2 to robot r_2. However, robot r_3 will also never agree to this task exchange as its costs will increase.

In recent work on task exchanges, Sung, Ayanian and Rus consider robots completing assigned tasks without knowledge of other robots or tasks [119]. When a robot comes into the communication range of another robot, the robots negotiate to swap their assigned tasks if this reduces the local task completion costs of each robot. In this approach,

robots are able to reach a steady state in the cost to complete each task. The authors then demonstrate the suitability of this approach for ensuring task completion in systems with robot failure. While this approach does not guarantee optimal solutions to MRTA problems, it seeks to ensure that all tasks are completed in dynamic environments.

2.4 Auctions

Auction-based methods are a popular approach for solving MRTA problems [24]. An auction is composed of three separate phases: the *initial phase* in which an auctioneer sends a request to the robots indicating the tasks up for auction; a *bidding phase* in which each robot evaluates the tasks up for auction and responds with a bid for those in which it is interested; and, a *winner determination* phase in which the auctioneer determines the winner for each task. Auctions offer a good middle ground between centralised and fully distributed approaches to task allocation. In particular, auctions allow individual robots to maintain private information (e.g. their state, current commitments, etc.) while sharing information that aids the team (e.g. estimated task completion costs).

In a MRTA auction every robot bids on the set of tasks available and is awarded tasks according to their bids. Bids are calculated using bidding rules which enable individual robots to measure the costs of tasks relative to the team objective. Additionally, some auction algorithms allow inter-task synergies to be considered during bid calculations. Generally, robots bid using first-price sealed-bids, that is, each robot bids once for each task or bundle of tasks and robots are not aware of the bids made by other robots [11]. Furthermore, auctions can be run without any centralised auctioneer if all robots send all bids to each other and in parallel perform the same winner determination routine [62, 63, 122]. Common auction types include *parallel*, *sequential*, *combinatorial*, and *sequential single-item (SSI)* auctions. Table 2.3 provides a comparison of some popular types of auctions and their properties.

Table 2.3: Auction Type Comparison (worst-case costs).

Auction Type	Inter-task Synergies	Bids per Auction	Bid Calculations per Robot	Winner Determination Calculations per Auction
Parallel	no	$\|T\|\|R\|$	$\|T\|$	$\|T\|\|R\|$
Sequential	some	$\|T\|\|R\|$	$\|T\|$	$\|T\|\|R\|$
Combinatorial	all	$2^{\|T\|}\|R\|$	$2^{\|T\|}$	$\|R\|^{\|T\|}$
SSI	some	$\|T\|\|R\|$	$\|T\|(\frac{\|T\|+1}{2})$	$\|T\|\|R\|$

2.4.1 Parallel Auctions

Parallel auctions are very simple with bid and winner determination calculations requiring minimal computation. Each robot calculates bids for each task individually and no inter-task synergies are considered. Every robot then submits the bids for each task to the auctioneer who, in turn, determines a winner for each task according to the lowest bid submitted for it. In the event of a tie, the winner is determined in a systematic manner applied uniformly across all robots (n.b., this applies to all the following auction mechanisms described).

Returning to the example MRTA problem with two robots and four tasks in a grid-world environment (Figure 2.3), the bids submitted — if we use a parallel auction with the MiniSum team objective — are presented in Table 2.4. After bidding, the robot r_1 is awarded tasks t_1 and t_2, and the robot r_2 is awarded tasks t_3 and t_4 with a minimum total path cost $\sum_{r_i \in R} \lambda_{r_i}(T_{r_i}) = 20$ with this allocation.

Because parallel auctions consider no inter-task synergies when bidding, their solutions costs are often highly sub-optimal and unbounded in the worse case [56]. This is clearly seen in the above example, for instance, robot r_1 was allocated tasks t_1 and t_2 after bidding costs 4 and 3 respectively. However, robot r_1's overall cost $\lambda_{r_i}(T_{r_i})$ for completing these two tasks is 10.

2.4.2 Sequential Auctions

Sequential auctions are also very simple with winner determination requiring minimum computation. One task per auction round is offered, bid on, and awarded. Robots take into consideration the inter-task synergies of tasks they have been allocated in previous rounds [11]. As such, the order in which tasks are offered for auction heavily influences the final allocations and costs to the robots. The number of bids submitted per auction and the winner determination calculations required to allocate each task are the same as parallel auctions. This means that, while bid calculations are marginally slower due to the consideration of inter-task synergies, the overall auction process runs very quickly.

We now present a sequential auction for Koenig's example MRTA problem (Figure 2.3). To achieve the MiniSum team objective, robots bid with marginal costs on each task offered for auction, this also allows some inter-task synergies to be considered. The bids submitted each round are shown in Table 2.5. Subsequently, robot r_1 is awarded task t_1 in the first round, and the cost of doing this task is considered in the bid calculation for additional tasks. As a result, in round two, task t_2 is awarded to robot r_2, which immediately results in a different task allocation than parallel auctions for this example. In round three, task t_3 is awarded to robot r_1, and in the final round, task t_4 is awarded to

Table 2.4: Parallel Auction Example (winning bids and assignments in bold).

Task	r_1 Bid	r_2 Bid
t_1	**4**	6
t_2	**3**	5
t_3	6	**4**
t_4	5	**3**
$\lambda_{r_i}(T_{r_i})$	10	10

Table 2.5: Sequential Auction Example (winning bids and assignments in bold).

Round	Task Offered	r_1 Bid	r_2 Bid
1	t_1	**4**	6
2	t_2	6	**5**
3	t_3	**2**	8
4	t_4	7	**0**
	$\lambda_{r_i}(T_{r_i})$	6	5

t_1				r_1			t_2
t_3				r_2			t_4

Figure 2.7: Exploration Task 4 (Koenig *et al.* [56]).

robot r_2. The minimum total path cost $\sum_{r_i \in R} \lambda_{r_i}(T_{r_i})$ of this allocation is 11 which is the optimal solution for this particular problem.

It is important to stress that optimality is not guaranteed. Take for example, Koenig's *exploration task 4* (Figure 2.7). This example is very similar to our previous example, however, now both parallel (Table 2.6) and sequential auctions (Table 2.7) generate the same sub-optimal result. There are some extensions to sequential auctions that can avoid some local minima, for instance, Mosteo and Montano consider the reallocation of existing task allocations by modifying the auction rules so that after each task assignment an individual robot can choose an allocated task they no longer want and offer this for auction to other robots before the next new task is offered [75]. Alternatively, Viguria, Maxa and Ollero allow robots to offer subsets of tasks for reallocation after each task allocation until the task allocation becomes stable [126]. However, despite the increase in inter-task synergies considered in these approaches, the solutions to the MRTA problem can remain highly suboptimal.

Table 2.6: Parallel Auction Example for Exploration Task 4.

Task	r_1 Bid	r_2 Bid
t_1	**4**	8
t_2	**3**	7
t_3	8	**4**
t_4	7	**3**
$\lambda_{r_i}(T_{r_i})$	10	10

Table 2.7: Sequential Auction Example for Exploration Task 4.

Round	Task Offered	r_1 Bid	r_2 Bid
1	t_1	**4**	8
2	t_2	**6**	7
3	t_3	14	**4**
4	t_4	15	**6**
	$\lambda_{r_i}(T_{r_i})$	10	10

2.4.3 Combinatorial Auctions

In combinatorial auctions robots calculate bids for subsets of the tasks on offer with inter-task synergies considered. Optimal solutions to the MRTA problem can be found using a single-round combinatorial auction where each robot is allocated at most one disjoint subset of tasks. However, this method is NP-complete and the computation tends to be intractable. It is therefore, not feasible for anything but the smallest scenarios [8, 96].

During the bidding phase, robots calculate the costs for all task subsets. For an optimal task allocation, robots must calculate bids for every subset of tasks $S = \{C \subseteq T\}$ and $|S| = 2^{|T|}$. The solution to the MRTA problem is then a tuple of $|R|$ pairwise disjoint subsets that ensures that one and only one subset is assigned to each robot $T_{r_i} \in S$ and $\cup_{r_i \in R} T_{r_i} = T$. For optimal winner determination with a team objective, we must find a tuple of allocations that globally minimises (or maximises) the cost function f over the set of all robots. A simple, but highly inefficient, approach to achieve this is to exhaustively search every tuple of pairwise disjoint combinations that are solutions to the MRTA problem — this requires $|R|^{|T|}$ tuples to be checked.

We demonstrate an optimal MRTA combinatorial auction using Koenig's exploration task 3 (Figure 2.3). The bids each robot calculates for every subset of tasks is presented in Table 2.8. For winner determination the union of every pairwise disjoint subset that forms the complete set is calculated (Table 2.9). The optimal task allocations are the rows of this table with the minimum total cost.

An exhaustive search of all tuples is impractical in computational time for most situations. As a result robotics researchers have developed strategies to bid on limited subsets of tasks, such as, bidding on bundles of n or fewer targets [8, 76, 96], or forming greedy bids on bundles of short paths [8]. However, even if the number of bids is reduced and a near optimal solution is accepted, winner determination still remains NP-complete [96]. To improve the feasibility of winner determination researchers have considered methods of searching the bid set using branching [99, 98, 100], heuristics (e.g. stochastic local search [48]), and anytime algorithms [97, 96].

Table 2.8: Combinatorial Auction Bidding Example.

Task Subset	r_1 Bid	r_2 Bid
$\{t_1\}$	4	6
$\{t_2\}$	3	5
$\{t_3\}$	6	4
$\{t_4\}$	5	3
$\{t_1, t_2\}$	10	12
$\{t_1, t_3\}$	6	6
$\{t_1, t_4\}$	13	12

Task Subset	r_1 Bid	r_2 Bid
$\{t_2, t_3\}$	12	13
$\{t_2, t_4\}$	5	5
$\{t_3, t_4\}$	12	12
$\{t_1, t_2, t_3\}$	12	13
$\{t_1, t_2, t_4\}$	13	12
$\{t_1, t_3, t_4\}$	13	12
$\{t_2, t_3, t_4\}$	12	13
$\{t_1, t_2, t_3, t_4\}$	14	14

Table 2.9: Combinatorial Auction Winner Determination Example (winning bids and assignments in bold).

r_1 Task Subset	r_2 Task Subset	Total Cost
$\emptyset$	$\{t_1, t_2, t_3, t_4\}$	14
$\{t_1\}$	$\{t_2, t_3, t_4\}$	17
$\{t_2\}$	$\{t_1, t_3, t_4\}$	15
$\{t_3\}$	$\{t_1, t_2, t_4\}$	18
$\{t_4\}$	$\{t_1, t_2, t_3\}$	18
$\{t_1, t_2\}$	$\{t_3, t_4\}$	22
$\mathbf{\{t_1, t_3\}}$	$\mathbf{\{t_2, t_4\}}$	**11**
$\{t_1, t_4\}$	$\{t_2, t_3\}$	26

r_1 Task Subset	r_2 Task Subset	Total Cost
$\{t_2, t_3\}$	$\{t_1, t_4\}$	24
$\mathbf{\{t_2, t_4\}}$	$\mathbf{\{t_1, t_3\}}$	**11**
$\{t_3, t_4\}$	$\{t_1, t_2\}$	22
$\{t_1, t_2, t_3\}$	$\{t_4\}$	15
$\{t_1, t_2, t_4\}$	$\{t_3\}$	17
$\{t_1, t_3, t_4\}$	$\{t_2\}$	18
$\{t_2, t_3, t_4\}$	$\{t_1\}$	18
$\{t_1, t_2, t_3, t_4\}$	$\emptyset$	14

2.4.4 Sequential Single-Item Auctions

SSI auctions generate MRTA solutions over multiple bidding rounds. In each auction round each robot calculates bids for all unallocated tasks and submits a single bid, generally the smallest, for an unallocated task of its choosing. At the conclusion of each bidding round, the robot that bid the least is awarded the task it chose, such that, the overall team cost increases the least according to the team objective.

SSI auctions are not guaranteed to generate optimal solutions even if the robot costs are calculated optimally [122]. However, solutions generated relative to a variety of team objectives are bounded [62, 63, 56], they run in polynomial time [56], and have been shown experimentally to perform very well [62, 122]. SSI auctions require more bid calculations than standard parallel and sequential auctions ($|T|(\frac{|T|+1}{2}) > |T|$), however, the winner determination time is the same ($|T||R|$). Furthermore, the true number of bid calculations per auction is generally lower than $|T|(\frac{|T|+1}{2})$. This reduction is achieved through caching bid calculations. As only one task is allocated in each auction round, only the robot who had a task awarded in the immediate previous auction round needs to

function SSI-Auction ($\bar{T}$,T_{r_i}, r_i, R)
Input: $\bar{T}$: the set of tasks to be assigned
T_{r_i}: the set of tasks presently assigned to robot r_i
r_i: the robot
R: the set of robots
Output: T_{r_i}: the set of tasks assigned to the robot r_i

1: **while** ($\bar{T} \neq \emptyset$)
2: /* *Bidding Stage* */
3: $\beta_{min} \leftarrow \infty$
4: **for each** task $t \in \bar{T}$
5: $\beta^t_{r_i} \leftarrow$CalcBid(T_{r_i},t);
6: $\beta_{min} \leftarrow$min($\beta_{min}, \beta^t_{r_i}$);
7: Send(β_{min}, R) | $B \leftarrow \bigcup_i$Receive(β_{min_i}, R);
8: /* *Winner Determination Stage* */
9: $(r', t) \leftarrow$ arg min$_{(r' \in R, t \in \bar{T})}$ bid(r', t);
10: **if** $r_i = r'$ **then**
11: $T_{r_i} \leftarrow T_{r_i} \cup \{t\}$;
12: $\bar{T} \leftarrow \bar{T} \backslash \{t\}$;

Figure 2.8: Algorithm for Sequential Single-Item Auctions.

calculate new bids. Any other robots that bid on the task awarded in the previous auction round simply select their next best bid for a different task and all other robots resubmit their bid from the previous round.

We give a distributed algorithm for SSI auctions that each robot runs locally in Figure 2.8. This algorithm assumes a set of robots which are supplied with a map of the environment, have perfect localisation, have error free communication with other robots, and do not break down. These constraints are applied to enable us to focus simply on the auction process. As an aside, to handle more complex dynamic environments, modifications can be made to the auction algorithm to consider noisy data and problems in communication.

The SSI auction begins and continues while there are unassigned tasks (Line 1). During the bidding stage (Lines 2-7) the robot calculates bids for every unassigned task and submits its single lowest bid for any one task to all other robots. The function CalcBid takes the set of previously assigned tasks T_{r_i} to robot r_i and the task t being bid on and uses a bidding rule to calculate a bid cost (Line 5). Each bid is a triple $\beta = \langle r_i, t, b_\lambda \rangle$ of a robot $r_i \in R$, a task $t \in T$ and a bid cost b_λ. The robots send their bids and receive all bids from other robots in parallel (Line 7). The winner determination stage (Lines 8-12) consists of each robot choosing the task with the lowest bid from the set of submitted bids. Ties are broken in an arbitrary manner. In line 9 we introduce a auxiliary function bid(r', t) defined as bid$(r', t) = b_\lambda$ such that $(r', t, b_\lambda) \in B$. The robot with the winning bid has the winning task assigned to it. All robots then remove the winning task from the

Table 2.10: SSI Auction Example for Exploration Task 3 (winning bids and assignments in bold).

Round	r_1 Bids Calculated	r_1 Bid Submitted	r_2 Bids Calculated	r_2 Bid Submitted
1	$\beta_{r_1}^{t_1} \leftarrow \langle r_1, t_1, 4\rangle$ $\beta_{r_1}^{t_2} \leftarrow \langle r_1, t_2, 3\rangle$ $\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_3, 6\rangle$ $\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 5\rangle$	$\boldsymbol{\beta_{r_1}^{t_2}}$	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 6\rangle$ $\beta_{r_2}^{t_2} \leftarrow \langle r_2, t_2, 5\rangle$ $\beta_{r_2}^{t_3} \leftarrow \langle r_2, t_3, 4\rangle$ $\beta_{r_2}^{t_4} \leftarrow \langle r_2, t_4, 3\rangle$	$\beta_{r_2}^{t_4}$
2	$\beta_{r_1}^{t_1} \leftarrow \langle r_1, t_1, 7\rangle$ $\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_3, 9\rangle$ $\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 2\rangle$	$\boldsymbol{\beta_{r_1}^{t_4}}$		$\beta_{r_2}^{t_4}$
3	$\beta_{r_1}^{t_1} \leftarrow \langle r_1, t_1, 9\rangle$ $\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_3, 7\rangle$	$\beta_{r_1}^{t_3}$		$\boldsymbol{\beta_{r_2}^{t_3}}$
4		$\beta_{r_1}^{t_1}$	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 2\rangle$	$\boldsymbol{\beta_{r_2}^{t_1}}$
$\lambda_{r_i}(T_{r_i})$		5		6

set of unassigned tasks and the next bidding round begins.

Lagoudakis *et al.* develop and analyse bidding rules for the MiniMax and MiniSum team objectives in SSI auctions [63]. For the MiniMax team objective each robot bids the cost of completing an additional task t in addition to completing its current allocation:

$$b_\lambda = \lambda_{r_i}(T_{r_i} \cup \{t\})$$

In considering the bounds for this bidding rule, the worst-case upper bound of the performance ratio is $2 * |R|$ away from the optimal. For the MiniSum team objective each robot bids for an additional task using marginal costs:

$$b_\lambda = \lambda_{r_i}(T_{r_i} \cup \{t\}) - \lambda_{r_i}(T_{r_i})$$

The worst-case upper bound of the performance ratio for the MiniSum team objective is a constant factor 2 away from the optimal.

We now demonstrate an SSI auction with the MiniSum team objective on Koenig *et al.* exploration task 3 (Figure 2.3). The bids calculated and submitted in each round are shown in Table 2.10. In the first round each robot has to calculate bids for all tasks. Subsequently the smallest bid submitted by each robot is equal, so via simple tie breaking robot r_1 is awarded its first task t_2. In the second round, robot r_1 needs to recalculate its bids for the remaining tasks taking into account the inter-task synergy of the task it was awarded in the first round. Meanwhile, robot r_2 resubmits its bid for task t_4, and now both robots bid for the same task, with robot r_1 awarded its second task. In the third round, robot r_1 recalculates bids for the remaining tasks taking into consideration the two tasks

Table 2.11: SSI Auction Example for Exploration Task 4.

Round	r_1 Bids Calculated	r_1 Bid Submitted	r_2 Bids Calculated	r_2 Bid Submitted
1	$\beta_{r_1}^{t_1} \leftarrow \langle r_1, t_1, 4\rangle$ $\beta_{r_1}^{t_2} \leftarrow \langle r_1, t_2, 3\rangle$ $\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_3, 8\rangle$ $\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 7\rangle$	$\boldsymbol{\beta_{r_1}^{t_2}}$	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 8\rangle$ $\beta_{r_2}^{t_2} \leftarrow \langle r_2, t_2, 7\rangle$ $\beta_{r_2}^{t_3} \leftarrow \langle r_2, t_3, 4\rangle$ $\beta_{r_2}^{t_4} \leftarrow \langle r_2, t_4, 3\rangle$	$\beta_{r_2}^{t_4}$
2	$\beta_{r_1}^{t_1} \leftarrow \langle r_1, t_1, 7\rangle$ $\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_3, 11\rangle$ $\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 4\rangle$	$\beta_{r_1}^{t_4}$		$\boldsymbol{\beta_{r_2}^{t_4}}$
3		$\boldsymbol{\beta_{r_1}^{t_1}}$	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 11\rangle$ $\beta_{r_2}^{t_3} \leftarrow \langle r_2, t_3, 7\rangle$	$\beta_{r_2}^{t_3}$
4	$\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_1, 4\rangle$	$\boldsymbol{\beta_{r_1}^{t_3}}$		$\beta_{r_2}^{t_3}$
$\lambda_{r_i}(T_{r_i})$		14		3

it already has allocated. Because the task that Robot r_2 was bidding for was allocated in the previous round it now selects one of its cached alternative bids and is subsequently awarded task t_3 during winner determination. Finally, only one task remains, robot r_2 recalculate its bid and is subsequently awarded the final task t_1. The final total cost across both the robots $\sum_{r_i \in R} \lambda_{r_i}(T_{r_i}) = 11$ which is the optimal solution to this problem.

Extensions

A key strength of SSI auctions is their ability to build on inter-task synergies during each task bidding round. However, this strength is also a drawback during early rounds of bidding when few tasks are allocated [138, 58]. A result of this is that robots have a greedy bias towards tasks that are close to their initial locations. This can see two tasks, that in an optimal solution would be allocated to one robot, be split and allocated to two different robots. Take, for example, Koenig *et al.* exploration task 4 (Figure 2.7). In this example SSI auctions generate a sub-optimal solution (Table 2.11), however, we note that the solution costs are still lower than the previous results from parallel and sequential auctions. To try and avoid this and further lower the team cost in SSI auctions a number of improvements and extensions to the bidding and winner determination phases of SSI auctions have been considered, such as, *rough route generation*, *lookaheads*, *rollouts*, *bundle-bids*, and *regret clearing*.

Rough route generation modifies the bidding phase of SSI auctions so that allocations of many additional tasks are considered. Individual robot paths are planned on the assumption of the allocation of these additional tasks. Robots then bid for any single task that they consider to be the most suitable, which is facilitated by bidding for either the closest task or the furthest task in the task set of assumed allocations. Empirical

experiments on the MiniSum team objective have shown this technique to work well in comparison to standard SSI auctions [102].

Lookaheads also modify the bidding phase of SSI auctions and consider the allocation of more than one task at a time to robots. The bidding phase is modified so that each robot calculates bids for up to l tasks. In the case of a two task lookahead, three bids are calculated, a bid each for the two individual cheapest additional tasks, and a bid for the cheapest pair of two additional tasks. During the winner determination stage combinations of bids for a total of l tasks are considered, and the robot that bid for any one task that gives the overall minimal increase in team costs is awarded the additional task. While this approach has been designed to avoid local minima, experimentally it has been shown, on average, to perform worse than standard SSI auctions [138].

In rollouts each bid is calculated with the assumption of robot r_i winning the task t and then all robots complete the remaining SSI auctions for the remaining tasks to determine the total team cost based on this allocation. This process is repeated for each robot's bid on every task and therefore is very time consuming. However, unlike lookaheads, rollouts are guaranteed to perform no worse than standard SSI auctions, this is because bids use the full team costs rather than partial costs. Furthermore, a mix of rollouts in early bidding rounds and standard SSI auctions in later rounds can overcome the problem of few inter-task synergies during early bidding rounds, and has substantially smaller computation requirements than full rollouts in all bidding rounds [138].

SSI auctions with bundles are a hybrid of standard SSI auctions and combinatorial auctions in which each robot bids on combinations of up to l tasks. During the winner determination phase, a total of l tasks are allocated, with each robot allocated $0 - l$ tasks. The performance and computational load on the robots for SSI auctions with bundles depends on the number of tasks l allocated each round. When $l = 1$ SSI auctions with bundles are equivalent to standard SSI auctions and, on the contrary, when $l = |T|$ they are equivalent to combinatorial auctions. As such there is a trade-off required between the number of tasks allocated and the quality of the solution generated. For MRTA problems with capacity constraints, this approach has been shown experimentally on average to produce lower team costs due to each bundle bid taking into account additional synergies between tasks. However, in some scenarios larger team costs than standard SSI auctions are generated as a consequence of the bidding process still only calculating bids based on partial task allocations [57].

Unlike, the previous four SSI auction extensions that make changes to the bidding phase of the auction process, regret clearing only modifies the winner determination phase. In standard SSI auctions the winner of any auction round is the robot who submits the lowest bid for any task, for SSI auctions with regret clearing this is modified such

that the difference between the lowest and second lowest bids for any task is maximised. This requires robots, in every round, to submit bids for all unallocated tasks therefore increasing the communication overhead compared to standard SSI auctions. However, there is no increase in computation costs compared to standard SSI auctions as both algorithms require bids for all tasks to be calculated. Furthermore, the bounds for standard SSI auctions do not apply under winner determination with regret clearing, for the MiniSum team objective the worst-case upper bound has been given as a factor $2 * |T|$ larger than minimal. Empirical evaluation shows that SSI auctions with regret clearing works well for MRTA problems with capacity constraints and for the MiniMax team objective for problems without capacity constraints [58].

Finally, SSI auctions have been extended to more complex MRTA problems. Thomas and Williams consider SSI auctions with heterogeneous tasks [120]. In this work robots can only complete tasks they are qualified for, the bidding and winner determination phases are also altered so that robots bid on tasks their abilities closest match. Ekici, Keskinocak and Koenig adapt SSI auctions for MRTA problems where the reward for completing individual tasks decreases over time [30]. This work also considers a number of different bidding rules adapted for the changed problem. Zheng and Koenig adapt SSI auctions for task coalitions where several robots need to work together at the same time to achieve a task [137]. In this approach the winner determination phase requires knowledge of all robots allocations and costs so that coalitions can be determined. Each of these extensions demonstrate the strength of SSI-like auctions in generating near optimal solutions in minimal time compared to traditional centralised approaches.

2.4.5 Task Clustering and Partitioning

A major computational challenge in the performance of auction mechanisms that consider inter-task synergies is their ability to handle large numbers of robots and tasks. In many auction algorithms increasing the number of tasks causes a combinatorial explosion in the number of calculations required to form task bids. Further compounding this, as the number of robots increases, the communication and computational requirements for winner determination also increase. As a result the suitability of these techniques in large real-world scenarios is limited.

Forming clusters of tasks has been explored by a number of researchers as a method to reduce the combinatorial explosion of increasing task counts. In early work on market-based task allocation Sandholm expanded the CNP by introducing *C-contracts* which replace the CNP's standard one item contract with a contract for a cluster of tasks all of which the contracted robot must complete. Sandholm shows that allocating clusters of tasks to robots can avoid some local minima that single item contracts become stuck

in; although, C-contracts can get stuck in different local minima [95]. In early work on multi-robot auctions, Dias and Stentz develop a clustering algorithm that connects geographically close tasks under the assumption that two tasks that are close have high inter-task synergies, robots then exchange clusters of tasks through an auction method which experimentally is shown to perform better than single task auctions [21].

Many subsequent clustering approaches for MRTA problems use the distance between tasks as a metric for cluster formation. Sariel and Balch discuss the allocation of clusters of tasks to robots where in some situations optimal MRTA solutions can be generated that are unable to be formed using single item auctions, however, in other situations clustering performs worse [102]. Zlot and Stentz use *K*-means clustering to form clusters of geographically close tasks. To determine an ideal cluster, the value of k is incremented from 2 to the total number of tasks with the value of k that generates clusters with the largest relative improvement over the previous value being used [139]. Elango, Nachiappan and Tiwari form clusters of tasks with *K*-means clustering which are then auctioned to robots with a goal of balancing the number of tasks across all robots such that the combined travel and idle costs are minimised [31].

Outside of the domain of multi-robot auctions for MRTA problems, clustering techniques have been applied to task set segmentation. *K*-means clustering [113, 129, 87] and Voronoi Diagrams [128, 130] have been used to form disjoint regions of unexplored terrain which are then assigned to individual robots for exploration. This technique is designed to disperse robots throughout an area with minimum need for close coordination between robots. *K*-means clustering has also been used for map segmentation in the RoboCup Rescue Simulation domain [74, 79]. Voronoi Diagrams have also been used by individual robots to achieve optimal task assignment in systems with no communication between robots, however, this approach is limited to scenarios in which there is only one task assignment per robot [53]. Finally, clustering has been used in a mix of centralised and distributed problem solving, e.g. a centralised graph partitioning algorithm can be used to split a MRTA problem into subproblems which are then solved by the robots contained in each subpartition [111, 67, 68, 55].

2.4.6 Post-Initial Allocation Task Reallocation

A major weakness in non-optimal auction algorithms with one-shot task assignments is their inability to avoid local minima. For instance, consider the bidding rules for the MiniMax and MiniSum team objectives in standard SSI auctions, when no tasks are assigned to an individual robot, the robot will always bid for the task closest to it which in many situations will not be an optimal task assignment (e.g. Figure 2.7). Once they have been assigned this task, all subsequent bids will factor in the inter-task synergies of this task

and as a result solutions that are far from optimal are developed [102].

Although the bidding rules can be modified to avoid this particular nearest-neighbour problem, any auction algorithm with one-shot task assignments and incomplete inter-task synergy information is prone to problems of local minima. Two common approaches for refining task allocations post-initial allocation are task swapping (Section 2.3.2) and repeated auctions which allow robots to reallocate tasks among themselves before or during the execution of tasks. For instance, in Dias *et al.* at fixed time intervals all robots re-auction all uncompleted tasks [25], in Zlot and Stentz robots randomly select complex task sets to repeatedly auction until a stable solution is found [139], in Zlot *et al.* when any individual robot completes a task the robot sequentially offers each of its remaining tasks for auction [140], and finally in Nanjanath and Gini upon each individual task completion robots re-auction all uncompleted tasks [80]. However, when applied to standard SSI auctions, and despite the consideration of few inter-task synergies in early bidding rounds, none of these reallocation approaches will change the task allocation, due to the greedy like bidding approach.

To refine a task allocation solution post-initial allocation in SSI auctions Nanjanath and Gini switch the bidding rule from the MiniSum team objective to a MiniMax-like approach [81]. Adapting their previous repeated auction approach [80] an initial allocation of tasks is made using a standard SSI auction. Then on each task completion, all uncompleted tasks are auctioned under the requirement of minimising execution time. Robots only exchange tasks if it improves the overall team objective and once a task has been exchanged, the robots involved in the exchange replan their paths to travel.

An alternative approach to improving the solution to SSI auction solution with capacity constraints is with K-swaps [136]. K-swaps generalise Sandholm's previous work on contracts for task exchanges [95] and allow two or more robots to exchange differing numbers of tasks to reduce the overall team cost. K-Swaps has been experimentally shown to make significant improvements to task allocations, however, the algorithm trades off larger improvements with pronouncedly increased computational time. Furthermore, task swaps and transfers can also be considered with auctions that incorporate simulated annealing to avoid local minima [133, 135, 134]. Such approach randomly selects tasks and robots for task exchanges and calculates probabilities of the task exchange being accepted and over time can generate an optimal solution.

Reallocating tasks and replanning paths may require much computational power and time. While arbitrary reallocation and replanning may not be the most efficient approach, knowing when to reallocate and how much of the system should be reallocated is a challenging problem. Finally, during task execution it is possible that task costs may change as new information about the environment is discovered [122], robots fail, or tasks added

[106]. In these situations even previously optimal solutions may no longer hold and new allocations of all uncompleted tasks should be determined. We discuss these dynamic environments in the following section.

2.5 Dynamic Environments

In the previous sections we have discussed MRTA problems in which all robots and tasks are known a priori and robots do not fail. In this section we discuss extensions to the MRTA problem where, during plan execution, additional tasks are inserted or robots fail requiring reallocation of uncompleted tasks. A key challenge in the reallocation of tasks is ensuring that tasks are completed, for instance, in some situations it is possible for robots to change their currently executing task so often that no task is ever completed [13].

2.5.1 Task Insertion

Despite a large body of work on auction-based algorithms for MRTA problems, few have considered the effects of dynamically appearing (online) tasks in the problem domain. While it can be argued that algorithms that continually change task allocations *could* handle dynamically inserted tasks, this has little experimental grounding. An important consideration in the handling of online tasks is deciding how much of the existing task allocation to modify. This can range from local replanning, where an individual robot is assigned an additional task and replans its path for the completion of all tasks, to a global reallocation of all uncompleted tasks. Global reallocation can be described as repeated static allocation, that is, each time the world changes and a task reallocation occurs the algorithm used considers the world as static at that point in time. In highly dynamic environments this is likely to be very time consuming and generally not feasible for real-time settings [7]. In contrast, local replanning is very sensitive to the quality of the initial solution and is limited in the range of improvement possible [5].

Previous work by Schoenig and Pagnucco has considered SSI auctions with dynamically inserted tasks and compared the costs of robots bidding only for the new task versus a full new auction of all uncompleted tasks [106]. Their results show, despite a large trade-off in computational time, a global reallocation of tasks produces lower team costs than local replanning. Zlot *et al.* consider MRTA problems in an exploration domain in which a robot generates additional tasks for allocation after each task completion. These tasks are sequentially offered for auction to other robots, however, if no buyer is found the generating robot retains the task [140]. Viguria, Maxa and Ollero's approach of repeatedly auctioning subsets of uncompleted tasks (detailed in Section 2.4.2) allows it to handle dynamically inserted tasks [126]. This approach sits between local replanning and global

reallocation in that robots only offer for auction tasks that they specifically consider to be of high cost. Additionally, these approaches avoid the problem of never completing any tasks through ensuring that the currently executing task is never offered for reallocation.

For MRTA problems without inter-task synergies, Jones *et al.* consider the formation of robot teams to complete tightly-coordinated online tasks [52]. Outside of the domain of market-based systems, popular approaches for handling online tasks include anytime algorithms [70], coalition formations [88], incremental task selection [104], and emergent cooperation [66].

2.5.2 Robot Failure

During task execution individual robots may suffer from a variety of malfunctions and failures, such as, communication drop-outs, partial robot malfunction, and robot death [23]. While detecting and responding to many of these forms of failure is complex and outside the scope of this thesis, we take interest in approaches to the reallocation of tasks upon full robot failure.

Botelho and Alami [10] consider the problem of robot failure in the CNP. In this work, when a robot is about to fail it sends out an emergency distress message to all other robots and one robot will come to its aid and complete the failed robot's task. However, in this work no inter-task synergies are explored as each additional task is allocated only after the completion of a previous task. In considering teams with inter-task synergies, Farinelli, Scerri, and Tambe's approach is for each robot to periodically re-evaluate its ability to complete all assigned tasks and if a robot can no longer complete all of its tasks it attempts to offload these to other robots [35].

An alternative to self awareness of imminent failure is external status monitoring. In this approach robots are required to listen for each other's robotic heartbeats or for broadcast messages on the status of tasks. The robot is assumed to be dead if no heartbeat is detected or status message is received after a set period of time. In Sariel, Balch and Stack when a robot failure is detected all uncompleted tasks in the complete systems are reauctioned across all operating robots [101]. Gerkey and Matarić handle robot failures through repeated auctioning all uncompleted tasks at fixed time intervals [43]. A less computational expensive approach is taken by Dias *et al.*, where only the tasks allocated to a failed robot are offered for auction [23]. Although, at a later moment in time a global reallocation of all tasks occurs.

Sariel, Balch and Erdogan also follow this broadcast status message or assume death approach in the formation of coalitions [103]. In this approach a coalition of one or more robots is required to complete a task and during execution it is possible for a robot to fail. To facilitate this each robot individually keeps track of the assumed state of all tasks and

robots in the environment. If the number of robots remaining in a coalition after failure is below the number of robots required to complete the task the coalition is disbanded and a new coalition formed.

2.6 Tasks with Collection and Delivery

A further extension of the MRTA problem is tasks that require collection and delivery. This increased complexity in the problem applies additional forced constraints on the robots and the paths they travel. These constraints include capacity constraints in the number of items robots can carry at any moment in time, which in turn, heavily influence the minimisation of time or energy used in completing all tasks. Little work has focused on distributed auctions for MRTA problems with collection and delivery. Again while it can be argued that many existing techniques for single point locations should continue to work for tasks with collection and delivery this has almost no experimental grounding. Despite this, a large body of work exists in the field of transport logistics.

2.6.1 Auctions for Transport Scheduling

Fischer, Müller and Pischel apply the CNP to transportation scheduling with fixed time windows [36]. In this work trucks bid for tasks from a central controller and can also make one-for-one swaps with other trucks before they begin to execute their plans. During the execution of plans, the trucks may face traffic delays and as such they can locally replan their routes or auction their uncompleted tasks. Their results show that global reallocation of uncompleted tasks provides a large reduction in distance travelled.

Kohout and Erol argue that Fischer, Müller and Pischel's generation of an initial allocation is poor and therefore global reallocation will produce much better results than local replanning [59]. In their analysis they study problems where multiple items can be transported together and additional jobs are announced sequentially. When a new job is announced each vehicle bids for the job according to the cost of completing the additional job relative to their existing commitments. To avoid problems where inserting additional tasks has large impacts on the completion time of other tasks, upon each task insertion, already scheduled tasks are permitted to be reallocated to other vehicles. In their empirical analysis they compare this approach to Solomon's insertion heuristic which is a popular operations research based approach [114]. Overall they show that their distributed approach is statistically equivalent to the centralised operations research approach.

In a similar vein, Mes, van der Heijden and van Harten compare distributed auctions in MAS to hierarchical operations research approaches in dynamic environments [72]. In this work tasks arrive sequentially and trucks can only carry one task at a time. Each

truck bid calculation for a task considers the time required to do the job and any waiting time between jobs before and after. During execution trucks can also swap future task commitments between each other to improve the overall solution. In the comparison to the operations research approaches the distributed auction approach performs substantially better in highly dynamic environments.

2.6.2 Task Decomposition

In large scale distribution environments, such as troop transportation, a single transportation task may be too large to be completed by one agent in one trip. In these situations, tasks may be able to be broken down into multiple smaller tasks which either require repeated trips by the same agent or individual trips being allocated to multiple agents. Allard and Shekh apply an hierarchical approach that groups agents with related capabilities to solve this particular problem [1]. In their approach a single agent bids for tasks on behalf of each set of grouped agents. At a later point, the individual agents from within each group to complete each task is determined. This approach allows for a reduction in the number of bid calculations and messages exchanged between agents and is experimentally shown to lead to a substantial reduction in computational time required to find a solution to this problem.

2.6.3 Clustering

Song and Regan consider the construction of bids for freight transportation in combinatorial auctions where only one shipment can be carried per truck at any moment in time [115]. This work is solely focused on the actions of a single agent in its private calculations of bids and the effects of combining various individual tasks into bundles for bids including situations where the agent has prior commitments. Despite the different auction mechanism considered, this work is highly relevant to our work as one can consider bundles to be synonymous with clusters and our robots also face situations where they also have prior commitments.

Outside the domain of distributed auctions, vehicle routing with collection and delivery has been studied in a variety of MAS architectures [18]. Additionally, in the area of discrete optimisation vehicle routing is a common problem. Surveys of different approaches to solving this problem class are presented in static [6, 84] and dynamic environments [7, 86]. Specifically related to our ideas, both Ganesh and Narendran [41] and Sáez, Cortés, and Núñez [93] suggest approaches that use task clustering and genetic algorithms to distribute tasks.

2.7 Summary

In this chapter we have explored existing work that contributes to the development of our auction approach for solving MRTA problems. In the following chapter the strengths and weakness of the variety of auction techniques examined in Section 2.4 influence the development of SSC auctions. The brief discussion of existing task clustering approaches in Section 2.4.5 is further expanded in the discussion section of Chapter 4. Finally, the SSC auction algorithm developed in the following chapter is applied to dynamic environments and tasks with collection and delivery (Sections 2.5 and 2.6) in Chapter 5.

Chapter 3

Sequential Single-Cluster Auctions

In this chapter we build on existing approaches of robots sequentially bidding on single tasks in a task set through the construction of and bidding for clusters of tasks. In our approach, one robot is allocated all tasks in a cluster and the rest none. We show empirically that this method, on average, results in lower team costs than standard SSI auctions with capacity constraints and performs much faster than SSI auctions with bundles. After we have established the foundations of this new auction method we extend it to include repeated auctioning of uncompleted tasks. Previous work on repeated auctioning of uncompleted single tasks by Nanjanath and Gini [81] has shown the robustness of robotic teams in reallocating tasks when unexpected delays occur while completing tasks. Our idea is, that upon completion of a single task, all robots create clusters of their uncompleted tasks and auction these task clusters with the goal of improving the minimisation of the overall team objective. This approach allows inter-task synergies to be considered which in auction methods that only allocate tasks once may not be explored. We begin the chapter by describing our algorithm for auctioning clusters of tasks (Section 3.1), we expand this algorithm for repeated auctions (Section 3.2), and finally we summarise the key results of this research (Section 3.3).

3.1 Sequential Auctions with Clusters

We now develop an extension to SSI auctions in which individual tasks are organised into clusters taking into account positive synergies between tasks. Robots bid on these clusters to solve the task allocation problem. We call this *sequential single-cluster auctions* (SSC auctions). SSC auctions assign fixed clusters of tasks to robots over multiple bidding rounds. At the conclusion of each bidding round one previously unassigned task cluster

Portions of the research in this chapter have been published in *Sequential Single-Cluster Auctions for Robot Task Allocation* (AI-11) and *Repeated Sequential Auctions with Dynamic Task Clusters* (AAAI-12).

$C = \{t_1, \ldots, t_o\}$ is assigned to the robot that bids the least for it so that the overall team cost increases the least.

An SSC auction consists of three phases: clustering phase, bidding phase, and winner determination phase. We begin with a set of unassigned tasks for completion. Before the auction, a clustering algorithm is used to allocate all individual tasks into clusters (clustering phase). Each cluster is formed with the goal of maximising the positive synergy between tasks in each cluster. Each task is assigned to one, and only one cluster. Clusters can be of varying sizes. During each auction round, all robots bid on all unassigned task clusters (bidding phase), the auctioneer then determines a winning bid and awards this cluster to the successful robot (winner determination phase). The winning robot then commits to completing all tasks contained in that cluster. Robots do not have to do all tasks in a cluster sequentially. When a robot is awarded a new cluster, the robot adds the tasks in this new cluster to its existing task assignment and replans its path to travel. Once all tasks are allocated, each robot executes its planned path to complete all assigned tasks.

We now describe in detail the specifics of each phase and the algorithm as a whole.

3.1.1 Clustering Phase

Expanding the definition of the MRTA problem (Section 2.2.1) we introduce a set of k clusters $K = \{C_1, \ldots, C_k\}$. We now need to allocate all tasks to one and only one cluster. This is achieved by taking any tuple $\langle T_{C_1}, \ldots, T_{C_k} \rangle$ of pairwise disjoint bundles of tasks, such that:

$$T_{C_j} \subseteq T \text{ with } T_{C_j} \cap T_{C_{j'}} = \emptyset, j \neq j', \forall j, j' = 1, \ldots, k$$

which satisfies:

$$\cup_{C_j \in K} T_{C_j} = T$$

Once we have organised all tasks into clusters, we must ensure that each cluster is allocated to one and only one robot. We do this by taking any tuple $\langle K_{r_1}, \ldots, K_{r_m} \rangle$ of pairwise disjoint bundles of task clusters, such that:

$$K_{r_i} \subseteq K \text{ with } K_{r_i} \cap K_{r_{i'}} = \emptyset, i \neq i', \forall i, i' = 1, \ldots, m$$

which satisfies:

$$\cup_{r_i \in R} K_{r_i} = K$$

As a result of this we have now allocated all tasks into clusters, and assigned all clusters to robots and therefore it holds that we still have a valid solution to the task allocation problem of all tasks being allocated such that each task is allocated to one and only one robot.

3.1.2 Bidding Phase

Now we consider a single round of an SSC auction. We assume that robot r_i has already been assigned the set of task clusters $K_{r_i} \subseteq K$ in previous rounds for all $r_i \in R$. Therefore, the set of unassigned task clusters U is defined by:

$$U \leftarrow K \setminus \cup_{r_i \in R} K_{r_i}$$

Each bid is a triple $\beta = \langle r_i, C_j, b_\lambda \rangle$ consisting of a robot r_i, a task cluster C_j and a bid cost b_λ. The bid cost calculation for each cluster depends on the team objective. For the MiniMax team objective:

$$b_\lambda = \lambda_{r_i}(K_{r_i} \cup C_j)$$

That is, the robot bids the costs to do all tasks assigned to it plus the tasks in the cluster it is bidding on. For the MiniSum team objective:

$$b_\lambda = \lambda_{r_i}(K_{r_i} \cup \{C_j\}) - \lambda_{r_i}(K_{r_i})$$

That is, the robot bids the increase in its costs for doing all of its currently allocated tasks plus the tasks in the cluster it is bidding on.

Every robot has to calculate individual bids for all unassigned task clusters $C' \in U$. Each set of individual robot bids $I_{r_i} = \{\beta_1, \ldots, \beta_{|U|}\}$ satisfies that $\forall C' \in U$ there exists exactly one bid $\beta \in I$ with $C_j = C'$.

Each robot submits its smallest bid for any single unassigned task cluster $\min_{\beta \in I_{r_i}} b_\lambda$ to all other robots. The set of submitted bids $B = \{\beta_1, \ldots, \beta_m\}$ satisfies:

1. $\forall \beta_i \in B$, it holds that $r_i \in R$ and $C_j \in U$; and

2. $\forall r' \in R$ there exists exactly one bid $\beta_i \in B$ with $r_i = r'$.

3.1.3 Winner Determination Phase

Once all bids have been received, the auctioneer evaluates a potentially winning bid $\beta_i \in B$ according to the value b_λ. The winning bid for both the MiniMax and MiniSum team objective is the bid β_i with the smallest b_λ. The auctioneer then assigns all tasks in the cluster C_j to the robot r_i.

3.1.4 SSC Auction Algorithm

The algorithm for SSC auctions is nearly identical to the algorithm previously given for SSI auctions (Section 2.4.4) and runs under the same assumptions of error-free communication and accurate localisation. The key difference between the algorithms is the bid calculation for and allocation of clusters of tasks rather than individual tasks. The algorithm that each robot runs independently is given in Figure 3.1. The SSC auction begins and continues while there are unassigned task clusters (Line 1). During the bidding stage (Lines 2-7) the robot calculates bids for every unassigned task cluster and submits its single lowest bid for any one cluster to all other robots. The function CalcBid takes the set of previously assigned clusters K_{r_i} to robot r_i and the cluster C being bid on and uses a bidding rule to calculate a bid cost (Line 5). The robots send their bids and receive all bids from other robots in parallel (Line 7). The winner-determination stage (Lines 8-12) consists of each robot choosing the task cluster with the lowest bid from the set of submitted bids. Ties are broken in an arbitrary manner. The robot with the winning bid has the winning task cluster assigned to it. All robots then remove the winning task cluster from the set of unassigned clusters and the next bidding round begins.

function SSC-Auction (U,K_{r_i}, r_i, R)
Input: U: the set of clusters to be assigned
K_{r_i}: the set of clusters presently assigned to robot r_i
r_i: the robot
R: the set of robots
Output: K_{r_i}: the set of clusters assigned to the robot

1: **while** ($U \neq \emptyset$)
2: /* *Bidding Stage* */
3: $\beta_{min} \leftarrow \infty$
4: **for each** cluster $C \in U$
5: $\beta_{r_i}^C \leftarrow$CalcBid(K_{r_i},C);
6: $\beta_{min} \leftarrow$min($\beta_{min}, \beta_{r_i}^C$);
7: Send(β_{min}, R) | $B \leftarrow \bigcup_i$Receive($\beta_{r_i}^C$, R);
8: /* *Winner-Determination Stage* */
9: $(r', C) \leftarrow$ arg min$_{(r' \in R, C \in U)}$ B;
10: **if** $r_i = r'$ **then**
11: $K_{r_i} \leftarrow K_{r_i} \cup C$;
12: $U \leftarrow U \backslash C$;

Figure 3.1: Algorithm for Sequential Single-Cluster Auctions.

3.1.5 Performance Comparison

We now compare the relative computation, communication, and solutions of SSC auctions to standard SSI auctions. These unique performance differences allow SSC auctions to operate in an efficient manner and generally result in a small team cost.

Theorem 3.1. *The number of rounds in an SSC auction is not greater than the number of rounds in an SSI auction.*

Proof. We define an SSI auction as the tuple $A_{\text{ssi}} = \langle R, T \rangle$. The number of rounds N_{ssi} in the auction A_{ssi} is equal to the number of tasks $N_{\text{ssi}} = |T|$ as only one task is allocated per round. We define an SSC auction as the tuple $A_{\text{ssc}} = \langle R, T, C \rangle$. The number of rounds N_{ssi} in the auction A_{ssc} is equal to the number of clusters $N_{\text{ssc}} = |C|$ as one cluster is allocated per round. Each cluster can have one or more tasks, therefore:

$$|C| \leq |T|$$

and consequently:

$$N_{\text{ssc}} \leq N_{\text{ssi}}$$

□

Theorem 3.2. *The communication cost in an SSC auction is at worst equal to an SSI auction.*

Proof. For a distributed SSI auction all robots are aware of all tasks before the auction. In each round of the auction every robot has to communicate its bid to every other robot so there are $|R|(|R| - 1) \approx |R|^2$ messages per round. From above there are N_{ssi} rounds in an SSI auction. Therefore in total there are in the order of $N_{\text{ssi}}|R|^2$ messages sent in a complete auction.
SSC auctions follow the same bidding and communication rules as SSI auctions except that robots bid on clusters rather than tasks. All robots are aware of all tasks and clusters before the auction. From Theorem 3.1 there are N_{ssc} rounds in an SSC auction. Therefore it holds that in total there are in the order of $N_{\text{ssc}}|R|^2$ messages sent in a complete auction. We again note that:

$$N_{\text{ssc}} \leq N_{\text{ssi}}$$

and therefore:

$$N_{\text{ssc}}|R|^2 \leq N_{\text{ssi}}|R|^2$$

□

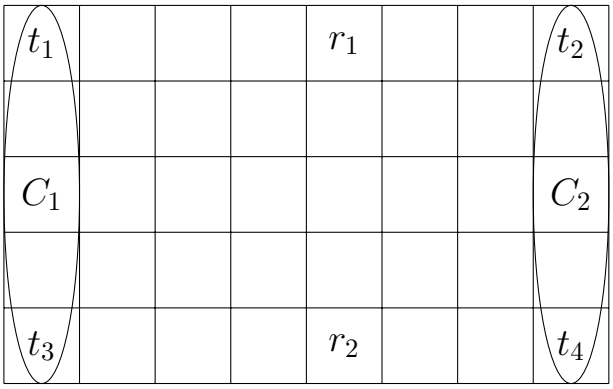


Figure 3.2: Exploration Task 4 with two clusters.

Table 3.1: SSC Auction Example for Exploration Task 4 (winning bids and assignments in bold).

Round	r_1 Bids Calculated	r_1 Bid Submitted	r_2 Bids Calculated	r_2 Bid Submitted
1	$\beta_{r_1}^{C_1} \leftarrow \langle r_1, C_1, 8\rangle$ $\beta_{r_1}^{C_2} \leftarrow \langle r_1, C_2, 7\rangle$	$\boldsymbol{\beta_{r_1}^{C_2}}$	$\beta_{r_2}^{C_1} \leftarrow \langle r_2, C_1, 8\rangle$ $\beta_{r_2}^{C_2} \leftarrow \langle r_2, C_2, 7\rangle$	$\beta_{r_2}^{C_2}$
2	$\beta_{r_1}^{C_1} \leftarrow \langle r_1, C_1, 11\rangle$	$\beta_{r_1}^{C_1}$		$\boldsymbol{\beta_{r_2}^{C_1}}$
$\lambda_{r_i}(T_{r_i})$		7		8

Theorem 3.3. *Winner determination time in an SSC auction is equal to winner determination time in an SSI auction.*

Proof. In an SSI auction each bid β_{ssi} consists of a robot r_i, a task t, and a cost b_λ. In an SSC auction the structure of a bid remains the same, with the exception that t is replaced by C_j (as defined in Section 3.1.2). For winner determination, we have a set of bids B and the value of each b_λ is compared in the same manner in both auction frameworks and the number of bids $|B|$ does not change. Therefore the winner determination time does not change. □

Additionally, SSC winner determination time is much faster than SSI with bundles. This is because in SSI with bundles each bid must include b_λ for each combination of the l tasks that is being bid on. To determine the winner in SSI with bundles each b_λ for each combination needs to be compared to all other bids and combinations.

Finally, when clusters enforce positive synergies between tasks the resultant team cost in an SSC auction is less than in an SSI auction. In the previous chapter, we analysed Exploration Task 4 (Koenig *et al.* [56]), a simple example MRTA problem in which SSI auctions fail to generate an optimal solution (Section 2.4.4). Now let's consider the same problem with an SSC auction with two clusters (Figure 3.2). First, we define two clusters: $C_1 = \{t_1, t_3\}$ and $C_2 = \{t_2, t_4\}$. Table 3.1 shows the bids for each cluster in each auction

round of an SSC auction for this example. During the first round of bidding both robots submits bids for cluster C_2 and by simple tie breaking this is awarded to robot r_1. During the second round of bidding robot r_1 must calculate new bids for the remaining cluster C_1 taking into consideration the tasks t_2 and t_4 that it won during the first round. Meanwhile, robot r_2 submits its previous cached alternative bid for cluster C_1, and subsequently wins the auction with the lowest bid. The overall total distance sum is $\sum_{r_i \in R} \lambda_{r_i}(T_{r_i}) = 15$ which is the optimal solution to this problem. However, we should note that if a cluster fails to employ synergies correctly, e.g. if $C_1 = \{t_1, t_4\}$ and $C_2 = \{t_2, t_3\}$, SSC auctions may result in team costs that are worse than SSI auctions.

3.1.6 Solution Bounds

We now consider the worst-case solution bounds for SSC auctions. As demonstrated in the two theorems below the worst-case solution bounds for SSC auctions depends heavily on the number of clusters k.

Theorem 3.4. *When the number of clusters k is equal to the number of tasks $|T|$ the solution bounds are identical to SSI auctions.*

Proof. In the situation of $k = |T|$ each cluster will contain exactly one task. The SSC auction therefore becomes identical to an SSI auction. Therefore, the solution bounds for SSI auctions hold. □

Theorem 3.5. *When the number of clusters $k = 1$ the worst-case solution may be arbitrarily large.*

Proof. When $k = 1$ all tasks will be in only one cluster, there will only be one auction round and only one robot will be assigned all tasks. If the distance between any two tasks is arbitrarily large (e.g. $\lambda(t_1, t_2)$ in Fig 3.3), the overall solution cost will be arbitrarily large. □

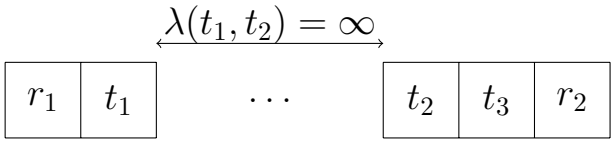


Figure 3.3: Example of cluster formation with high inter-task costs.

Furthermore for any value of $k < |T|$ the worst-case solution may be arbitrarily bad if the clustering algorithm fails to correctly consider positive inter-task synergies [102]. Consider the formation of two clusters for the three tasks in the example in Fig 3.3. A good set of clusters would be $C_1 = \{t_1\}$ and $C_2 = \{t_2, t_3\}$ as this leads to the optimal solution. However, if the clustering algorithm formed the clusters $C_1 = \{t_1, t_2\}$ and $C_2 = \{t_3\}$, the solution would be arbitrarily bad. Additionally, it is important to stress that actual bounds on the distances between tasks during cluster formation is dependent on the specifics of the algorithm used for clustering. This is explored in some further detail in the following chapter.

In these two theorems we have considered the bounds at the two extreme values of k at which either the benefits of clustering previously unexplored inter-task synergies (large k) or robot-task synergies through auctions (small k) are neglected. Although these bounds suggest that any value of k smaller than the number of tasks may result in worse solutions to MRTA problems than SSI auctions, we stress that these are worst-case bounds and other SSI-like auction extensions also have bounds that are worse than standard SSI auctions but experimentally perform better [138, 57, 58].

3.1.7 Experiment Setup

To experimentally evaluate SSC auctions we use the Java Agent Development Framework (JADE) [4] to model autonomous robots and interface with the Player/Stage Project [42] to simulate a near realistic robotic environment. Using this setup we model an office-like environment with 16 rooms each containing four interconnecting doors that can be independently opened or closed to allow or restrict travel between rooms (Figure 3.4). This environment has become the standard testbed in recent literature [122, 138, 57, 58, 136, 81, 137]. Each robot is supplied with a map of the environment at a resolution of 510x510 grid units. Each grid unit is representative of a 5cm by 5cm area of space and gives an overall simulated space of 25.5m by 25.5m.

In each experiment, the doors between different rooms and the hallway are either open or closed. We test on 25 randomly generated configurations of opened and closed doors with each robot starting in a different random location. Robots can only travel between rooms through open doors and they cannot open or close doors. However, it is guaranteed there is at least one path between each room and every other room. These constraints reduce the number of possible configurations from 2^{40} to a maximum of 2^{26} permutations, which additionally allow us to focus on the quality of the bidding algorithm in a consistent environment. Without the constraints it is possible for configurations to be produced with areas that robots are unable to traverse, for instance, tasks inside completely closed rooms. Furthermore, if robots were able to open and close doors without penalty this

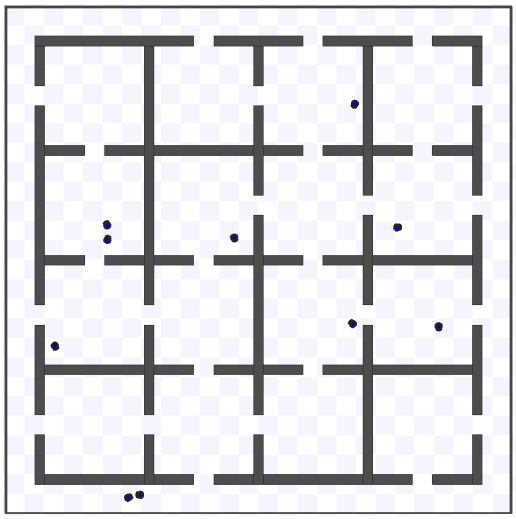


Figure 3.4: A simulation of robots operating in an office-like environment (cf. [122]).

would simplify the world to only one configuration, and with penalties could produce additional unnecessary complexities in solving the problem.

For bidding and path planning, robots use the cheapest-insertion and two-opt [16] heuristics to quickly calculate costs between tasks. Furthermore, in each experiment robots are set a fixed task capacity constraint of the ratio of the number of tasks to the number of robots ($|T_{r_i}| \leq \frac{|T|}{|R|}$). The purpose of this capacity constraint is to ensure robots are allocated a fair balance of tasks. Robots stop being allocated additional tasks once these capacities are met. For each configuration we test with $|R| \in \{2, 4, 6, 8, 10\}$ and $|T| \in \{6, \ldots, 60\}$.

We use standard *K*-means clustering [69] to quickly create clusters of geographically close tasks, which generally have high inter-task synergies, to be auctioned. It is important to note that *K*-means clustering does not take into account walls and closed doors. This means that it is possible for tasks to be clustered together that may have a large navigational distance between them (low inter-task synergy). However, this approach provides a good representation of a real-world environment where it would be extremely complex to always create an optimal or ideal grouping of tasks (this is explored more in the following chapter). For our experiments we test two different total numbers of task clusters. Our first experiment uses a cluster count of half the number of tasks $k = \frac{1}{2}|T|$, giving an average of two tasks per cluster. Our second experiment uses a cluster count of two-thirds the number of tasks $k = \frac{2}{3}|T|$. This gives an average of 1.5 tasks per cluster providing a balance between isolated single tasks and groups of close tasks.

To compare the effectiveness of SSC auctions we also run parallel, SSI, and SSI with bundles auctions on the same 25 configurations. For SSI with bundles we test $l_{\text{bundles}} = 2$ and $l_{\text{bundles}} = 3$ with a *non-cautious auctioneer*, that is, all l_{bundles} tasks are allocated in

each round. Furthermore, we test *hard* and *soft* capacity constraints for SSI with bundles. Hard capacity constraints ensure that all robots are allocated exactly their capacity of tasks. Soft capacity constraints allow robots to go slightly over their capacity, provided they are under their capacity before the round winner determination and allocation. This comparison of capacity constraints is necessary because SSC auctions can only operate with soft capacity constraints as all tasks in a cluster must be allocated to a single robot and there is no guarantee that the number of tasks in each cluster will be evenly balanced.

3.1.8 Results

MiniMax

We begin our analysis of the MiniMax team objective by considering the mean results presented in Table 3.2. We observe that SSC auctions produce a lower mean MiniMax result than SSI auctions in all robot/task combinations when the number of robots is greater than two. Overall there is a mean maximum distance reduction of 12.3% when the number of clusters $k = \frac{1}{2}|T|$ and a reduction of 11.4% when the number of clusters $k = \frac{2}{3}|T|$. If we exclude the experiments with only two robots, the mean improvement increases to 15.0% for $k = \frac{1}{2}|T|$ and 13.6% for $k = \frac{2}{3}|T|$. Despite $k = \frac{1}{2}|T|$ having an overall larger mean improvement, we note that, $k = \frac{1}{2}|T|$ does not result in lower mean MiniMax distances than $k = \frac{2}{3}|T|$ in all robot/task combinations.

In considering the results for SSI auctions with bundles we observe an overall mean improvement against standard SSI auctions for both bundle sizes and capacity constraints tested. For bundles with hard capacity constraints there was 1.4% improvement for $l_{\text{bundles}} = 2$ and 0.6% for $l_{\text{bundles}} = 3$. When we exclude the experiments with only two robots $l_{\text{bundles}} = 2$ improves further to 2.1%, however, $l_{\text{bundles}} = 3$ marginally decreases to 0.5%. If we relax the capacity constraints to soft limits the improvement for $k = 2$ increases to 7.7% and 10.5% for $l_{\text{bundles}} = 3$. Again excluding the two robot experiments $l_{\text{bundles}} = 2$ further increases to 9.2% and again $l_{\text{bundles}} = 3$ decreases to 9.9%. Interestingly auctions where $l_{\text{bundles}} = 3$ which consider more robot/task combinations do not always result in lower results than auctions where $l_{\text{bundles}} = 2$. This result, however, is consistent with Koenig *et al.* prior results for SSI auctions with bundles where a non-cautious auctioneer has been used [57].

We now compare the relative improvements made by both SSC and SSI with bundle auctions over standard SSI auctions. In considering overall means, one can conclude that SSC has a small advantage over SSI with bundles and soft capacities. Meanwhile, SSC with bundles and hard capacities does not offer much improvement at all over standard SSI auctions. This result is not entirely surprising as the tighter capacity constraints mean

less inter-task synergy is permitted.

To further compare the differences in the behaviour of SSC and SSI with bundles we plot the relative improvement against standard SSI auctions for each controlled dimension (Figures 3.5, 3.6, and 3.7). In considering the effect of the number of robots on the relative improvement (Figure 3.5) SSC auctions show an increasing improvement as the number of robots increases, whereas, SSI with bundles do not show any trend regardless of capacity constraint type. SSC auctions also show an increasing improvement when compared to the number of tasks (Figure 3.6), again SSI with bundles show no clear trend. Finally, we plot each auction type against the size of each robot's capacity (Figure 3.7). In these plots none of the auction types show any trend or effect caused by the capacity constraint on their relative results.

Table 3.2: Mean SSC Auctions Objective Experimental Results for the MiniMax Team Objective (percentage improvement compared to SSI auctions in brackets).

			Standard		SSC		SSI bundles $l_{\text{bundles}} = 2$		SSI bundles $l_{\text{bundles}} = 3$	
Capacity	Robots	Tasks	Parallel	SSI	$k = \frac{1}{2}\|T\|$	$k = \frac{2}{3}\|T\|$	Hard-Cap	Soft-Cap	Hard-Cap	Soft-Cap
3	2	6	1039	1130	1197 (-5.9%)	1078 (4.6%)	1102 (2.4%)	1080 (4.4%)	1030 (8.9%)	966 (14.5%)
3	4	12	1094	1138	951 (16.5%)	1085 (4.6%)	1083 (4.8%)	1080 (5.1%)	1048 (7.9%)	1023 (10.1%)
3	6	18	1060	1155	1049 (9.2%)	972 (15.8%)	1037 (10.2%)	1020 (11.7%)	1022 (11.6%)	936 (19.0%)
3	8	24	1199	1112	1055 (5.2%)	1026 (7.7%)	1032 (7.3%)	1021 (8.2%)	1070 (3.8%)	1054 (5.2%)
3	10	30	1092	1159	976 (15.8%)	962 (17.0%)	1202 (-3.7%)	1103 (4.8%)	1049 (9.5%)	1004 (13.4%)
4	2	8	1360	1357	1296 (4.5%)	1310 (3.5%)	1397 (-2.9%)	1318 (2.9%)	1361 (-0.3%)	1140 (16.0%)
4	4	16	1518	1377	1194 (13.4%)	1261 (8.5%)	1352 (1.8%)	1225 (11.1%)	1509 (-9.5%)	1383 (-0.4%)
4	6	24	1401	1377	1298 (5.7%)	1188 (13.7%)	1258 (8.7%)	1189 (13.7%)	1398 (-1.5%)	1198 (13.1%)
4	8	32	1390	1353	1101 (18.7%)	1110 (18.0%)	1376 (-1.7%)	1218 (9.9%)	1452 (-7.3%)	1206 (10.9%)
4	10	40	1541	1401	1044 (25.5%)	1147 (18.1%)	1397 (0.3%)	1119 (20.1%)	1549 (-10.6%)	1381 (1.4%)
5	2	10	1515	1481	1493 (-0.8%)	1486 (-0.4%)	1534 (-3.6%)	1505 (-1.7%)	1528 (-3.2%)	1375 (7.2%)
5	4	20	1671	1581	1400 (11.4%)	1447 (8.5%)	1459 (7.7%)	1430 (9.6%)	1526 (3.5%)	1409 (10.9%)
5	6	30	1643	1663	1534 (7.7%)	1286 (22.7%)	1554 (6.6%)	1425 (14.3%)	1524 (8.4%)	1363 (18.1%)
5	8	40	1598	1615	1274 (21.1%)	1425 (11.7%)	1460 (9.5%)	1452 (10.1%)	1587 (1.7%)	1575 (2.5%)
5	10	50	1644	1676	1259 (24.8%)	1384 (17.4%)	1604 (4.3%)	1456 (13.1%)	1631 (2.7%)	1291 (22.9%)
6	2	12	1768	1724	1637 (5.0%)	1679 (2.6%)	1744 (-1.2%)	1707 (1.0%)	1723 (0.0%)	1496 (13.2%)
6	4	24	1777	1654	1564 (5.5%)	1541 (6.8%)	1766 (-6.7%)	1614 (2.5%)	1706 (-3.1%)	1556 (5.9%)
6	6	36	1829	1770	1465 (17.2%)	1509 (14.7%)	1827 (-3.3%)	1709 (3.4%)	1765 (0.3%)	1574 (11.1%)
6	8	48	1768	1690	1438 (14.9%)	1462 (13.5%)	1730 (-2.4%)	1602 (5.2%)	1722 (-1.9%)	1546 (8.5%)
6	10	60	1825	1704	1321 (22.5%)	1427 (16.2%)	1791 (-5.1%)	1614 (5.3%)	1741 (-2.2%)	1600 (6.1%)
Combined Mean:			1487	1456	1277 (12.3%)	1289 (11.4%)	1435 (1.4%)	1344 (7.7%)	1447 (0.6%)	1304 (10.5%)
Mean (excluding R=2):			1503	1464	1245 (15.0%)	1265 (13.6%)	1433 (2.1%)	1329 (9.2%)	1456 (0.5%)	1318 (9.9%)

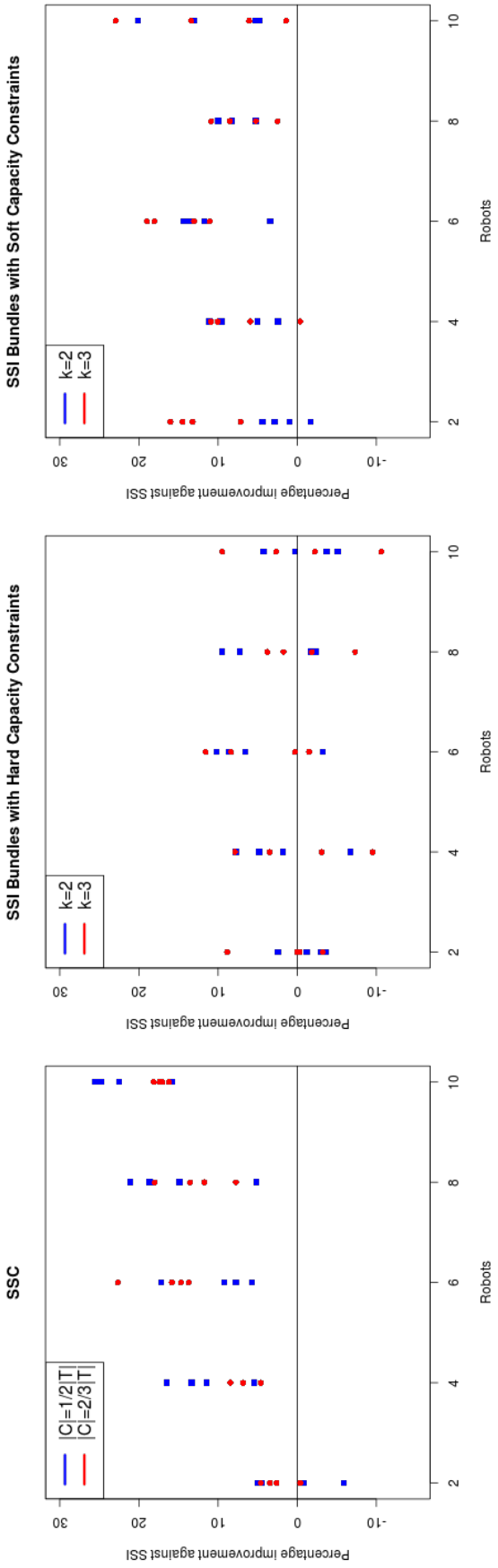


Figure 3.5: Comparison of improvement percentage to the number of robots for SSI and SSC with bundles over SSI auctions for the MiniMax team objective.

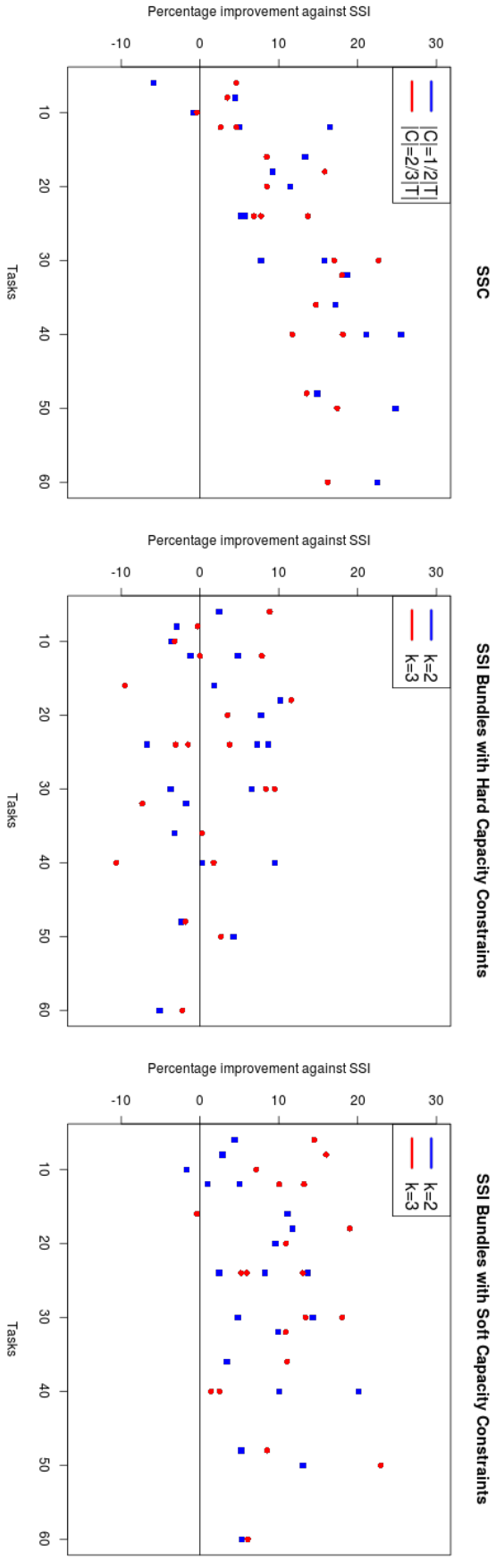


Figure 3.6: Comparison of improvement percentage to the number of tasks for SSI and SSC with bundles over SSI auctions for the MiniMax team objective.

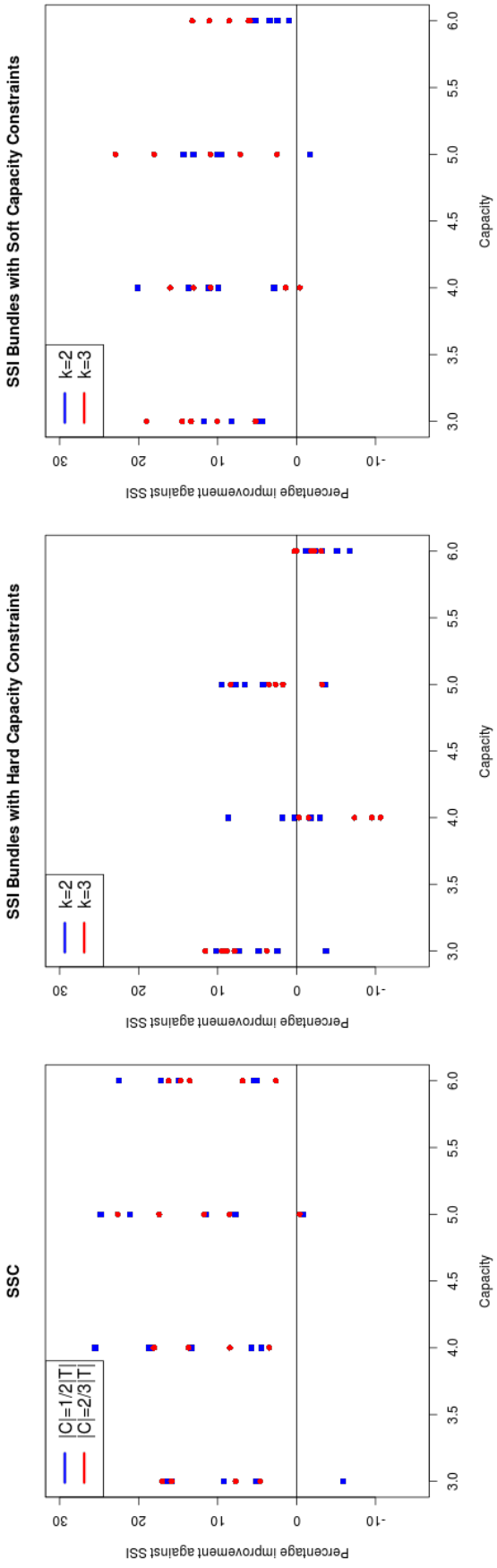


Figure 3.7: Comparison of improvement percentage to the robot capacity for SSI and SSC with bundles over SSI auctions for the MiniMax team objective.

MiniSum

The mean results of the MiniSum team objective is shown in Table 3.3. Overall SSC outperforms SSI auctions in every robot task combination except in the combination $|R| = 2, |T| = 10$. For $k = \frac{1}{2}|T|$ the mean improvement over SSI auctions is 12.4% and, excluding the results for only two robots, the improvement increases to 13.4%. For $k = \frac{2}{3}|T|$ the mean improvement over SSI auctions is 8.2% and again, excluding the results for only two robots, the improvement also increases to 8.8%. It is interesting to note that, unlike the results for the MiniMax team objective, in every robot/task combination where there is more than two robots $k = \frac{1}{2}|T|$ has a larger improvement than $k = \frac{2}{3}|T|$.

The results for SSI with bundles also differ substantially from the corresponding MiniMax team objective results. Considering SSI with bundles where $l_{\text{bundles}} = 2$ both the hard and soft capacity constraints result in overall means that are worse than standard SSI auctions. For SSI with bundles where $l_{\text{bundles}} = 3$ the overall percentage improvement compared to standard SSI exceeded that of the MiniMax team objective results. Furthermore, the mean results for $l_{\text{bundles}} = 3$ with soft capacities sit between the results of the two sets of SSC results.

Figures 3.8, 3.9, and 3.10 plot the comparison between standard SSI, SSC, and SSI with bundles for the three controlled dimensions. Considering the effect the number of robots has on the improvement compared to standard SSI (Figure 3.8), SSC appears to have a small increasing improvement as the number of robots increases, however, this is much less than the results for the MiniMax team objective. Meanwhile, SSI with bundles do not appear to be influenced by the number of robots. The comparison against the number of tasks (Figure 3.9) does not show any clear trend for SSC auctions. SSI with bundles produces results with high variance, however. Despite this it appears that when the hard capacity constraint is applied, the improvement against standard SSI auctions decreases as the number of tasks increases. Finally the comparison against capacity (Figure 3.10) appears to show no trend for SSC auctions and a decreasing trend for SSI auctions with bundles.

Table 3.3: Mean SSC Auctions Experimental Results for the MiniSum Team Objective (percentage improvement compared to SSI auctions in brackets).

			Standard		SSC		SSI bundles $l_{\text{bundles}} = 2$		SSI bundles $l_{\text{bundles}} = 3$	
Capacity	Robots	Tasks	Parallel	SSI	$k = \frac{1}{2}\|T\|$	$k = \frac{2}{3}\|T\|$	Hard-Cap	Soft-Cap	Hard-Cap	Soft-Cap
3	2	6	1653	1819	1623 (10.8%)	1651 (9.2%)	1731 (4.9%)	1677 (7.8%)	1751 (3.8%)	1775 (2.4%)
3	4	12	2797	2867	2404 (16.2%)	2438 (15.0%)	2940 (-2.5%)	2676 (6.7%)	2535 (11.6%)	2444 (14.8%)
3	6	18	3580	3542	2962 (16.4%)	3086 (12.9%)	3384 (4.4%)	3170 (10.5%)	3057 (13.7%)	2885 (18.5%)
3	8	24	4723	4395	3571 (18.7%)	3782 (13.9%)	4095 (6.8%)	3888 (11.5%)	3799 (13.6%)	3527 (19.8%)
3	10	30	5057	4928	3940 (20.0%)	4085 (17.1%)	4708 (4.5%)	4437 (10.0%)	4317 (12.4%)	3938 (20.1%)
4	2	8	2231	2090	2013 (3.7%)	2085 (0.3%)	2296 (-9.9%)	2071 (0.9%)	2207 (-5.6%)	2087 (0.2%)
4	4	16	3887	3522	2934 (16.7%)	3154 (10.5%)	3713 (-5.4%)	3291 (6.5%)	3270 (7.2%)	2970 (15.7%)
4	6	24	4868	4413	3738 (15.3%)	3964 (10.2%)	4600 (-4.2%)	4136 (6.3%)	4326 (2.0%)	3608 (18.3%)
4	8	32	5871	5302	4195 (20.9%)	4608 (13.1%)	5618 (-5.9%)	4841 (8.7%)	5116 (3.5%)	4401 (17.0%)
4	10	40	6901	5865	4605 (21.5%)	4978 (15.1%)	6305 (-7.5%)	5548 (5.4%)	5628 (4.0%)	5038 (14.1%)
5	2	10	2551	2383	2408 (-1.0%)	2395 (-0.5%)	2593 (-8.8%)	2522 (-5.8%)	2519 (-5.7%)	2369 (0.6%)
5	4	20	4303	3840	3425 (10.8%)	3732 (2.8%)	4250 (-10.7%)	4040 (-5.2%)	3671 (4.4%)	3491 (9.1%)
5	6	30	5649	4864	4641 (4.6%)	4658 (4.2%)	5232 (-7.6%)	5123 (-5.3%)	4784 (1.7%)	4250 (12.6%)
5	8	40	6800	5441	5040 (7.4%)	5386 (1.0%)	6188 (-13.7%)	6086 (-11.9%)	5764 (-5.9%)	5313 (2.4%)
5	10	50	7690	6188	5601 (9.5%)	5749 (7.1%)	6981 (-12.8%)	6740 (-8.9%)	6003 (3.0%)	5552 (10.3%)
6	2	12	2964	2832	2694 (4.9%)	2775 (2.0%)	2965 (-4.7%)	2846 (-0.5%)	2908 (-2.7%)	2800 (1.1%)
6	4	24	4858	4180	3940 (5.7%)	4046 (3.2%)	4865 (-16.4%)	4604 (-10.2%)	4362 (-4.4%)	4049 (3.1%)
6	6	36	6423	5589	4974 (11.0%)	5145 (7.9%)	6087 (-8.9%)	5809 (-3.9%)	5540 (0.9%)	5140 (8.0%)
6	8	48	7656	6223	5725 (8.0%)	5988 (3.8%)	7522 (-20.9%)	6916 (-11.1%)	6373 (-2.4%)	5967 (4.1%)
6	10	60	8717	7222	6225 (13.8%)	6646 (8.0%)	8261 (-14.4%)	7668 (-6.2%)	7093 (1.8%)	6339 (12.2%)
Combined Mean:			4959	4375	3833 (12.4%)	4018 (8.2%)	4717 (-7.8%)	4405 (-0.7%)	4251 (2.8%)	3897 (10.9%)
Mean (excluding R=2):			5611	4899	4245 (13.4%)	4465 (8.8%)	5297 (-8.1%)	4936 (-0.8%)	4727 (3.5%)	4306 (12.1%)

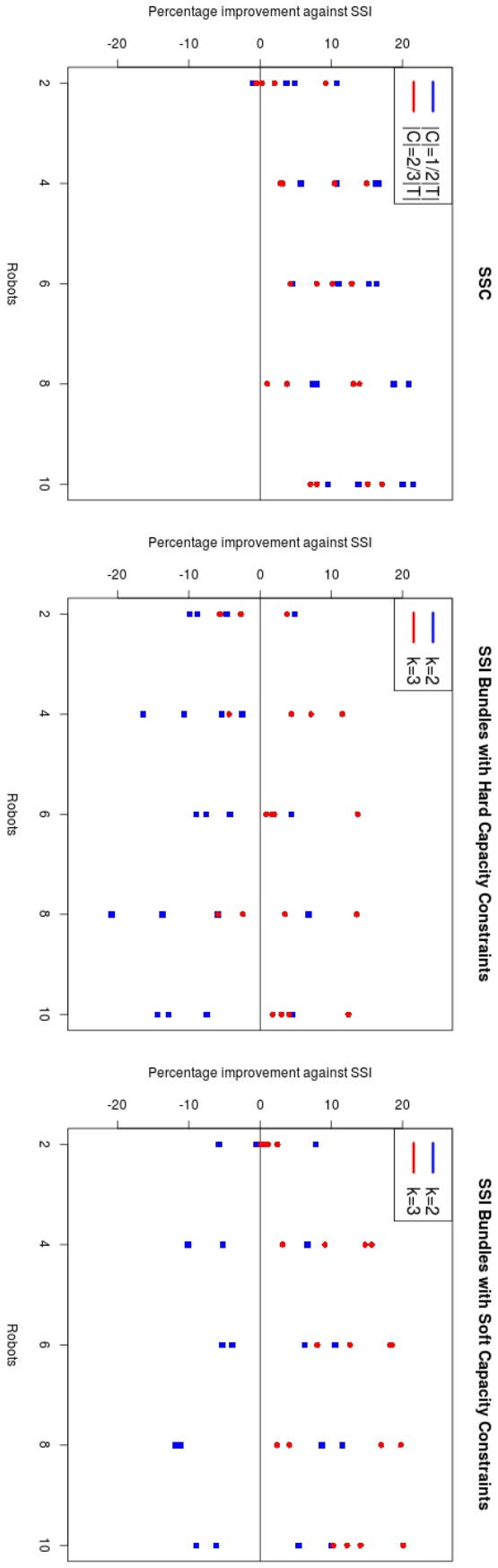


Figure 3.8: Comparison of improvement percentage to the number of robots for SSI and SSC with bundles over SSI auctions for the MiniSum team objective.

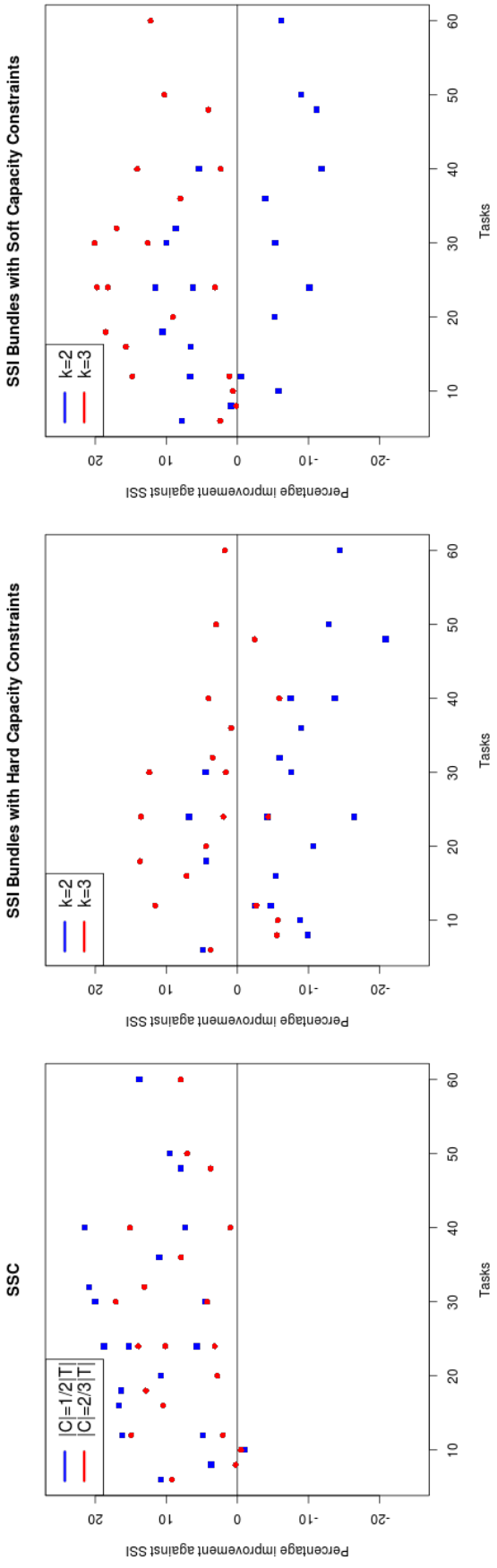


Figure 3.9: Comparison of improvement percentage to the number of tasks for SSI and SSC with bundles over SSI auctions for the MiniSum team objective.

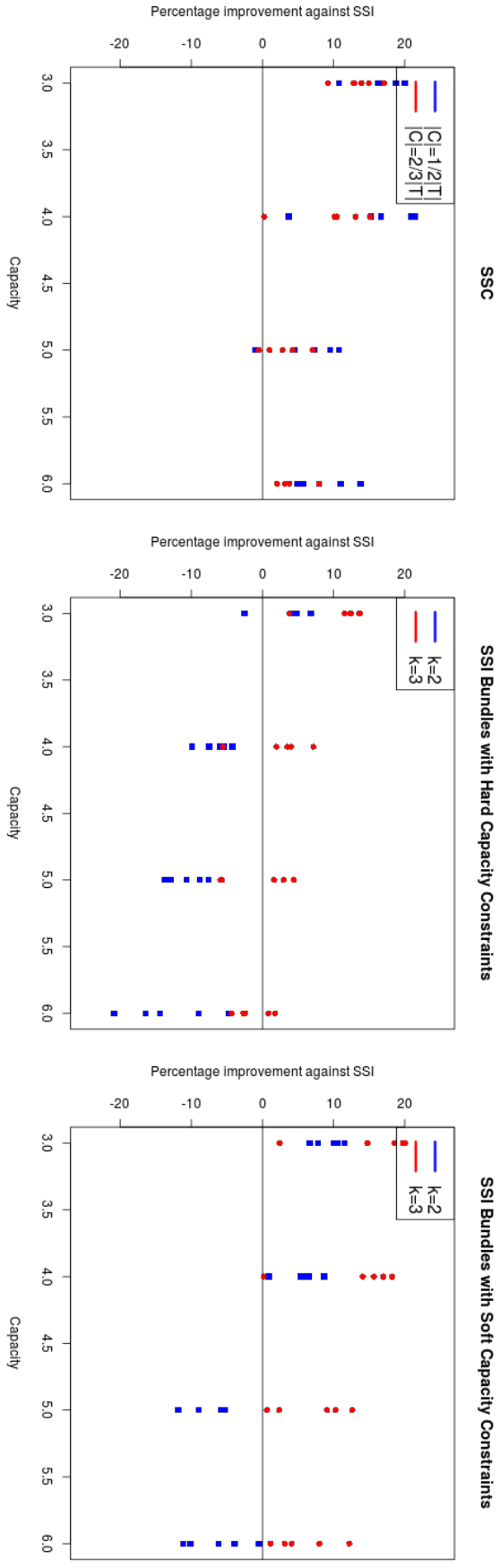


Figure 3.10: Comparison of improvement percentage to the robot capacity for SSI and SSC with bundles over SSI auctions for the MiniSum team objective.

Computational Time

Table 3.4 shows the mean time to run auctions and allocate all tasks for each robot/task combination. These results are from a system with a 2.8GHz Intel Core i7 CPU, 8GB RAM, running Ubuntu 11.04 x64. For all auctions except SSC we begin timing when the robots are informed of the tasks to bid on and stop timing when all tasks have been allocated. For the SSC auctions we begin timing when the clustering algorithm begins and stop when all tasks have been allocated.

Parallel auctions are always the quickest auction to finish, however, they produce the most sub-optimal distance results. Standard SSI auctions are on average around five times slower than parallel auctions. SSC auctions run in near identical time to SSI auctions. This is an important point because SSC auctions need to generate the task clusters before auctions can begin which can take considerable time. However, once the auctioning phases begin they are quicker than SSI auctions because they have fewer auction rounds. This result validates our properties from Section 3.1.5 and analysing both the mean distance results and the timing results empirically demonstrates that SSC auctions can result in a lower team objective distance in a similar time to SSI auctions. Finally, SSI auctions with bundles perform nearly twice as slow as SSI and SSC auctions and almost ten times slower than parallel auctions.

Table 3.4: Mean Total Task Allocation Determination Time *(seconds) for SSC Auctions*

			Standard		SSC		SSI bundles $l_{\text{bundles}} = 2$		SSI bundles $l_{\text{bundles}} = 3$	
Capacity	Robots	Tasks	Parallel	SSI	$k = \frac{1}{2}\|T\|$	$k = \frac{2}{3}\|T\|$	Hard-Cap	Soft-Cap	Hard-Cap	Soft-Cap
3	2	6	0.90	1.37	1.37	1.40	1.44	1.40	1.39	1.41
3	4	12	1.59	3.54	3.56	3.55	3.77	3.86	3.79	3.74
3	6	18	2.13	6.55	6.42	6.44	8.65	8.66	8.61	8.61
3	8	24	2.78	10.83	10.75	10.68	17.66	17.78	17.59	17.65
3	10	30	3.76	16.53	16.22	16.24	33.63	33.33	33.17	33.32
4	2	8	1.18	1.76	1.86	1.80	1.82	1.84	1.86	1.81
4	4	16	1.87	5.28	5.24	5.21	5.60	5.64	5.50	5.57
4	6	24	2.53	10.43	10.26	10.29	14.18	14.27	14.21	14.04
4	8	32	3.42	17.68	17.54	17.36	30.01	29.76	29.76	29.59
4	10	40	4.67	27.36	27.06	26.89	57.80	58.27	57.60	57.78
5	2	10	1.37	2.25	2.30	2.28	2.32	2.34	2.35	2.35
5	4	20	2.17	7.39	7.29	7.40	7.75	7.74	7.84	7.71
5	6	30	2.92	15.20	14.84	15.04	20.96	20.99	21.14	21.00
5	8	40	3.91	26.59	25.82	26.22	45.41	45.48	45.62	45.40
5	10	50	5.55	40.92	40.53	40.59	88.88	88.54	88.96	88.89
6	2	12	1.51	2.80	2.86	2.88	2.99	3.01	2.97	2.94
6	4	24	2.56	10.09	9.91	9.89	10.47	10.76	10.72	10.63
6	6	36	3.49	21.09	20.63	20.74	29.33	29.46	29.28	29.33
6	8	48	4.59	37.01	36.32	36.52	64.39	64.30	65.02	65.05
6	10	60	6.49	56.91	56.61	56.10	124.02	124.81	126.21	127.09
Combined Mean:			2.97	16.08	15.87	15.88	28.55	28.61	28.68	28.70

3.2 Repeated SSC auctions with Dynamic Task Clusters

The auction algorithm we have described so far is a non-optimal one-shot task allocation auction algorithm and suffers from the problems of greedy initial bias and local minima as previously described in Section 2.4.6. Previously studied approaches for improving this is to repeatedly auction and redistribute tasks that are not completed either at certain time intervals or upon each single task completion [25, 81] . The benefits of this are two-fold:

1) it is hoped that robots will gradually improve the quality of their solution; and
2) any unexpected failures or delay in the execution of the current allocation can be addressed through reallocation.

In this section we apply the ideas of repeated auctions of uncompleted tasks to SSC auctions. Our approach is, upon completion of a single task, all robots create new clusters of their uncompleted tasks and auction these task clusters. This allows inter-task synergies that were not previously explored to be considered and as a result the new allocation may alter the team cost. During the repeated auction process any robot which is actively travelling towards a task continue to do so. This enables the reallocation of uncompleted tasks to occur in parallel with active task completion. This approach differs from previous work on repeated auctions in that the dynamic generation of differing task cluster memberships explores different synergies where previous work has, due to static task costs, modified the team objective.

3.2.1 Algorithm Description

We now describe our procedure for repeated auctions of task clusters upon individual task completion. We make the same assumptions as our previous algorithm (Section 3.1.4) with the additional rule that auctions are run sequentially, that is, if an auction is running after the completion of a task and a second robot completes a task, the auction generated by the second task completion does not begin until after the first auction is complete. We also make the assumption that auctions complete as quickly as possible and the time to complete tasks is much greater than the time to run an auction to reallocate tasks.

Our algorithm operates as follows: when a robot completes a task it signals to all other robots that an auction for the redistribution of tasks is to begin. All robots then create clusters of the uncompleted tasks they are currently allocated, and inform all other robots of these clusters, excluding the cluster containing the task they are currently completing. The robots then all run an SSC auction on these clusters. After all clusters have been distributed via the auction each robot replans its path based on its new allocation of tasks. While the auction runs each robot continues to complete its current task.

The algorithm that each robot executes is presented in Figure 3.11. All robots are assigned an initial allocation of tasks and a clustering factor which is used to calculate the number of task clusters required. The algorithm runs on all robots until all tasks are completed. Initially each robot plans a path to complete all allocated tasks (Line 1). Each robot then drives to the first task in its path and in parallel listens for a signal to begin an auction (Line 4). However, if a robot has no tasks currently allocated to it then it just listens for the signal to begin an auction (Line 7).

The function DriveToTask (Lines 8-13) controls the movement of the robot towards its current task and signals the start of an auction on arrival at the task. The robot will continue travelling towards its current task until it reaches it (Lines 8-9). Once the robot arrives at its current task it will signal all robots to begin an auction and wait until the auction is complete (Lines 10-11). Upon completion of the auction the robot will remove its current task from its set of tasks to complete (Lines 12-13). The function then returns,

function RepeatedSSCAuctions (T, r_i, R, F_{r_i}, *CF*)
Input: T: the set of Tasks to be completed
r_i: the robot
R: the set of robots
F_{r_i}: the set of completed tasks
CF: the factor of clusters to tasks
Output: $T = \emptyset$: All tasks completed
1: $P_{T_{r_i}} \leftarrow$ MinimisePath(T_{r_i})
2: **while** ($T \neq \emptyset$)
3: **if** ($T_{r_i} \neq \emptyset$)
4: DriveToTask($t_0 \in P_{T_{r_i}}$, r_i, R, F_{r_i}) |
$T_{r_i} \leftarrow$ListenForAuction($t_0 \in P_{T_{r_i}}$, T_{r_i}, R, *CF*)
5: $P_{T_{r_i}} \leftarrow$ MinimisePath(T_{r_i})
6: **else**
7: $T_{r_i} \leftarrow$ListenForAuction($t_0 \in P_{T_{r_i}}$, T_{r_i}, R, *CF*)

function DriveToTask (t, r_i, R, F_{r_i})
Input: t: the task to be completed
r_i: the robot
R: the set of robots
F_{r_i}: the set of completed tasks
Output: $r_{il} = t_l$: The robot r_i located at task t
8: **while** ($r_{il} \neq t_l$)
9: Robot r_i moves towards task t
10: Signal all robots $r_j \in R$ to begin auction
11: Wait for auction to finish
12: $F_{r_i} \leftarrow F_{r_i} \cup \{t\}$
13: $T_{r_i} \leftarrow T_{r_i} \backslash \{t\}$

Figure 3.11: Algorithm for Repeated Auctions with Dynamic Clustering.

function ListenForAuction (t, T_{r_i}, r_i, R, *CF*)
Input: t: the currently initialised task
T_{r_i}: the set of Tasks allocated to the robot
r_i: the robot
R: the set of robots
CF: the factor of clusters to tasks
Output: T_{r_i}: the set of Tasks allocated to the robot
14: Wait for signal to begin auction
15: $K_{r_i} \leftarrow$ Clustering($k \leftarrow CF * |T_{r_i}|, T_{r_i}$)
16: $U_{r_i} \leftarrow K_{r_i} \backslash \{C_t \in K_{r_i}\}$
17: Send(U_{r_i}, R) | $U \leftarrow \bigcup_{1 \leq j \leq |R|}$ Receive(U_{r_j}, R)
18: **if** ($U \neq \emptyset$)
19: $K_{r_i} \leftarrow$ SSC-Auction(U, K_{r_i}, r_i, R)
20: $T_{r_i} \leftarrow \{t \in C | C \in K_{r_i}\}$
21: **if** ($r_{il} = t_l$)
22: Signal Auction Finished
23: **else if** ($T_{r_i} \neq \emptyset$)
24: ListenForAuction(t, T_{r_i}, r_i, R, *CF*)

Figure 3.12: Algorithm for Repeated Auctions with Dynamic Clustering (cont.).

the robot replans its path to travel to complete its remaining tasks (Line 5), and if the robot still has tasks to complete it will begin travelling to the next task.

The function ListenForAuction (Lines 14-24) sets up and controls the SSC auction for reallocating the assigned tasks. Upon the signalling of an auction each robot forms clusters of tasks (Line 14). To determine k in each auction we multiply the constant cluster factor *CF* by the number of currently allocated tasks. The constant cluster factor is a value between 0 and 1 and describes the ratio of tasks to clusters. In our experiments we typically use values of $\frac{1}{2}$ and $\frac{2}{3}$ for *CF*. This gives us an average of 2 tasks/cluster and 1.5 tasks/cluster respectively. After the formation of the task clusters the robot removes the cluster containing its currently initialised task (Line 16). Here C_t refers to the cluster that contains task t. This subset forms the robot's contribution to the set of clusters for auction which is sent to all robots. In parallel the robot receives a set of clusters from all other robots. These sets are merged to form the complete set of all clusters for auction (Line 17). If there are clusters for auction, all robots run the SSC auction algorithm simultaneously (Line 19). The robot then resets its allocated tasks with all tasks in the post-auction cluster set (Line 20). Finally, the robot that signalled the start of the auction signals the end of the auction (Line 22) and other robots with tasks to complete listen for the next auction (Line 24).

3.2.2 A Simple Example

Figure 3.13 provides an example of an auction using our algorithm with the MiniMax team objective. We have four robots and 24 tasks. Each robot has an initial allocation of tasks such that: $T_{r_1} = \{t_1, \ldots, t_6\}$, $T_{r_2} = \{t_7, \ldots, t_{12}\}$, $T_{r_3} = \{t_{13}, \ldots, t_{18}\}$ and $T_{r_4} = \{t_{19}, \ldots, t_{24}\}$ (Figure 3.12a). All robots then move towards their first task. Robot r_1 signals the start of an auction after arriving at task t_1. At this point r_2 is approaching t_{10}, r_3 is approaching t_{15}, and r_4 is approaching t_{24}. Each robot with $CF = \frac{1}{2}$ then uses a clustering algorithm to allocate its tasks into three task clusters (Figure 3.12b). All robots then remove the cluster containing their currently initialised task and exchange their remaining clusters with all robots to form the clusters for auction $U = \{C_1, \ldots, C_8\}$. The SSC auction then begins (Figure 3.12c). During the first four bidding rounds C_2 is allocated to r_1, C_3 is allocated to r_2, C_6 is allocated to r_3, and C_7 is allocated to r_4. The next four bidding rounds allocate the remaining clusters. C_5 is allocated to r_1, C_1 is allocated to r_2, C_8 is allocated to r_3, and C_4 is allocated to r_4. The complete new allocation is shown in Figure 3.12d.

3.2.3 Communication Costs

For repeated SSC auctions with dynamic clusters the number of messages exchanged in each auction round is $|R|^2$ as previously defined in Section 3.1.5. At the beginning of each auction all robots have to exchange their clusters to be auctioned. As such, there is an additional $|R|^2$ number of messages exchanged. There are then $|U|$ rounds per auction. Therefore in total there are $(1 + |U|) * |R|^2$ messages per auction and there are at most $|T|$ auctions one for each task completion. We can conclude that for repeated SSC auctions with dynamic clusters there are much greater communication requirements than auctions that only assign tasks once.

3.2.4 Experimental Setup

We test our algorithm on the same scenario as our previous experiments. For each configuration we test with $|R| \in \{4, 6, 8, 10\}$, $|T| \in \{16, \ldots, 60\}$, and provide results for both the MiniMax and MiniSum team objectives. Each experiment configuration is tested with two different initial task allocations: SSC auctions with $k = \frac{1}{2}|T|$ and $k = \frac{2}{3}|T|$. Each of these initial allocations produces a differing task to robot assignment and therefore different initial paths and costs. These initial path costs are included in our results (Tables 3.5 and 3.6) and provide reference for how much repeated auctions improve the overall team objective. We also test each experimental configuration with two different clustering factors for calculating the number of clusters required, $CF = \frac{1}{2}$ and $CF = \frac{2}{3}$.

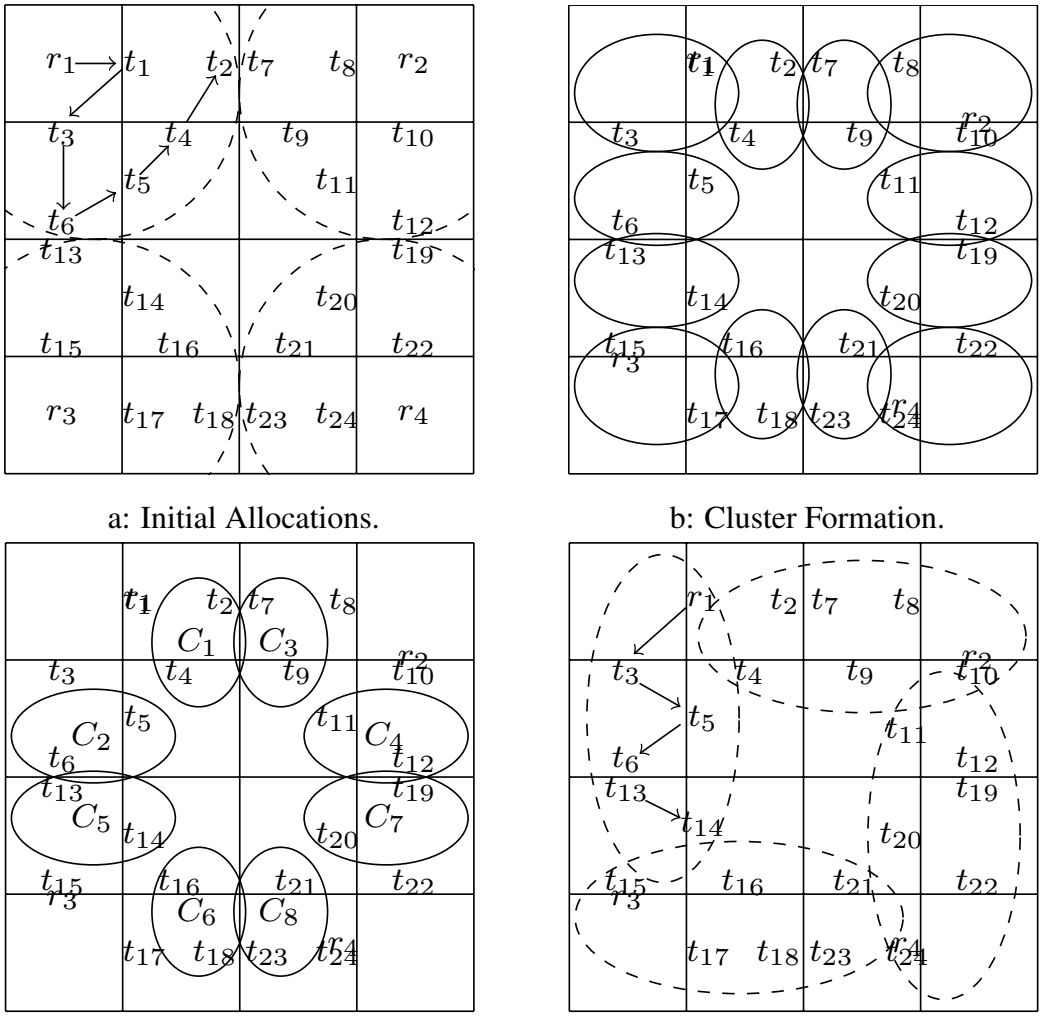


Figure 3.13: Example of a repeated SSC auction lowering the team costs (dashed lines show approximate task allocation boundaries, solid ellipses show clusters).

Capacity Constraints

In each experiment, robots are set a fixed maximum total task fulfilment capacity constraint of double the number of tasks divided by the number of robots:

$$|T_{r_i} \cup F_{r_i}| \leq 2 * \frac{|T|}{|R|}$$

These capacity constraints are twice the size of the capacity constraints in our earlier experiments. However, this is necessary due to the design and application of the constraint in the algorithm. Previously, hard and soft capacities dictated how strictly the capacity constraint was applied. Furthermore, due to clusters having varying numbers of tasks, SSC auctions could only operate with soft capacity constraints. With repeated auctions of

dynamically formed clusters we can, over time, increase the strictness of the application of the capacity constraint. This is due to the number of items in each cluster decreasing to 1 as the number of uncompleted tasks decreases. As a result of this, when the number of tasks is high, robots will be allocated tasks that may cause them to exceed their capacity constraints. Then, throughout task completion, as task clusters are reformed and uncompleted tasks redistributed, the number of tasks completed by each robot can be limited to exactly its capacity constraint.

3.2.5 Results

MiniMax

Table 3.5 shows the mean results for the MiniMax team objective. Repeated auctions improved on the initial task allocations in all robot/task combinations. Generally, the largest improvements occurred when the initial task allocation had been generated by $k = \frac{1}{2}|T|$. However, the initial set of allocations generated by $k = \frac{1}{2}|T|$ also had higher initial costs than those generated by $k = \frac{2}{3}|T|$. Repeated auctions with clusters generated by a clustering factor of $CF = \frac{2}{3}$ also generally produced improvements greater than those generated by $CF = \frac{1}{2}$. This is particularly apparent in comparing the results where the initial allocation was generated by $k = \frac{1}{2}|T|$ and improved by $CF = \frac{2}{3}$ to those initially generated by $k = \frac{2}{3}|T|$ and improved by $CF = \frac{1}{2}$. Despite the higher initial costs generated by $k = \frac{1}{2}|T|$ after the repeated auctions with $CF = \frac{2}{3}$, the average final cost was lower than those that had begun with a lower initial cost and improved by $CF = \frac{1}{2}$. Finally, unlike the original experimental results for SSC auctions there does not appear to be any trends that the size of the improvement is proportional to the number of tasks or robots.

MiniSum

The mean results for the MiniSum team objective are presented in Table 3.6. In these results, repeated auctions also improved on the initial task allocations in all robot/task combinations. However, the size of the improvement is much smaller than those of the MiniMax team objective. Again, the initial cost generated by $k = \frac{1}{2}|T|$ is larger than that generated by $k = \frac{2}{3}|T|$. This result is a direct reversal of the earlier MiniSum team objective results in which $k = \frac{2}{3}|T|$ generated higher costs than $k = \frac{1}{2}|T|$. We suspect that the reversal of this pattern is due to the larger capacity constraint in this set of experiments. In the earlier experiments, robots gained a greater benefit by slightly exceeding the soft capacity constraint, whereas with a larger capacity constraint not all robots will exceed it and therefore the corresponding benefit is less. As a result, the size of the improvement in

Table 3.5: Mean Repeated SSC Auctions Results for the MiniMax team objective (percentage improvement of final cost compared to initial cost in brackets).

			Initial Allocation $k = \frac{1}{2}\|T\|$			Initial Allocation $k = \frac{2}{3}\|T\|$		
Capacity	Robots	Tasks	Initial Cost	Final Cost $CF = \frac{1}{2}$	Final Cost $CF = \frac{2}{3}$	Initial Cost	Final Cost $CF = \frac{1}{2}$	Final Cost $CF = \frac{2}{3}$
4	4	16	1014	935 (7.8%)	876 (13.6%)	913	908 (0.6%)	895 (2.0%)
4	6	24	993	854 (14.0%)	820 (17.4%)	822	812 (1.2%)	756 (8.0%)
4	8	32	892	774 (13.2%)	720 (19.2%)	829	774 (6.7%)	768 (7.4%)
4	10	40	819	742 (9.4%)	679 (17.1%)	728	661 (9.2%)	682 (6.3%)
5	4	20	1097	967 (11.9%)	972 (11.4%)	999	966 (3.4%)	982 (1.7%)
5	6	30	1012	902 (10.9%)	875 (13.5%)	930	857 (7.8%)	839 (9.7%)
5	8	40	901	781 (13.2%)	773 (14.1%)	821	768 (6.4%)	747 (9.0%)
5	10	50	815	746 (8.4%)	711 (12.8%)	764	726 (5.0%)	705 (7.6%)
6	4	24	1213	1116 (8.0%)	1028 (15.2%)	1028	1032 (-0.3%)	985 (4.2%)
6	6	36	1122	950 (15.4%)	950 (15.4%)	954	931 (2.5%)	854 (10.5%)
6	8	48	994	848 (14.7%)	827 (16.8%)	890	813 (8.7%)	811 (8.9%)
6	10	60	888	764 (14.0%)	721 (18.8%)	863	760 (11.9%)	734 (15.0%)
Combined Mean:			980	865 (11.7%)	829 (15.4%)	878	834 (5.0%)	813 (7.4%)

Table 3.6: Mean Repeated SSC Auctions Results for the MiniSum team objective (percentage improvement of final cost compared to initial cost in brackets).

			Initial Allocation $k = \frac{1}{2}\|T\|$			Initial Allocation $k = \frac{2}{3}\|T\|$		
Capacity	Robots	Tasks	Initial Cost	Final Cost $CF = \frac{1}{2}$	Final Cost $CF = \frac{2}{3}$	Initial Cost	Final Cost $CF = \frac{1}{2}$	Final Cost $CF = \frac{2}{3}$
4	4	16	2356	2286 (3.0%)	2307 (2.1%)	2367	2276 (3.9%)	2272 (4.0%)
4	6	24	2905	2736 (5.8%)	2801 (3.6%)	2696	2696 (0.0%)	2780 (-3.1%)
4	8	32	3245	3131 (3.5%)	3125 (3.7%)	3233	3173 (1.8%)	3078 (4.8%)
4	10	40	3599	3309 (8.1%)	3296 (8.4%)	3266	3200 (2.0%)	3238 (0.9%)
5	4	20	2729	2685 (1.6%)	2657 (2.6%)	2670	2652 (0.7%)	2603 (2.5%)
5	6	30	3293	3192 (3.1%)	3175 (3.6%)	3189	3063 (4.0%)	3021 (5.3%)
5	8	40	3702	3467 (6.4%)	3478 (6.1%)	3467	3447 (0.6%)	3543 (-2.2%)
5	10	50	3859	3671 (4.9%)	3669 (4.9%)	3745	3591 (4.1%)	3635 (2.9%)
6	4	24	3101	2939 (5.3%)	2891 (6.8%)	2893	2858 (1.2%)	2846 (1.6%)
6	6	36	3599	3386 (5.9%)	3474 (3.5%)	3457	3406 (1.5%)	3371 (2.5%)
6	8	48	4092	3740 (8.6%)	3810 (6.9%)	3975	3767 (5.2%)	3779 (4.9%)
6	10	60	4315	4065 (5.8%)	4074 (5.6%)	4321	4121 (4.6%)	4160 (3.7%)
Combined Mean:			3400	3217 (5.4%)	3230 (5.0%)	3273	3187 (2.6%)	3194 (2.4%)

final cost compared to initial cost for $k = \frac{1}{2}|T|$ is double the improvement where initial costs are generated by $k = \frac{2}{3}|T|$. Unlike the MiniMax team objective results there is no clear advantage of either clustering factor outperforming the other. There is also no indication that the size of the improvement is dependent on the number of tasks or robots.

3.3 Summary

In this chapter we have shown the benefits of SSC auctions as an alternative to SSI auctions for the allocation of tasks to robots. We developed the theoretical foundations of SSC auctions and outlined their unique behavioural properties. Using the standard multi-robot routing testbed we demonstrated empirically that SSC auctions can produce smaller team objective results than SSI auctions. We also compared these results to another extension of SSI-like auctions, SSI auctions with bundles, and showed that SSC auctions perform much quicker.

Our second set of experiments focused on improving the solution to the task allocation problem as robots completed their tasks. This algorithm uses repeated auctions to reallocate uncompleted tasks among robots as individual tasks are completed. The results of our empirical experiments demonstrate this algorithm ensures a better final allocation of tasks to robots than algorithms that only allocate tasks once.

These results give us scope for further investigation into the use of task clustering in auctions for robot task allocation. In the next chapter we investigate the properties and techniques of cluster formation and the impact of differing cluster formation techniques on multi-robot task allocation. In the following chapter we extend SSC auctions to operate in dynamic environments where tasks are dynamically inserted into the running system.

Chapter 4

Cluster Formation Techniques for SSC Auctions

In the previous chapter we developed SSC auctions for solving MRTA problems and demonstrated their benefits using K-means clustering. In this chapter we use SSC auctions to compare *centroid-based clustering* (e.g. K-means) with *agglomerative clustering*. We also consider the differences between *straight line distance* (SLD) and *true path distance* (TPD) (which takes into consideration obstacles between tasks) as cost metrics in cluster formation. Overall, our empirical results show agglomerative clustering with a TPD metric when used in SSC auctions produces low cost solutions to MRTA problems. Additionally, our analysis of the time required to form clusters shows that using a TPD metric is around 100 times slower than using straight line distances.

This chapter is structured as follows. First, we provide an algorithm for, and an example of, each clustering technique (Section 4.1). Next, we consider the time required for each clustering algorithm to complete (Section 4.2). We then report our empirical results for each clustering technique when used in SSC auctions for task allocation (Section 4.3), and also consider the effects of a priority allocation of clusters based upon the number of items in individual clusters (Section 4.4). We conclude with a discussion of additional questions raised in the experimental results (Section 4.5).

4.1 Clustering Techniques

We now consider two different models of cluster formation. In centroid-based clustering each task is assigned to a single cluster based on the task's proximity to the central vector of the cluster. During the formation of clusters, tasks transfer between clusters which in turn, may cause the central vector of each cluster to change, and this process continues until an equilibrium is found where no tasks move. For instance, in K-means clustering, a cluster's centre of mass is its central vector and in *K-medoids clustering* individual tasks

Portions of the research in this chapter have been published in *Analysis of Cluster Formation Techniques for Multi-Robot Task Allocation using Sequential Single-Cluster Auctions* (AI-12).

function Centroid (T,k)
Input: T: the set of tasks to be clustered
k: the number of clusters to form
Output: K: the set of clusters

1: $V \leftarrow$ InitialiseCentralVectors(T,k);
2: **while** *cluster memberships have changed*
3: */* Task Cluster Assignment Stage */*
4: **for each** task $t \in T$
5: $\lambda^t_{v_j} \leftarrow \infty$;
6: **for each** central vector $v_i \in V$
7: $\lambda^t_{v_i} \leftarrow$CalcDistance($t$,$v_i$);
8: $\lambda^t_{v_j} \leftarrow \min(\lambda^t_{v_j}, \lambda^t_{v_i})$;
9: $C_{v_j} \leftarrow C_{v_j} \cup \{t\}$;
10: */* Update Central Vectors Stage */*
11: **for each** cluster $C \in K$
12: $V_C \leftarrow$CalcCentralVector(C);

Figure 4.1: Algorithm for Centroid-based Clustering.

are the central vectors. In agglomerative clustering pairs of clusters are recursively merged according to certain linkage criteria, of which *single linkage* and *complete linkage* are two common linkage criteria. Single linkage clustering combines two clusters according to the minimum cost between any one task and any other task in a different cluster. Complete linkage clustering merges clusters by minimising the maximum cost between any one task and any other task in a different cluster.

Each clustering technique uses a cost metric to calculate the distance between tasks and clusters. In this analysis we consider the differences between two common metrics. The first metric, SLD, uses simple straight line Euclidean distances between two points. Although this metric requires few computations it fails to consider obstacles between these points (e.g. closed doors and walls). As an alternative, the TPD metric measures the actual cost required for a robot to travel between two points taking into consideration walls and other known obstacles. To calculate the TPD metric we require an occupancy grid map or similar data structure describing the operating environment and an algorithm to find a path from one point in this environment to another. As a result, the calculation of the TPD metric is much slower in comparison to the SLD metric.

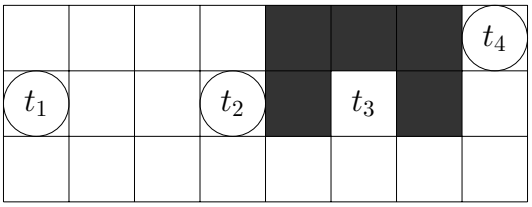
4.1.1 Centroid-based Clustering

We present a metric-independent centroid-based clustering algorithm in Figure 4.1. Before the main loop of the algorithm begins, the initial central vectors of all clusters must be selected (Line 1). A common approach for this is to randomly select k tasks from the set of data to be clustered and then use each of these as initial central vectors. The algorithm

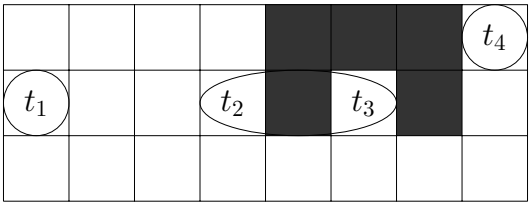
function CalcCentralVector(C)
Input: C: the cluster
Output: V_c: the central vector of this cluster

1: $x \leftarrow 0$
2: $y \leftarrow 0$
3: **for each** task $t \in C$
4: $\quad x \leftarrow x + t_x$
5: $\quad y \leftarrow y + t_y$
6: $V_{cx} \leftarrow \frac{x}{|C|}$
7: $V_{cy} \leftarrow \frac{y}{|C|}$

Figure 4.2: Algorithm for *K*-means clustering.



a: Initial Cluster Centres.



b: Stable Cluster Centres.

Figure 4.3: Formation of three clusters of four tasks using *K*-means clustering with a straight line distance metric.

then alternates between two stages until the membership of all clusters is stable. During the *task cluster assignment* stage (Lines 3 - 9) every task is considered independently. The distance between the task and every cluster's central vector is calculated and the task is reassigned from its current cluster to the cluster with the minimum distance to itself. During the *update central vectors* stage (Lines 10 - 12) the central vector of each cluster is recalculated to reflect the changes in the membership of each cluster. The algorithm then repeats until no task moves between clusters.

K-means Clustering

K-means clustering is a very common form of centroid-based clustering where the central vector of each cluster is the centre of mass of the locations of each individual item in the cluster [69]. We give an algorithm for calculating the central vector of each cluster in Figure 4.2 and give a simple example of clustering using the *K*-means algorithm in Figure 4.3. In this example we use a SLD metric to calculate the cost from each individual task to the central vectors of each cluster. Using Pythagoras' theorem each distance calculation is computationally small even if there are many changes in the central vectors of clusters. In contrast, if we apply a TPD metric in the cost calculation, that considers obstacles such as walls and doors in a map, the cost calculations require much more computation which we explore in further detail in Section 4.2. Overall, *K*-means clustering effectively partitions a set of tasks into a Voronoi diagram with non-overlapping partitions.

K-medoids Clustering

K-medoids clustering is a modification of *K*-means clustering such that the central vector of the cluster must be a member item of the cluster [90]. This approach works on minimising the pairwise distances between items in each cluster as opposed to forming clusters around arbitrary points in free space. We present an algorithm for calculating the central vector of a cluster as a *K*-medoid in Figure 4.4. In this algorithm we first calculate the cost of items in the cluster relative to the current medoid (Lines 1-5). We then calculate the costs using every other item in the cluster as a possible medoid (Lines 5-14). If an alternative item in the cluster has a lower total cost to all other items in the cluster than the current medoid, then it becomes the new medoid for the cluster (Lines 15-16). We note that each iteration of the *K*-medoids algorithm requires calculations of the distances between each task and every other task in a cluster for updating each cluster's medoid. As such, in problems where an equilibrium is found in few iterations, standard *K*-means clustering is likely to be much faster than *K*-medoids. However, if we cache the inter-task distance calculations, in problems that take many iterations to find an equilibrium, *K*-medoids can be faster than *K*-means.

function CalcCentralVector(C, V_c)
Input: C: the cluster
V_c: the current central vector of this cluster
Output: V_c: the central vector of this cluster

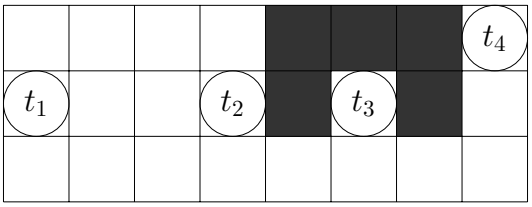
1: */* Current Medoid Cost Calculation */*
2: $\lambda_{V_c} \leftarrow 0$
3: **for each** task $t \in C$
4: $\lambda_{V_c} \leftarrow \lambda_{V_c}$+calcDistance($V_c, t$)
5: */* Replacement Medoid Cost Calculation */*
6: $\lambda_{\text{new}} \leftarrow \infty$
7: $V_{\text{new}} \leftarrow \emptyset$
8: **for each** task $t_i \in C$
9: $\lambda_{t_i} \leftarrow 0$
10: **for each** task $t_j \in C$
11: $\lambda_{t_i} \leftarrow \lambda_{t_i}$+calcDistance($t_i, t_j$)
12: **if** ($\lambda_{t_i} < \lambda_{\text{new}}$)
13: $\lambda_{\text{new}} \leftarrow \lambda_{t_i}$
14: $V_{\text{new}} \leftarrow t_i$
15: **if** ($\lambda_{\text{new}} < \lambda_{V_c}$)
16: $V_c \leftarrow V_{\text{new}}$

Figure 4.4: Algorithm for *K*-medoids Clustering.

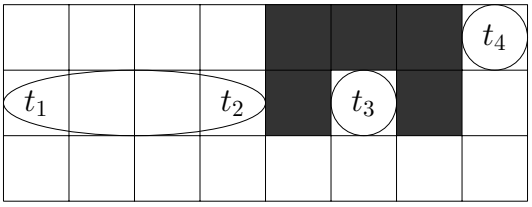
function Agglomerative (T,k)
Input: T: the set of tasks to be clustered
k: the number of clusters to form
Output: K: the set of clusters

1: $K \leftarrow \emptyset$;
2: **for each** task $t_i \in T$
3: $C_i \leftarrow \{t_i\}$;
4: $K \leftarrow K \cup C_i$;
5: **while** $|K| > k$
6: $\lambda_{C_b}^{C_a} \leftarrow \infty$;
7: **for each** cluster $C_i \in K$
8: **for each** cluster $C_j \in K \backslash C_i$
9: $\lambda_{C_j}^{C_i} \leftarrow$CalcLinkage($C_i$,$C_j$);
10: $\lambda_{C_b}^{C_a} \leftarrow \min(\lambda_{C_b}^{C_a}, \lambda_{C_j}^{C_i})$
11: $C_{\text{merged}} \leftarrow C_a \cup C_b$;
12: $K \leftarrow K \cup \{C_{\text{merged}}\}$;
13: $K \leftarrow K \backslash \{C_a, C_b\}$;

Figure 4.5: Algorithm for Agglomerative Clustering.



a: Initial Clusters.



b: Final Merged Clusters.

Figure 4.6: Formation of three clusters of four tasks using agglomerative clustering with a true path distance metric.

4.1.2 Agglomerative Clustering

We now consider agglomerative clustering which runs in exactly k iterations and, unlike centroid-based clustering, once any two tasks are paired together they are never unpaired. In Figure 4.5 we present an algorithm to perform agglomerative clustering and give an example cluster formation with a TPD metric in Figure 4.6. The algorithm begins with each task being assigned to a cluster containing only itself (Lines 1 - 4) (Figure 4.6a). Clusters are then repeatedly merged until the number of clusters is equal to k (Lines 5 - 13) (Figure 4.6b). To merge clusters we calculate the distance between every individual task in each cluster and every task in every other cluster (Lines 7 - 10). The linkage cost calculation (Line 9) is a key component of agglomerative clustering as it determines the relative cost of merging two existing clusters together. The two clusters containing the two tasks with the minimum linkage distance between them are merged (Lines 11 - 13). The algorithm then repeats until there are k clusters.

We now detail two common linkage criteria:

Single Linkage Clustering

Single linkage clustering merges clusters according to the minimum cost between any one task and any other task in a different cluster [112]. The linkage cost calculation function for this type of clustering is given in Figure 4.7. In this function the distance between

function CalcLinkage(C_i,C_j)
Input: C_i: a cluster of tasks
C_j: a second cluster of tasks
Output: $\lambda_{C_j}^{C_i}$: the linkage cost

1: $\lambda_{C_j}^{C_i} \leftarrow \infty$;
2: **for each** task $t_i \in C_i$
3: **for each** task $t_j \in C_j$
4: $\lambda_{t_j}^{t_i} \leftarrow$CalcDistance($t_i$,$t_j$);
5: **if** $\lambda_{t_j}^{t_i} < \lambda_{C_j}^{C_i}$ **then**
6: $\lambda_{C_j}^{C_i} \leftarrow \lambda_{t_j}^{t_i}$;

Figure 4.7: Algorithm for Single Linkage Criteria.

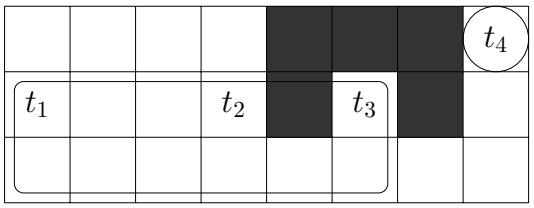


Figure 4.8: Formation of two clusters of four tasks using single linkage clustering with a true path distance metric.

every task in one cluster is calculated to every task in another cluster. The minimum distance between any two tasks in different clusters becomes the linkage cost.

We now extend the previous example given in Figure 4.6 and lower the number of clusters required to two. Using single linkage clustering as the linkage criteria, the first merge from four clusters (Figure 4.6a) to three clusters (Figure 4.6b) remains the same. To now merge from three clusters to two, we calculate the linkage cost from each cluster to every other cluster. Using a TPD metric, in this example, the costs for each merge are:

$$\lambda_{\{t_3\}}^{\{t_1,t_2\}} = \text{CalcDistance}(t_2,t_3) = 4$$
$$\lambda_{\{t_4\}}^{\{t_1,t_2\}} = \text{CalcDistance}(t_2,t_4) = 7$$
$$\lambda_{\{t_4\}}^{\{t_3\}} = \text{CalcDistance}(t_3,t_4) = 5$$

The lowest cost is to merge the clusters $\{t_1, t_2\}$ and $\{t_3\}$ and the final two clusters are shown in Figure 4.8.

This approach to cluster formation generates cluster shapes that are very different to that of centroid-based clustering. Clusters generated by *K*-means and *K*-medoids clustering never overlap and the tasks inside each cluster are all geographically near to each other. However, in single linkage clustering, because the merging of two clusters depends on the distance between any two closest tasks, the shapes formed are generally long narrow chains [78]. As a result, in our experiments, due to the differences in the approach of these two algorithms, we expect vastly different task allocations to robots. In centroid-based clustering we expect that each robot will be generally constrained to completing tasks within an isolated area. In comparison, task clusters formed using single linkage clustering are more likely to see robot paths crossing over each other.

Complete Linkage Clustering

Complete linkage clustering connects clusters of tasks according to the highest cost from any one task to any other task in a different cluster. Across the set of clusters the two clusters with minimal high inter-task costs are merged. As a result, this approach generates dense clusters of geographically close tasks which are similar in shape to that of *K*-means clustering.

Figure 4.9 presents the linkage cost calculation function for complete linkage clustering. This function is nearly identical to the previous function for single linkage clustering, with the exception that the inequality to determine the cost between tasks (Line 5) has been reversed so that the largest distance between any two tasks becomes the linkage cost.

We again extend the previous example given in Figure 4.6 and lower the number of clusters required to two. The first merge from four clusters (Figure 4.6a) to three clusters (Figure 4.6b) remains the same and, using a TPD metric, the costs for the next merge are:

$$\lambda^{\{t_1,t_2\}}_{\{t_3\}} = \text{CalcDistance}(t_1,t_3) = 7$$
$$\lambda^{\{t_1,t_2\}}_{\{t_4\}} = \text{CalcDistance}(t_1,t_4) = 10$$
$$\lambda^{\{t_3\}}_{\{t_4\}} = \text{CalcDistance}(t_3,t_4) = 5$$

The lowest cost is to merge the clusters $\{t_3\}$ and $\{t_4\}$ and the final two clusters are shown in Figure 4.10.

function CalcLinkage(C_i,C_j)
Input: C_i: a cluster of tasks
C_j: a second cluster of tasks
Output: $\lambda^{C_i}_{C_j}$: the linkage cost

1: $\lambda^{C_i}_{C_j} \leftarrow 0$;
2: **for each** task $t_i \in C_i$
3: **for each** task $t_j \in C_j$
4: $\lambda^{t_i}_{t_j} \leftarrow$CalcDistance($t_i$,$t_j$);
5: **if** $\lambda^{t_i}_{t_j} > \lambda^{C_i}_{C_j}$ **then**
6: $\lambda^{C_i}_{C_j} \leftarrow \lambda^{t_i}_{t_j}$;

Figure 4.9: Algorithm for Complete Linkage Criteria (difference between Figure 4.7 highlighted in yellow).

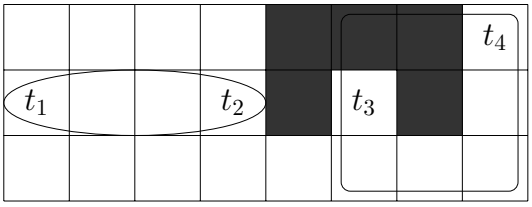


Figure 4.10: Formation of two clusters of four tasks using complete linkage clustering with a true path distance metric.

4.2 Cluster Formation Time Analysis

Minimising the time required to formulate clusters is crucial to efficiently finding good solutions to MRTA problems. In centroid-based clustering, the time complexity for each iteration is $O(k|T|)$ and the calculation of each central vector is $O(|C|)$ for *K*-means clustering and $O(|C|^2)$ for *K*-medoids clustering. Additionally, the total number of possible different cluster formations can be approximated by $k^{|T|}/k!$ [54] and, as such, the number of iterations until a solution converges may be extremely large, unless a constraint on the maximum number of iterations is applied. Further compounding this for *K*-means clustering is that few distance calculations can be cached. In contrast, the time complexity for agglomerative clustering is $O(|T|^2 \log k)$ as the distance between every item and every other item is calculated only once and only one possible cluster formation is generated for any given number of clusters k [61].

The time to calculate the distance metric also needs to be accounted for. As previously discussed in Section 4.1, the time required for calculation of the straight line Euclidean distance between tasks is minimal compared to the time required to calculate the true path distance between tasks taking into account obstacles. Furthermore, the total number of path calculations required for agglomerative clustering and *K*-medoids clustering is constrained by the number of tasks. Overall, generally speaking, agglomerative clustering is quicker than centroid-based clustering, especially *K*-means clustering.

To empirically evaluate the real time requirements of each clustering algorithm we simulate 25 office-like environments as previously described in Section 3.1.7. For each configuration we test a wide range of total tasks to be clustered $|T| \in \{16, \ldots, 60\}$. For each task set we repeat the clustering process for two different values of k, $k = \frac{1}{2}|T|$ and $k = \frac{2}{3}|T|$. We use *K*-means clustering as our centroid-based clustering algorithm with a maximum of 1,000 iterations for the SLD metric and 50 iterations for the TPD metric. In most configurations tested *K*-means clustering achieves equilibrium before these limits are reached. However, in a few configurations these limits were applied and the much lower number of iterations for the TPD metric was required as the time otherwise required for repeated cost calculations would heavily skew the results. For agglomerative clustering we use single-linkage clustering as our linkage criteria. For the calculation of the true path cost between tasks we perform an A* search on an occupancy grid map of each office environment with the Euclidean line distance between the two tasks used as an heuristic.

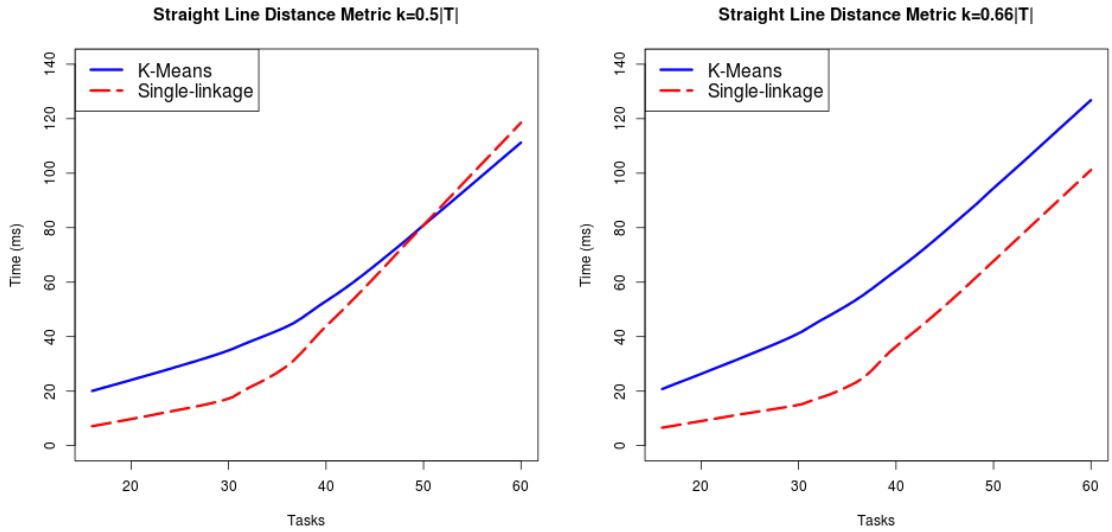


Figure 4.11: Cluster Formation Time using a SLD Metric.

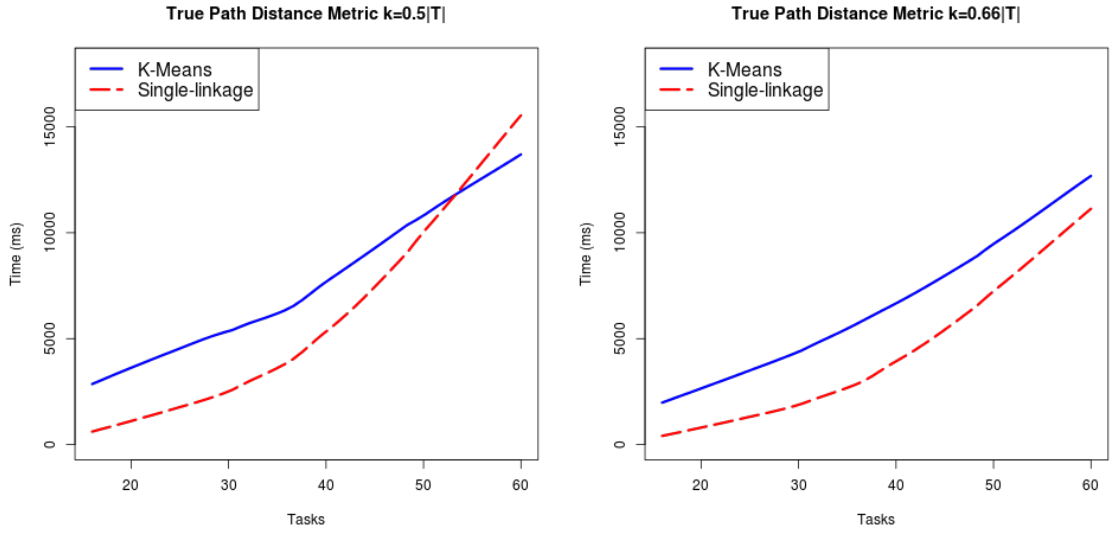


Figure 4.12: Cluster Formation Times using a TPD Metric.

The results of clustering using a SLD metric are plotted in Figure 4.11 and the results of clustering using a TPD metric are plotted in Figure 4.12. These plots show that generally, as expected, agglomerative clustering completes quicker than K-means clustering. However, when there are a large number of tasks and $k = \frac{1}{2}|T|$ agglomerative clustering takes a long period of time to complete. This is due to the large number of cluster merges required when k is small. We note that K-means clustering does not suffer this problem as the stabilisation of clusters is independent of the value of k. Finally, we also observe that the use of a TPD metric causes both clustering algorithms to perform about 100 times slower compared to the SLD metric.

4.3 Empirical Analysis using SSC Auctions

We now test each clustering technique with SSC auctions to solve the multi-robot task allocation problem for both the MiniMax and the MiniSum team objectives with a homogeneous mobile robot team of varying size $|R| \in \{4, 6, 8, 10\}$. In each of the 25 office configurations robots are initially positioned in different random locations. Unlike our experiments in the previous chapter robots are not limited by capacity constraints. The removal of capacity constraints allows us to consider the full influence of each clustering technique on the task allocation without the results being artificially restricted. We also include standard SSI auctions without capacity constraints as a control variable. Finally, we note that robots always bid with the true path cost for tasks. The SLD metric is only used for cluster formation. We present the mean results of the maximum distance and the summation of all distances travelled for the two team objectives in Tables 4.1, 4.2, 4.3 and 4.4.

4.3.1 MiniMax

The results for SSC auctions with robots bidding according to the MiniMax team objective are shown for clusters formed with $k = \frac{1}{2}|T|$ in Table 4.1 and for clusters formed with $k = \frac{2}{3}|T|$ in Table 4.2. Both of these result tables show that, unsurprisingly, the use of a TPD metric results in task allocations that have lower mean maximum robot travel distances. For the SLD metric complete linkage clustering produces the best solutions and *K*-medoids clustering performs the worst. When the TPD metric is applied the costs for each clustering algorithm substantially improves. In particular, complete linkage clustering with $k = \frac{2}{3}|T|$ on average produces lower costs than standard SSI auctions. We also note, that as the number of robots increases the maximum distance travelled decreases regardless of the cluster formation technique used. The results also show that when there are more clusters ($k = \frac{2}{3}|T|$) the maximum distance travelled by the robots is the smallest. These two observations contradict our results in the previous chapter (Table 3.2), in particular, previously the number of clusters was inconsequential to the results. We suspect that this change in result is due to the removal of the capacity constraint.

Statistical Validity

To confirm the statistical validity of these results we perform *non-parametric one-sided Wilcoxon signed-rank tests* for each robot/task/cluster combination. We choose this statistical test as we cannot make distribution assumptions due to the differences in robot initial locations and the map configurations of opened and closed doors for each exper-

Table 4.1: Comparison of Clustering Techniques for the MiniMax Team Objective with $k = \frac{1}{2}|T|$ (percentage improvement compared to the SLD metric in brackets).

				Straight Line Distance Metric				True Path Distance Metric			
Robots	Tasks	Clusters	SSI Control	*K*-means	*K*-medoids	Single linkage	Complete linkage	*K*-means	*K*-medoids	Single linkage	Complete linkage
4	16	8	826	1013	1102	962	941	914 (9.7%)	940 (14.7%)	887 (7.8%)	870 (7.5%)
6	24	12	743	993	908	960	934	818 (17.6%)	858 (5.6%)	785 (18.3%)	747 (20.1%)
8	32	16	705	892	966	925	827	776 (13.0%)	808 (16.3%)	698 (24.5%)	689 (16.8%)
10	40	20	616	819	905	865	806	682 (16.7%)	692 (23.6%)	642 (25.8%)	624 (22.6%)
4	20	10	918	1097	1180	1093	1056	988 (9.9%)	1060 (10.2%)	945 (13.6%)	929 (12.0%)
6	30	15	840	1012	1063	1034	978	895 (11.6%)	931 (12.4%)	851 (17.7%)	824 (15.7%)
8	40	20	766	901	985	925	883	807 (10.4%)	819 (16.8%)	788 (14.7%)	754 (14.6%)
10	50	25	665	815	918	876	848	742 (8.9%)	740 (19.4%)	685 (21.9%)	677 (20.1%)
4	24	12	1026	1213	1231	1173	1131	1054 (13.1%)	1097 (10.9%)	1042 (11.2%)	1010 (10.7%)
6	36	18	886	1122	1050	1088	1016	966 (14.0%)	942 (10.4%)	918 (15.6%)	871 (14.2%)
8	48	24	776	993	1006	988	967	869 (12.5%)	873 (13.2%)	817 (17.4%)	792 (18.0%)
10	60	30	707	888	961	877	841	815 (8.3%)	788 (18.0%)	696 (20.6%)	724 (13.9%)
Combined Mean:			789	980	1023	981	936	860 (12.2%)	879 (14.1%)	813 (17.1%)	793 (15.3%)

Table 4.2: Comparison of Clustering Techniques for the MiniMax Team Objective with $k = \frac{2}{3}|T|$ (percentage improvement compared to SLD metric in brackets).

				Straight Line Distance Metric				True Path Distance Metric			
Robots	Tasks	Clusters	SSI Control	*K*-means	*K*-medoids	Single linkage	Complete linkage	*K*-means	*K*-medoids	Single linkage	Complete linkage
4	16	11	826	913	945	882	877	865 (5.3%)	894 (5.4%)	878 (0.4%)	848 (3.3%)
6	24	16	743	822	888	871	850	758 (7.8%)	804 (9.5%)	727 (16.5%)	741 (12.8%)
8	32	21	705	827	862	841	824	728 (11.9%)	735 (14.7%)	680 (19.2%)	684 (17.0%)
10	40	27	616	725	773	744	711	665 (8.3%)	643 (16.9%)	618 (17.0%)	623 (12.4%)
4	20	13	918	999	1079	1039	1042	990 (1.0%)	991 (8.2%)	921 (11.4%)	907 (13.0%)
6	30	20	840	930	949	940	952	853 (8.2%)	846 (10.9%)	827 (12.0%)	818 (14.1%)
8	40	27	766	821	864	831	806	760 (7.4%)	762 (11.8%)	737 (11.3%)	741 (8.0%)
10	50	33	665	764	815	754	749	704 (7.8%)	719 (11.9%)	658 (12.8%)	665 (11.2%)
4	24	16	1026	1028	1148	1152	1123	1033 (-0.4%)	1088 (5.2%)	981 (14.8%)	1007 (10.3%)
6	36	24	886	954	1042	971	951	935 (2.0%)	932 (10.5%)	887 (8.6%)	874 (8.1%)
8	48	32	776	890	923	862	824	837 (6.0%)	838 (9.2%)	784 (9.1%)	778 (5.6%)
10	60	40	707	863	879	747	762	730 (15.3%)	741 (15.7%)	709 (5.2%)	720 (5.6%)
Combined Mean:			789	878	931	886	873	821 (6.4%)	833 (10.5%)	784 (11.6%)	784 (10.2%)

iment. We seek to confirm that the use of a TPD metric in cluster formation results in lower final travel distances than clusters formed using a SLD metric. Our null hypothesis is defined as:

$$H_0 : \mu\lambda_{\text{TPD}} \geq \mu\lambda_{\text{SLD}}$$

and our alternative hypothesis as:

$$H_1 : \mu\lambda_{\text{TPD}} < \mu\lambda_{\text{SLD}}$$

As an aside, it is important to note that a sample size of 25 configurations is generally considered small for significance testing. However, to draw conclusions across the full set of experimental data we perform tests of significance independently for each of our 12 robot/task/cluster permutations and 4 clustering algorithms. We then discuss our results for each clustering algorithm in the context of the results of these 12 significance tests. This approach to significance testing has been used previously for comparisons of SSI-like auction algorithms [120].

For *K*-means clustering with $k = \frac{1}{2}|T|$ we get a significant result with confidence greater than 0.95 that the costs generated with the TPD metric are lower than the SLD metric for all robot/task/cluster combinations tested, except for configurations with 10 robots and 50 or 60 tasks. For *K*-means clustering with $k = \frac{2}{3}|T|$ only half of our results are significant. The most extreme situation of non-significance is our mean results for the configuration of 4 robots, 24 tasks, and 16 clusters which has true path distances resulting in a higher maximum distance travelled than the use of a SLD metric.

Using *K*-medoids clustering we get significant results for all but one robot/task/cluster combination (6 robots, 24 tasks, 12 clusters), although, in both metrics *K*-medoids clustering generated the most costly robot task allocations. We speculate that the cause of this non-significant result is due to centroid-based clustering seeking to form non-overlapping clusters of geographically close tasks, whereas robots, in seeking to minimise their path travelled, may not confine themselves to local geographic areas.

Our results for agglomerative clustering have much stronger statistical significance. Using single linkage criteria, all robot/task/cluster combinations (except 4 robots, 16 tasks, 11 clusters, $k = \frac{2}{3}|T|$) obtain confidence 0.97 and greater for both $k = \frac{1}{2}|T|$ and $k = \frac{2}{3}|T|$. Using complete linkage criteria we obtain confidence levels of at least 0.99 for $k = \frac{1}{2}|T|$. We obtain slightly weaker confidence levels of at least 0.95 for most robot/task/cluster combinations with $k = \frac{2}{3}|T|$, and in two combinations (4 robots, 16 tasks, 11 clusters and 10 robots, 60 tasks, 40 clusters) our results are not significant. Overall, we can conclude that using TPD metrics for cluster formation in SSC auctions produces much better solutions to the multi-robot task allocation problem than straight

line distances for the MiniMax team objective.

SSI Control Comparison

A surprise in the result data for agglomerative clustering using the TPD metric was in certain robot/task/cluster combinations and overall for $k = \frac{2}{3}|T|$ the mean results were lower than the SSI auction control results. These results are of interest as few SSI auction extensions have outperformed standard SSI auctions without capacity constraints (cf. Section 2.4.4). Although, as not all mean results are lower than the SSI control results, we cannot conclude that our technique produces better task allocation solutions than SSI. However, we can use a *non-parametric two-sided Wilcoxon signed-rank tests* for each robot/task/cluster combination to see if our results are statistically equivalent. For this test, we define our null hypothesis as:

$$H_0 : \mu\lambda_{\text{agglomerative}} \neq \mu\lambda_{\text{SSI}}$$

and our alternative hypothesis as:

$$H_1 : \mu\lambda_{\text{agglomerative}} = \mu\lambda_{\text{SSI}}$$

In all combinations (except single linkage, 4 robots, 16 tasks, 8 clusters, $k = \frac{1}{2}|T|$), for both single linkage and complete linkage criteria and for $k = \frac{1}{2}|T|$ and $k = \frac{2}{3}|T|$, we obtain confidence levels that support our alternative hypothesis. We then tested the hypothesis that SSC auctions with agglomerative clustering produces lower mean results than standard SSI auctions:

$$H_2 : \mu\lambda_{\text{agglomerative}} < \mu\lambda_{\text{SSI}}$$

however, no support for this was found in our results. Finally, we tested the opposite hypothesis that standard SSI auctions produce smaller results than SSC auctions with agglomerative clustering:

$$H_3 : \mu\lambda_{\text{agglomerative}} > \mu\lambda_{\text{SSI}}$$

In this test, three robot/task/cluster combinations (single linkage with $k = \frac{1}{2}|T|$, 4 robots, 16 tasks, 8 clusters, and 8 robots, 48 tasks, 24 clusters and complete linkage with $k = \frac{2}{3}|T|$, 10 robots, 60 tasks, 40 clusters) showed significant outcomes. From this we can strongly conclude that when using agglomerative clustering with a TPD metric in SSC auctions for the MiniMax team objective the results are statistically equivalent to SSI auctions.

4.3.2 MiniSum

Our results for robots bidding according to the MiniSum team objective are shown for $k = \frac{1}{2}|T|$ in Table 4.3 and for $k = \frac{2}{3}|T|$ in Table 4.4. This data also shows clusters formed using a TPD metric produce the best results. In particular, agglomerative clustering with a TPD metric on average produces lower costs than standard SSI auctions in the majority of robot/task/cluster combinations tested. The SLD metric data also continues the trend shown previously in the MiniMax team objective data that the greater the number of clusters ($k = \frac{2}{3}|T|$) the lower the overall cost. However, this observation isn't supported in the TPD metric data. This instead, shows no advantage between either cluster size tested.

In considering the trade-off between the two metrics, single linkage clustering produces the lowest average costs, independent of the metric used. It also has the smallest improvement between the two metrics. This is also a reversal of the previous MiniMax team objective results. Furthermore, the percentage improvement of all the clustering algorithms using TPD metrics compared to SLD metrics is also much smaller in these results. Overall, *K*-medoids shows the largest improvement in the use of the TPD metric in comparison to the SLD metric, despite this, for both metrics it generates the most costly allocations.

Statistical Validity

Again we perform one-sided Wilcoxon signed-rank tests to compare the advantage of the TPD metric over the SLD metric. We define our null hypothesis again as:

$$H_0 : \mu\lambda_{\text{TPD}} \geq \mu\lambda_{\text{SLD}}$$

and alternative hypothesis as:

$$H_1 : \mu\lambda_{\text{TPD}} < \mu\lambda_{\text{SLD}}$$

For *K*-means clustering with $k = \frac{1}{2}|T|$ we confirm a very significant result that the total team cost using the TPD metric is lower than the cost using the SLD metric with confidence 0.99. When tested with clusters of $k = \frac{2}{3}|T|$ we generally get significant results greater than 0.95, however, in three combinations tested this doesn't hold (10 robots, 40 tasks, 27 clusters, and 8 robots, 40 tasks, 27 clusters, and 4 robots, 24 tasks, 16 clusters). While this is an improvement over the results for the MiniMax team objective (where half the results were non-significant), a quarter of robot/task/clusters combinations generating non-significant results is high, and this is despite our overall means showing the

use of a TPD metric outperforming a SLD metric in all robot/task/cluster combinations. For *K*-medoids clustering we get significance results at 0.95 confidence for all but one robot/task/cluster combination tested (6 robots, 36 tasks, 24 clusters).

Using agglomerative clusters we again get strong statistically significant results. For single linkage clustering, clusters formed with $k = \frac{1}{2}|T|$ measure a confidence result of 0.98 and for $k = \frac{2}{3}|T|$ generally a confidence of at least 0.95. With $k = \frac{2}{3}|T|$ we did have two non-significant results for 4 robots, 16 tasks, 11 clusters and 10 robots, 60 tasks, 40 clusters, however, in these combinations the results with the SLD metric are already of good cost. For complete linkage clustering with $k = \frac{1}{2}|T|$ the confidence is at least 0.999 and with $k = \frac{2}{3}|T|$ 0.95 across all robot/task/cluster combinations with no exceptions.

SSI Control Comparison

Over half our mean results for agglomerative clustering with the TPD distance metric show lower costs than SSI control auctions. We again use a non-parametric two-sided Wilcoxon signed-rank tests for each robot/task/cluster combination to see if our results are statistically equivalent. The null hypothesis is defined as:

$$H_0 : \mu\lambda_{\text{agglomerative}} \neq \mu\lambda_{\text{SSI}}$$

and alternative hypothesis as:

$$H_1 : \mu\lambda_{\text{agglomerative}} = \mu\lambda_{\text{SSI}}$$

In all but one combination, for both single linkage and complete linkage criteria, we obtain confidence levels that demonstrate that our results are statistically equivalent. Furthermore, the non-significant result was for complete linkage clustering with $k = \frac{2}{3}|T|$, 4 robots, 20 tasks, 13 clusters. In this result, the mean cost for agglomerative clustering was $\mu\lambda_{\text{agglomerative}} = 2462$ and for the SSI auction $\mu\lambda_{\text{SSI}} = 2516$, *i.e.* our non-significant result was a consequence of our results being much lower than SSI auctions.

Additionally, testing the hypothesis that SSC auctions with agglomerative clustering produces lower mean results than standard SSI auctions:

$$H_2 : \mu\lambda_{\text{agglomerative}} < \mu\lambda_{\text{SSI}}$$

gives us significant results in five robot/task/combinations, two using single linkage clustering and three using complete linkage clustering. The reverse one-sided test:

$$H_3 : \mu\lambda_{\text{agglomerative}} > \mu\lambda_{\text{SSI}}$$

Table 4.3: Comparison of Clustering Techniques for the MiniSum Team Objective with $k = \frac{1}{2}|T|$ (percentage improvement compared to SLD metric in brackets).

				Straight Line Distance Metric				True Path Distance Metric			
Robots	Tasks	Clusters	SSI Control	*K*-means	*K*-medoids	Single linkage	Complete linkage	*K*-means	*K*-medoids	Single linkage	Complete linkage
4	16	8	2189	2324	2382	2246	2301	2212 (4.8%)	2285 (4.1%)	2155 (4.1%)	2154 (6.4%)
6	24	12	2516	2812	2769	2707	2735	2573 (8.5%)	2644 (4.5%)	2505 (7.5%)	2526 (7.7%)
8	32	16	2821	3119	3343	3061	3062	2897 (7.1%)	2957 (11.6%)	2820 (7.9%)	2814 (8.1%)
10	40	20	3015	3451	3518	3293	3329	3106 (10.0%)	3178 (9.7%)	3012 (8.6%)	2998 (9.9%)
4	20	10	2532	2715	2727	2635	2650	2499 (8.0%)	2603 (4.5%)	2473 (6.2%)	2442 (7.9%)
6	30	15	2887	3236	3291	3074	3155	2963 (8.4%)	3000 (8.8%)	2865 (6.8%)	2869 (9.1%)
8	40	20	3200	3621	3662	3499	3525	3318 (8.4%)	3370 (8.0%)	3218 (8.0%)	3207 (9.0%)
10	50	25	3435	3785	4106	3742	3745	3469 (8.3%)	3578 (12.9%)	3423 (8.5%)	3417 (8.7%)
4	24	12	2800	3036	2959	2873	2906	2791 (8.1%)	2864 (3.2%)	2734 (4.9%)	2733 (6.0%)
6	36	18	3216	3530	3545	3463	3511	3284 (7.0%)	3288 (7.3%)	3265 (5.7%)	3226 (8.1%)
8	48	24	3536	3924	4116	3808	3855	3604 (8.2%)	3690 (10.4%)	3544 (6.9%)	3502 (9.2%)
10	60	30	3826	4203	4402	4051	4036	3877 (7.7%)	3939 (10.5%)	3797 (6.3%)	3772 (6.5%)
Combined Mean:			2998	3313	3402	3204	3234	3050 (8.0%)	3116 (8.4%)	2984 (6.9%)	2972 (8.1%)

Table 4.4: Comparison of Clustering Techniques for the MiniSum Team Objective with $k = \frac{2}{3}|T|$ (percentage improvement compared to SLD metric in brackets).

				Straight Line Distance Metric				True Path Distance Metric			
Robots	Tasks	Clusters	SSI Control	*K*-means	*K*-medoids	Single linkage	Complete linkage	*K*-means	*K*-medoids	Single linkage	Complete linkage
4	16	11	2189	2262	2283	2197	2197	2156 (4.7%)	2205 (3.5%)	2135 (2.8%)	2128 (3.1%)
6	24	16	2516	2655	2735	2646	2650	2541 (4.3%)	2571 (6.0%)	2514 (5.0%)	2506 (5.4%)
8	32	21	2821	3080	3121	3002	3006	2913 (5.4%)	2913 (6.7%)	2812 (6.3%)	2819 (6.2%)
10	40	27	3015	3129	3350	3182	3152	3052 (2.5%)	3111 (7.1%)	3010 (5.4%)	3003 (4.7%)
4	20	13	2532	2640	2669	2570	2566	2506 (5.1%)	2553 (4.3%)	2475 (3.7%)	2462 (4.1%)
6	30	20	2887	3104	3089	3054	3062	2917 (6.0%)	2936 (5.0%)	2887 (5.5%)	2888 (5.7%)
8	40	27	3200	3349	3564	3407	3355	3245 (3.1%)	3299 (7.5%)	3234 (5.1%)	3227 (3.8%)
10	50	33	3435	3640	3838	3509	3570	3453 (5.2%)	3482 (9.3%)	3385 (3.6%)	3401 (4.7%)
4	24	16	2800	2855	2959	2865	2911	2798 (2.0%)	2827 (4.5%)	2769 (3.4%)	2792 (4.1%)
6	36	24	3216	3356	3446	3392	3408	3282 (2.2%)	3294 (4.4%)	3235 (4.6%)	3225 (5.4%)
8	48	32	3536	3796	3909	3639	3698	3539 (6.8%)	3575 (8.6%)	3352 (2.4%)	3537 (4.4%)
10	60	40	3826	4143	4097	3856	3901	3845 (7.2%)	3923 (4.3%)	3817 (1.0%)	3763 (3.5%)
Combined Mean:			2998	3167	3255	3110	3123	3021 (4.6%)	3057 (6.1%)	2985 (4.0%)	2979 (4.6%)

does not give us any significant results. From this we can comprehensively conclude that for the MiniSum team objective when using agglomerative clustering with a TPD metric in SSC auctions the results are at least statistically equivalent to SSI auctions.

4.4 Priority Allocation Based on Cluster Size

In previous sections, four straightforward algorithms for generating k clusters of tasks have been analysed with respect to their usefulness in SSC auctions. In each of these algorithms we have crude control over the total number of clusters k and by increasing or decreasing its value with respect to the number of tasks $|T|$ we can control the average number of tasks in each cluster. This, however, does not give us control over the maximum or minimum number of tasks in each cluster. Consequently, this lack of fine control over the number of tasks in each cluster can see a large variance in the number of tasks contained in large and small clusters. For instance, in a naïve extreme case, one cluster may have $|T| - k$ tasks and every other cluster will have exactly one task. Without additional information about the tasks or robots in the problem a general assumption can be made that the smaller clusters will have lower costs than larger clusters and therefore will often be allocated before larger clusters [11].

A primary purpose of allocating clusters of tasks is to exploit positive inter-task synergies that may otherwise not be considered in single task allocation routines. However, in situations where many small clusters are allocated first there may be large negative synergies between existing task allocations and a large unallocated task cluster. As such, instead of reducing the overall team costs one robot may be forced to complete many tasks while other robots are idle.

Consider, for example, a simple two robot, four task allocation problem given in Figure 4.13. We first generate three clusters: $C_1 = \{t_1, t_2\}$, $C_2 = \{t_3\}$ and $C_3 = \{t_4\}$ and a subsequent standard SSC auction using the MiniMax team objective is given in Table 4.5. The overall maximum cost to any one robot in this example is $\lambda_{r_2}(T_{r_2}) = 6$ which is twice the cost of the single task allocated to the other robot $\lambda_{r_1}(T_{r_1}) = 3$.

Let us now modify the bidding rules for SSC auctions such that robots may only submit bids for unallocated clusters containing the most tasks. Table 4.6 shows the results of running an SSC auction with this modified bidding rule. In the first round of bidding both robots now bid for cluster C_1 which contains two tasks and it is subsequently allocated to robot r_1 which immediately changes the task allocation compared to the previous example. In the two following auction rounds clusters C_2 and C_3 are allocated to robot r_2. The post-allocation overall maximum cost to any one robot in this example is $\lambda_{r_1}(T_{r_1}) = 5$ which is the optimal solution for this simple problem.

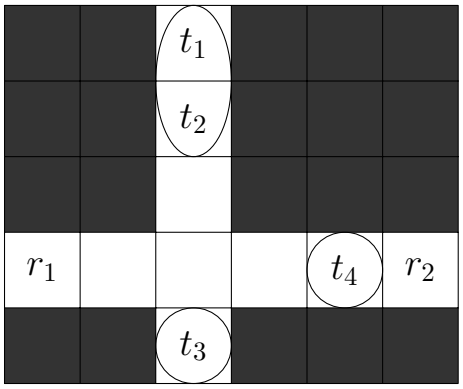


Figure 4.13: Priority Cluster Allocation Example.

Table 4.5: Standard SSC auction for example problem given in Figure 4.13 (winning bids and assignments in bold).

Round	r_1 Bids Calculated	r_1 Bid Submitted	r_2 Bids Calculated	r_2 Bid Submitted
1	$\beta_{r_1}^{C_1} \leftarrow \langle r_1, C_1, 5\rangle$ $\beta_{r_1}^{C_2} \leftarrow \langle r_1, C_2, 3\rangle$ $\beta_{r_1}^{C_3} \leftarrow \langle r_1, C_3, 4\rangle$	$\beta_{r_1}^{C_2}$	$\beta_{r_2}^{C_1} \leftarrow \langle r_2, C_1, 6\rangle$ $\beta_{r_2}^{C_2} \leftarrow \langle r_2, C_2, 4\rangle$ $\beta_{r_2}^{C_3} \leftarrow \langle r_2, C_3, 1\rangle$	$\boldsymbol{\beta_{r_2}^{C_3}}$
2		$\boldsymbol{\beta_{r_1}^{C_2}}$	$\beta_{r_2}^{C_1} \leftarrow \langle r_2, C_1, 6\rangle$ $\beta_{r_2}^{C_2} \leftarrow \langle r_2, C_2, 4\rangle$	$\beta_{r_2}^{C_2}$
3	$\beta_{r_1}^{C_1} \leftarrow \langle r_1, C_1, 7\rangle$	$\beta_{r_1}^{C_1}$		$\boldsymbol{\beta_{r_2}^{C_1}}$
$\lambda_{r_i}(T_{r_i})$		3		6

4.4.1 Empirical Experiments

To further explore the impact of large cluster priority bidding we ran empirical experiments with this modified bidding rule under the same set of conditions as the experiments in Section 4.3. In these experiments we use agglomerative clustering with a TPD metric as this was the previous best performing clustering approach and forms a baseline for further improvements. Additionally, we retain standard SSI auctions as a control variable. The results of these experiments for the MiniMax team objective are presented in Table 4.7 and for the MiniSum team objective in Table 4.8. Overall, these results show that priority bidding with the MiniMax team objective lowers the overall maximum team cost in the majority of robot/task/cluster combinations, and conversely, for the MiniSum team objective the costs increase.

Table 4.6: Modified bidding rule SSC auction for example problem given in Figure 4.13.

Round	r_1 Bids Calculated	r_1 Bid Submitted	r_2 Bids Calculated	r_2 Bid Submitted
1	$\beta_{r_1}^{C_1} \leftarrow \langle r_1, C_1, 5\rangle$ $\beta_{r_1}^{C_2} \leftarrow \langle r_1, C_2, 3\rangle$ $\beta_{r_1}^{C_3} \leftarrow \langle r_1, C_3, 4\rangle$	$\boldsymbol{\beta_{r_1}^{C_1}}$	$\beta_{r_2}^{C_1} \leftarrow \langle r_2, C_1, 6\rangle$ $\beta_{r_2}^{C_2} \leftarrow \langle r_2, C_2, 4\rangle$ $\beta_{r_2}^{C_3} \leftarrow \langle r_2, C_3, 1\rangle$	$\beta_{r_2}^{C_1}$
2	$\beta_{r_1}^{C_2} \leftarrow \langle r_1, C_2, 7\rangle$ $\beta_{r_1}^{C_3} \leftarrow \langle r_1, C_3, 9\rangle$	$\beta_{r_1}^{C_2}$		$\boldsymbol{\beta_{r_2}^{C_3}}$
3		$\beta_{r_1}^{C_2}$	$\beta_{r_2}^{C_2} \leftarrow \langle r_2, C_2, 4\rangle$	$\boldsymbol{\beta_{r_2}^{C_2}}$
$\lambda_{r_i}(T_{r_i})$		5		4

Table 4.7: Empirical Results for Priority Bidding for the MiniMax Team Objective (percentage improvement compared to previous results in brackets).

			Single Linkage Clustering		Complete Linkage Clustering	
Robots	Tasks	SSI	$k = \frac{1}{2}\|T\|$	$k = \frac{2}{3}\|T\|$	$k = \frac{1}{2}\|T\|$	$k = \frac{2}{3}\|T\|$
4	16	826	814 (8.2%)	834 (5.1%)	854 (1.9%)	832 (1.9%)
6	24	743	688 (12.3%)	706 (2.9%)	712 (4.6%)	695 (6.3%)
8	32	705	679 (2.7%)	675 (0.8%)	662 (3.8%)	665 (2.7%)
10	40	616	589 (8.2%)	586 (5.2%)	584 (6.4%)	586 (5.9%)
4	20	918	921 (2.6%)	924 (-0.3%)	930 (-0.1%)	947 (-4.4%)
6	30	840	794 (6.6%)	830 (-0.4%)	795 (3.4%)	798 (2.5%)
8	40	766	686 (13.0%)	698 (5.3%)	672 (11.0%)	689 (7.1%)
10	50	665	615 (10.1%)	608 (7.6%)	643 (5.0%)	611 (8.2%)
4	24	1026	991 (4.9%)	984 (-0.3%)	953 (5.7%)	1010 (-0.3%)
6	36	886	878 (4.4%)	881 (0.7%)	864 (0.9%)	885 (-1.2%)
8	48	776	743 (9.1%)	760 (3.0%)	727 (8.2%)	741 (4.7%)
10	60	707	673 (3.3%)	671 (5.4%)	662 (8.6%)	676 (6.1%)
Combined Mean:		789	756 (7.0%)	763 (2.7%)	755 (4.8%)	761 (2.9%)

Statistical Validity

We now consider the statistical significance of the MiniMax team objective results relative to the SSI control auction results. The results earlier in this chapter showed that agglomerative clustering with a TPD metric was statistically equivalent to standard SSI auctions (Sections 4.3.1 and 4.3.2). In these latest results the MiniMax team objective costs have further decreased relative to the SSI control auction results. In particular, all results for six or more robots have lower mean costs than standard SSI auctions.

We again seek to validate the hypothesis that SSC auctions with agglomerative clustering produces lower mean results than standard SSI auctions. For this test we set up a one-sided Wilcoxon signed-rank test for each robot/task/cluster combination as follows.

Table 4.8: Empirical Results for Priority Bidding for the MiniSum Team Objective.

			Single Linkage Clustering		Complete Linkage Clustering	
Robots	Tasks	SSI	$k = \frac{1}{2}\|T\|$	$k = \frac{2}{3}\|T\|$	$k = \frac{1}{2}\|T\|$	$k = \frac{2}{3}\|T\|$
4	16	2189	2191 (-1.7%)	2209 (-3.5%)	2170 (-0.8%)	2219 (-4.3%)
6	24	2516	2543 (-1.5%)	2542 (-1.1%)	2522 (0.1%)	2541 (-1.4%)
8	32	2821	2875 (-2.0%)	2867 (-2.0%)	2845 (-1.1%)	2832 (-0.5%)
10	40	3015	3063 (-1.7%)	3009 (0.0%)	2994 (0.1%)	3018 (-0.5%)
4	20	2532	2545 (-2.9%)	2528 (-2.2%)	2539 (-4.0%)	2542 (-3.2%)
6	30	2887	2932 (-2.3%)	2930 (-1.5%)	2930 (-2.2%)	2928 (-1.4%)
8	40	3200	3251 (-1.0%)	3271 (-1.1%)	3202 (0.2%)	3251 (-0.8%)
10	50	3435	3470 (-1.4%)	3447 (-1.8%)	3446 (-0.8%)	3465 (-1.9%)
4	24	2800	2797 (-2.3%)	2809 (-1.4%)	2798 (-2.4%)	2800 (-0.3%)
6	36	3216	3290 (-0.8%)	3278 (-1.3%)	3285 (-1.8%)	3295 (-2.2%)
8	48	3536	3661 (-3.3%)	3603 (-1.4%)	3574 (-2.1%)	3641 (-2.9%)
10	60	3826	3874 (-2.0%)	3922 (-2.7%)	3907 (-3.6%)	3888 (-3.3%)
Combined Mean:		2998	3041 (-1.9%)	3035 (-1.7%)	3018 (-1.6%)	3035 (-1.9%)

The null hypothesis is defined as:

$$H_0 : \mu\lambda_{\text{agglomerative}} \geq \mu\lambda_{\text{SSI}}$$

and alternative hypothesis as:

$$H_1 : \mu\lambda_{\text{agglomerative}} < \mu\lambda_{\text{SSI}}$$

In all but two[1] significance test results for combinations with 8 robots and for all combinations with 10 robots we get confidence levels greater than 0.90. This result demonstrates that allocating large clusters of tasks before smaller clusters for the MiniMax team objective with a large number of robots produces lower mean maximum robot costs than standard SSI auctions. Furthermore, these results and conclusion are a reflection of and further confirmation of our initial results in Chapter 3. In particular, earlier plots (Figures 3.5, 3.6, and 3.7) compared the effect of number of tasks, robots, and capacity constraints and clearly showed that the number of robots is proportional to the relative improvement of SSC auctions over standard SSI auctions.

[1]Single-linkage $k = \frac{1}{2}|T|$, 8 robots, 32 tasks and single-linkage $k = \frac{2}{3}|T|$, 8 robots, 48 tasks.

4.5 Discussion

A major observation throughout this chapter's experiments is that agglomerative clustering consistently produces lower MRTA solution costs than centroid-based clustering. This holds true even for complete linkage clustering which produces dense clusters of tasks similar to *K*-means clustering. To provide a possible explanation of the advantages of using agglomerative clustering in solving MRTA problems with SSC auctions we need to consider the structure of the clusters that are formed by each algorithm.

4.5.1 Overlapping Clusters

A key difference between centroid-based clustering and agglomerative clustering is the pairing of tasks. In centroid-based clustering each task cluster is an exclusive non-overlapping geographic area of tasks. While in some situations this is a good approach, e.g. forming a single cluster of a set of tasks in a far corner of the environment. In other situations it can prove problematic. For instance, consider a cluster of evenly spaced tasks. In this situation a robot may need to travel to one far edge of the cluster and then work inwards around the perimeter of the cluster to complete all tasks in the shortest path. In comparison agglomerative clustering allows task clusters to overlap and is more likely to reflect paths that robots are likely to travel along.

An example problem that generates overlapping clusters with complete linkage clustering is given in Figure 4.14. Beginning with six single-item clusters:

$$C_1 = \{t_1\}$$
$$C_2 = \{t_2\}$$
$$C_3 = \{t_3\}$$
$$C_4 = \{t_4\}$$
$$C_5 = \{t_5\}$$
$$C_6 = \{t_6\}$$

The subsequent merge sequence to form two clusters is:

$$C_7 \leftarrow C_4 \cup C_5 = \{t_4, t_5\}$$
$$C_8 \leftarrow C_7 \cup C_3 = \{t_4, t_5, t_3\}$$
$$C_9 \leftarrow C_8 \cup C_2 = \{t_4, t_5, t_3, t_2\}$$
$$C_{10} \leftarrow C_1 \cup C_6 = \{t_1, t_6\}$$

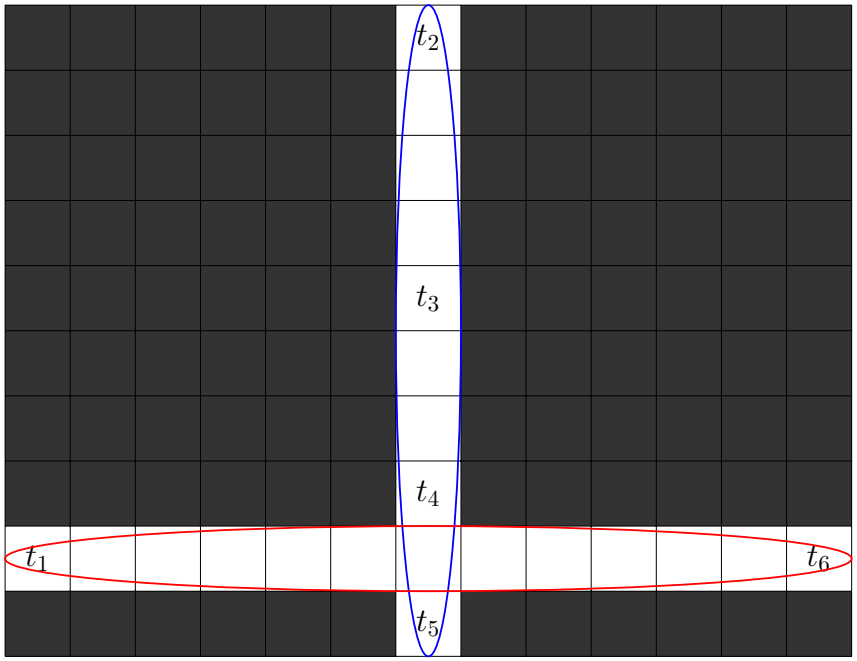


Figure 4.14: Overlapping Clusters Example - Complete Linkage Clustering.

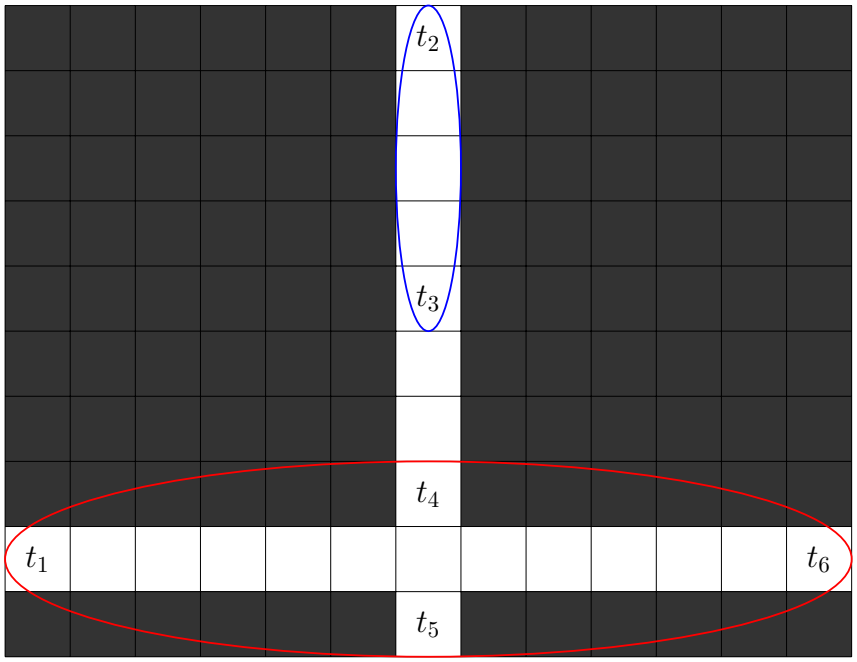


Figure 4.15: Overlapping Clusters Example - K-means Clustering.

The final two clusters generated by complete linkage clustering are C_9 and C_{10}. If we consider the shape of these clusters, C_9 travels exclusively in the vertical direction and C_{10} in the horizontal, and the clusters cross over in between tasks t_4 and t_5. In comparison, if we generate two clusters with *K*-means clustering, for this example, the stable clusters formed are $C_1 = \{t_1, t_4, t_5, t_6\}$ and $C_2 = \{t_2, t_3\}$ (Figure 4.15). The differences in the shape of these clusters can clearly be seen through this simple example and the effects of this on task allocation and completion using SSC auctions are large.

4.5.2 Determining k

An additional component influencing cluster formation in centroid-based clustering is the choice of the initial k central vectors. Our approach is to randomly select k tasks to use as initial cluster central vectors, while other approaches include selecting initial central vectors that are representative of the distribution and density of the data [33], placing central vectors at the far edges of the environment [116], or using agglomerative clustering to form initial clusters [73]. The efficiency of centroid-based clustering algorithms can be heavily influenced by the selection of initial central vectors. Bad initial central vectors may require many iterations to find stable clusters or they may also become constrained by local minima [71, 47, 50]. However, avoiding these problems through naïve approaches, such as, repeated cluster formations with different initial central vectors [116], or including additional knowledge of the usage of the data set to be clustered [19] introduces a trade-off between performance and efficiency.

Determining the ideal number of clusters k to generate is a problem in both centroid-based and agglomerative clustering. A large k value results in clusters that contain few tasks and, as such, little inter-task synergy is considered. On the other hand, small k value results in clusters containing many tasks and can lead to robots being allocated tasks and resultant paths that would be better suited to other robots. A clustering approach that sought a balance between these two challenges would be ideal, however, the time complexity may be much greater than our existing approaches.

Common approaches in related literature for determining good k values include using statistical variances to compare clusters generated by different k values [60, 89, 121, 28], minimising cluster silhouettes [91], minimising intra-cluster distortion [118], and splitting large and removing small clusters [3]. There are also clustering algorithms that don't require an initial value for k (e.g. DBSCAN [32] and OPTICS [2]), however, these algorithms require other configuration parameters, such as, maximum inter-task distances. Furthermore, determining an ideal number of clusters may not avoid the problems of local minima [116]. Throughout our experiments we have used values of k that seek to generate an average cluster size of 1.5 - 2 tasks per cluster. The reasoning behind our approach is to

generate many small clusters that have high internal inter-task synergies, but, also retain enough clusters that the auction process considers additional inter-task synergies that the clustering process misses.

An alternative clustering approach, fuzzy clustering [132] (e.g. Fuzzy c-means [29]) in which tasks may belong to multiple clusters may be of interest for future work. For instance, any one single cluster is assigned to a robot, all tasks in this cluster with cluster membership scores above a certain threshold are allocated to the robot and removed from the remaining unallocated clusters. This approach is similar to SSI auctions with bundle bids in that individual robots may be bidding on different clusters that both contain the same task t and $1 \leq i \leq |C|$ tasks are allocated to a robot in any given round. However, unlike SSI auctions with bundle bids, during each auction round only one robot is awarded tasks instead of multiple robots.

4.5.3 Cluster Size

Closely related to determining the number of clusters is limiting the total number of tasks in each cluster. The purpose for such a limit is to enforce domain constraints [19], for instance, if we use capacity constraints it makes no sense for the number of tasks in a cluster to exceed a single robot's capacity constraint. This restriction is easy to apply to agglomerative clustering. However, these restrictions do not guarantee good cluster solutions. For instance, it is possible that two clusters at extreme opposite ends of an environment will be merged, if all intermediate task clusters are at capacity. In centroid-based clustering task size limits can be applied by splitting large clusters, or assigning tasks that are far from the central vector to alternative clusters. Finally, domain constraints can be enforced through the use of semi-supervised clustering. For instance, Bilenko, Basu, and Mooney take this approach in the development of *MPCK-Means clustering* which incorporates constraints and learning into *K*-means clustering [9].

Of particular interest in the application of SSC auctions for solving MRTA problems is understanding why priority allocation of large task clusters lowers the overall team costs for the MiniMax team objective but not for the MiniSum team objective. A possible contributing factor is the range for improvement relative to the optimal solution. Previously, standard SSI auctions with the MiniSum team objective have been shown to generate solutions within a factor of 1.10 of optimal and therefore there is little room for improvement. In comparison, the MiniMax team objective has been shown to be a factor of 1.44 away from the optimal and therefore has a much greater range of potential improvement [122]. Given our statistical results show that SSC auctions using agglomerative clustering are equivalent to SSI auctions we assume that these factors are comparable for SSC auctions.

Furthermore, in previous work on SSI auctions the allocation of tasks with high costs to all robots first has been considered for the MiniMax team objective. It is argued that this approach performs better than iteratively assigning tasks with the smallest cost as the remaining lower cost tasks can be used to balance the allocation and execution time across the robots [58]. Large cluster priority bidding can be compared to this high cost bidding as both approaches are similar. Clusters that generally would be assigned in later auction rounds are assigned first and our results show this improves the overall team cost. This approach, however, does not hold for the MiniSum team objective as, unlike the MiniMax team objective, balancing task allocations and running times evenly across the robots will not minimise the total distance travelled. Instead, the MiniSum team objective works by iteratively adding the tasks with the smallest costs which in some instances can see large imbalances in the task loads between individual robots. This imbalance can be best addressed through the application of capacity constraints.

4.6 Summary

In this chapter, we have presented an analysis of clustering techniques to solve the multi-robot task allocation problem using SSC auctions. We have considered the time required using two different clustering approaches and the effect of calculating true path distances instead of straight line distances in the formation of clusters. In summary, for both MiniMax and MiniSum team objectives tested, we have shown the power of using a TPD metric in the formation of clusters to solve the multi-robot task allocation problem. Despite our cluster formation time measurements showing that the use of TPD metrics are around 100 times slower than straight line calculations, we believe that the overall travel distance saved as a result of better clusters far outweighs the extra initial time spent on cluster formation. Furthermore, it can be argued that the cluster formation time with a TPD metric will be much smaller than the expected execution time of the robots completing all allocated tasks.

Our empirical results show the benefit of using agglomerative clustering with a TPD metric over other cluster formation techniques. We applied statistical significance testing to our results and showed that for SSC auctions with agglomerative clustering produce on average statistically equivalent solutions to MRTA problems without capacity constraints. Additionally, we considered the priority allocation of large clusters to robots, which for the MiniMax team objective resulted in statistical significant lower costs than standard SSI auctions.

In the next chapter we apply single linkage clustering to an extended MRTA problem with tasks requiring collection and delivery. Auctioning clusters of tasks in this problem domain is much more difficult as we are required to consider both the collection and delivery locations of objects when forming clusters. Furthermore, we consider the insertion of additional tasks into a running system, and the failure of robots during task execution.

Chapter 5

Task Allocation with Collection and Delivery in Dynamic Environments

Consider a team of autonomous mobile robots operating as courier delivery vehicles in a large office-like environment. Each robot is required to collect from and deliver parcels to a variety of locations around the office building. Robots may be constrained to a fixed capacity in the number of parcels they may carry at any one time and, after a parcel is picked up, it can only be delivered to its intended destination. Our goal is for the robots to be allocated and deliver all parcels as effectively and efficiently as possible.

This problem is an extension of the standard MRTA problem and, as discussed in Chapter 2, there are many existing approaches for solving this class of problem. However, most existing techniques require that all tasks are static and known before allocation. In many real-world situations, additional tasks are dynamically discovered during execution. Furthermore, in real-world scenarios it is likely that, over time, robots will fail requiring their tasks to be reassigned to other operating robots.

In this chapter we consider this extension of the MRTA problem in a dynamic domain. First, we consider the application of clustering to tasks with collection and delivery (Section 5.1). Second, we describe two dynamic scenarios requiring task reallocation: *dynamic task insertion* (Section 5.2) and *handling robot failure* (Section 5.3). In both scenarios, we evaluate the performance and execution trade-off between reallocating a subset of the total tasks and all uncompleted tasks. As both scenarios require individual robots to commit to additional tasks, the problems of inserting tasks and removing robots appear to be equivalent problems (see Theorem 2.1). However, our empirical results for these two problems differ substantially.

Portions of the research in this chapter have been published in *Repeated Sequential Single-Cluster Auctions with Dynamic Tasks for Multi-Robot Task Allocation with Pickup and Delivery* (MATES-13) and *Repeated Auctions for Reallocation of Tasks with Pickup and Delivery Upon Robot Failure* (PRIMA-13).

5.1 MRTA Problems with Collection and Delivery

MRTA problems with tasks requiring collection and delivery are an extension to the standard MRTA problem. In the standard MRTA definition, each task is a single point in the environment (cf. Section 2.2.1). For tasks with collection and delivery, we define each task t as a tuple $t = \langle p_c, p_d \rangle$ of a collection location p_c and a delivery location p_d. A robot is considered to be executing a task once it has visited its collection location and until the point it has reached its delivery location. Robots may also have capacity constraints in the number of tasks they are able to execute at any moment in time. However, unlike the capacity constraints used in previous chapters, each robot may complete any number of tasks. Once all initial tasks are allocated to robots, each robot begins executing the tasks that have been assigned to it. As robots complete each individual task $t_{\text{complete}} \in T$, by arriving at the delivery location p_d, the completed task is removed from the set of tasks:

$$T \leftarrow T \backslash \{t_{\text{complete}}\}$$

5.1.1 Clustering Tasks with Collection and Delivery

The formation of clusters of tasks that have both collection and delivery locations is much more difficult than cluster formation with tasks that are simply single point locations. In developing our clustering approach, we considered three different approaches for generating a metric that describes each task. First, we attempted to cluster based on the midpoint of the line segment between the collection and delivery location. However, this metric lacks any information about where either end point is located and could result in clusters that may have geographically close mid points but large distances between end points.

Second, we attempted to cluster by forming vectors of collection and delivery locations and chaining together tasks that have a delivery location close to the collection location of another task. This approach to clustering closely matches the bidding pattern of robots in auctions, and as a result, clustering would provide no benefit.

Our third, and successful, approach was to cluster the collection and delivery locations of tasks separately. First, we form clusters of tasks based on collection locations (Figure 5.1). Second, within each collection location cluster we cluster again into smaller clusters based on delivery locations (Figure 5.2). The complete set of all small clusters is then used by robots in bidding.

We formulate this algorithm in Figure 5.3. We begin with no final clusters set (Line 1). We initially cluster all tasks based on collection location (Line 3). We then iterate over each of these collection location clusters (Line 5). Within each collection location cluster we cluster again based on delivery location (Line 6). Finally, we merge each set

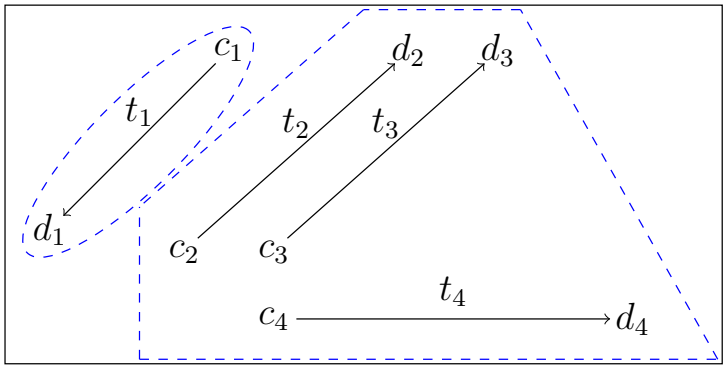


Figure 5.1: Four tasks clustered into two collection task clusters.

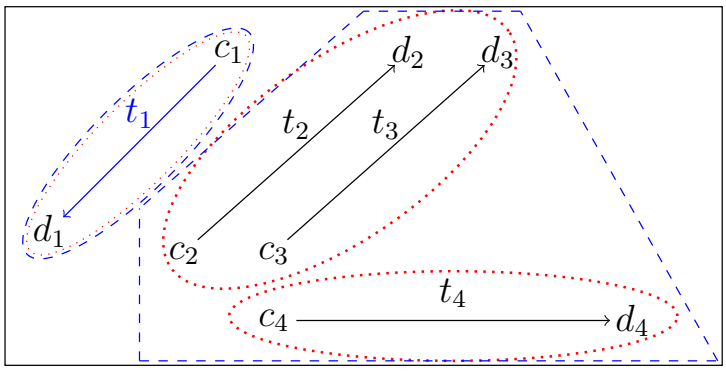


Figure 5.2: Formation of three final task clusters based on delivery locations.

function TwoStepClustering (T, k_c, k_d)
Input: T: the set of tasks to be clustered
k_c: the number of collection clusters to be formed
k_d: the number of delivery clusters to be formed per collection cluster
Output: K: the set of clusters for auction

1: $K = \emptyset$;
2: */* Collection location clustering */*
3: $CollectionClusters \leftarrow$ CalcCollectionClusters(T,k_c);
4: */* Delivery location clustering */*
5: **for each** collection cluster $C_c \in CollectionClusters$
6: $DeliveryClusters \leftarrow$ CalcDeliveryClusters(C_c,k_d);
7: $K \leftarrow K \cup \{DeliveryClusters\}$;

Figure 5.3: Algorithm for clustering of tasks with collection and delivery.

of delivery location clusters into the final set of clusters for auction (Line 7). As both clustering functions CalcCollectionClusters and CalcDeliveryClusters consider only one side of a task's location this allows us to use any existing clustering algorithm that clusters based on point locations, e.g. *K*-means clustering or single-linkage clustering.

In our algorithm we need two k values k_c and k_d; one for each clustering function. For our experiments we seek to find k overall clusters for auction such that:

$$k = k_c * k_d$$

This ensures that each collection cluster is split into equal numbers of delivery clusters. However a problem can arise, due to each cluster containing a varying number of tasks, when considering the clustering of delivery locations inside a collection cluster. If the number of tasks in the cluster $|p|$ is less than k_d we can only form $|p|$ clusters, and as a result, in total we will have less than k clusters. Without modifying the behaviour of the algorithm used to form the collection clusters we cannot prevent this occurring. As a result, in these situations we end up with fewer clusters than we initially sought.

To mitigate the effect of this during the formation of a large number of clusters we suggest a novel solution to gradually increase the value of k_d used in remaining cluster formations:

$$k_d = k_d + \frac{k_d - |p|}{|\textit{remaining collection clusters}|}$$

First, we calculate the difference between the requested number of clusters k_d and the number of tasks in the cluster $|p|$. We then divide this by the number of remaining collection clusters that have yet to have delivery clusters formed inside them. We add this value to k_d and continue to the next collection cluster for clustering.

5.2 Dynamic Task Insertion

In a dynamic environment, during the execution of a set of tasks, new tasks to complete may be inserted into the system. To guarantee the completion of this new task t_n we must ensure that the valid complete solution to the task allocation problem $\cup_{r_i \in R} T_{r_i} = T$ continues to hold when an additional task is inserted into the system:

$$\cup_{r_i \in R} T \cup \{t_n\}_{r_i \in R} = T \cup \{t_n\}$$

The simplest way to meet this requirement is, on the dynamic insertion of a new task, instantaneously assigning it to a robot r_i's set of assigned tasks:

$$T_{r_i} \leftarrow T_{r_i} \cup \{t_n\}$$

After the assignment the robot can either *locally replan* its path or it can signal a repeated auction to *globally reallocate* and replan tasks across all robots.

A global reallocation of tasks considers all uninitialised tasks across all robots:

$$\bar{T} \leftarrow \bigcup_i T_{r_i} \backslash T_{r_{\text{init}}}$$

Uninitialised tasks are tasks where a robot has not visited the collection location of an assigned task. Each robot retains the tasks that it has initialised:

$$T_{r_i} \leftarrow T_{r_{\text{init}}}$$

When calculating bids for additional tasks, the completion of these retained tasks is taken into consideration.

Generally speaking, it can be expected that global reallocation of tasks will generate lower cost solutions than local replanning. This is due to the additional consideration of inter-task synergies between tasks that have previously been allocated and any new tasks inserted into the system. Enabling robots to give up tasks that they have previously been allocated allows them to regenerate their task execution plans taking advantage of these additional inter-task synergies.

5.2.1 Empirical Experiments

To empirically contrast local replanning and global reallocation we again simulate 25 office-like environments. For each configuration we test with 10 robots and 60 tasks. We use single-linkage clustering with a TPD metric for our clustering algorithm. For each team objective, we compare three different ratios of dynamic to static tasks, with 25%, 50%, and 75% of tasks unknown at the start. We also compare our results to a baseline one-off task allocation with all tasks known. For simplicity, in our experiments, a new task may only be inserted immediately after a task delivery and is initially allocated to the robot that completed the delivery. Additionally, we compare the effects of different task execution capacities of 1, 3, and 5. We note that when the task execution capacity is 1 this problem is equivalent to operations research's *dynamic stacker crane problem* [7].

To begin, we allocate all initially known tasks $T_{\text{known}} \subseteq T$. For this initial allocation, the number of clusters formed is:

$$k = \frac{1}{2}|T_{\text{known}}|$$

For any repeated auctions, the number of clusters formed individually by each robot is:

$$k = \frac{1}{2}|T_r|$$

The mean results for the MiniMax team objective are presented in Table 5.1 and for the MiniSum team objective in Table 5.2. In considering capacity constraints, unsurprisingly, the absence of constraints produces the lowest team costs and, the more restrictive the constraints, the higher the cost. This directly leads to the largest reduction in team costs occurring in scenarios with highly restrictive constraints. In both team objectives, when robots are restricted to a capacity of only executing one task at a time, a global reallocation of tasks produces better results than the baseline of all tasks known. This result reinforces our previous results for repeated auctions with standard MRTA problems (Section 3.2) and Schoenig and Pagnucco's results for SSI auctions with dynamic tasks [106]. Again, the explanation for this is that during a one-off task allocation, due to the greedy nature of SSI and SSC auctions, each robot can reach a local minimum in its bidding preferences. However, in dynamic environments, due to the introduction of new tasks and the subsequent changes in inter-task synergies this minimum no longer hold and as a result a new local minimum is found during the repeated auction. Overall, global reallocation generally produces lower overall results than local replanning.

MiniMax

Examining the MiniMax team objective results in further detail, the average advantage of global reallocation over local replanning ranges from 11.8% to 36.5%. For local replanning the lowest cost plans are generated when 50% of tasks are unknown. This is unexpected as a naïve assumption can be made that situations with 25% unknown tasks should generate lower cost plans as robots are aware of more tasks during the initial allocation. We speculate that the cause of this result stems from robots discovering additional tasks during the later stages of plan execution and, with no ability to redistribute tasks, they are forced to travel greater distances. This hypothesis is supported by the results for global reallocation failing to show this trend and the large relative improvement gains over local replanning in scenarios with 25% unknown tasks. On the contrary, across the global replanning results, the lowest cost plans are generated when only 25% of tasks are unknown. We speculate that in these instances the knowledge of many tasks helps with

Table 5.1: Dynamic Task Insertion Results with the MiniMax Team Objective (percentage improvement of reallocation compared to replanning in brackets).

		Local Replanning		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	7857	8228	7276	7275
3	3985	5430	4976	6256
5	3070	4620	4471	6250
∞	2602	4438	4418	6250
		Global Reallocation		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	7857	5228 (36.5%)	5593 (23.1%)	5911 (18.7%)
3	3985	3800 (30.0%)	4389 (11.8%)	4877 (22.0%)
5	3070	3469 (24.9%)	3800 (15.0%)	5000 (20.0%)
∞	2602	3415 (23.1%)	3810 (13.8%)	5165 (17.4%)

Table 5.2: Dynamic Task Insertion Results with the MiniSum Team Objective.

		Local Replanning		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	41539	41527	42025	45305
3	21245	26922	28316	33238
5	16519	22554	25720	30096
∞	10150	16613	19675	27985
		Global Reallocation		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	41539	37449 (9.8%)	40278 (4.2%)	36907 (18.5%)
3	21245	25775 (4.3%)	28303 (0.0%)	26594 (20.0%)
5	16519	22639 (-0.4%)	24692 (4.0%)	22238 (26.1%)
∞	10150	17813 (-7.2%)	22207 (-13%)	20332 (27.3%)

the formation of new clusters and repeated auctions of tasks. Overall, in the worst case of 75% tasks unknown, on average, a robot travels a maximum of twice the distance of the baseline result.

The plots in Figure 5.4 show the distribution of these results for each capacity constraint. Some further insights can be gained by examining these plots. First, all plots show, regardless of capacity constraints, that in highly dynamic environments (75% unknown tasks) all maximum distance scores are within similar ranges. A possible explanation of this is due to few tasks being initially known, robots are naturally constrained to executing few tasks at any moment in time and therefore are not able to fully take advantage of inter-task synergies. Second, the plots clearly show in all cases the advantage of global reallocation over local replanning. Finally, the effects of the extreme restriction of an executing capacity constraint of 1 is clear as this plot shows an opposite trend to the max-

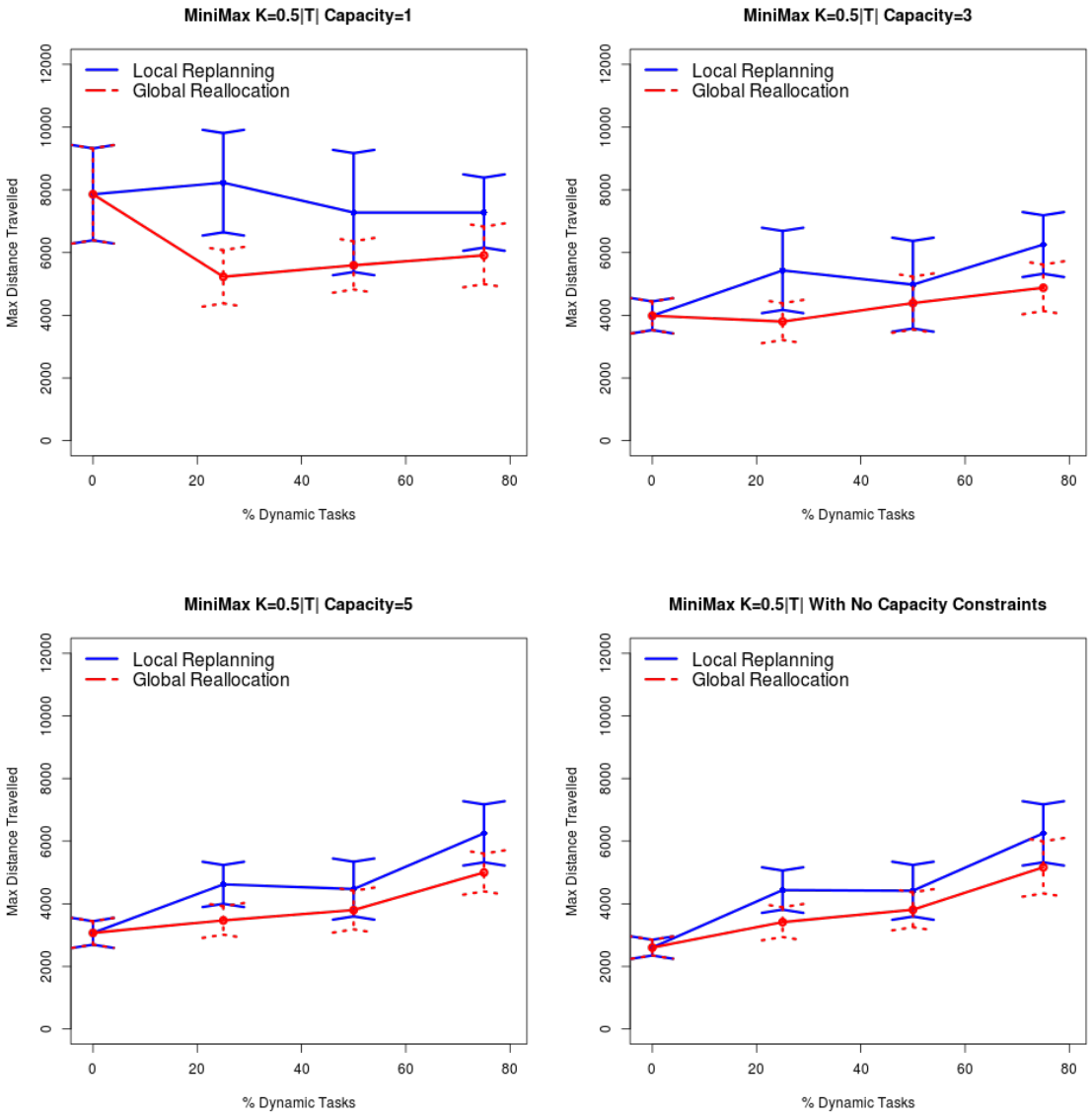


Figure 5.4: Distribution of Results for Task Insertion with the MiniMax Team Objective.

imum distance cost compared to all other plots. This reinforces the previous conclusions that post-initial reallocation of tasks is of key benefit in minimising overall team costs.

MiniSum

The MiniSum team objective results (Table 5.2) show a much smaller benefit in global reallocation over local replanning. These differences range from a 13% increase to a 27% decrease in total distance travelled. Local replanning produces the lowest cost solutions when only 25% of tasks are unknown at the start. Contrarily in global reallocation, the lowest cost plans and largest relative improvements over local replanning are generated in the highly dynamic environment of 75% of tasks initially unknown. We speculate that this is due to the high numbers of repeated cluster formations exposing many different inter-task synergies during each repeated auction. Additionally, in scenarios with no capacity constraints, local replanning surprisingly outperforms global reallocation. A possible explanation of this is that if a few robots commit to executing large numbers of tasks in parallel they, as a consequence of our experiment design, are more likely to discover additional tasks than robots with few tasks. Furthermore, it can then be assumed that there will be few uninitialised tasks for auction during a global reallocation, and this leaves little room for cost reductions. Again, in the worst-case scenario the maximum distance result is twice as large as the corresponding baseline result.

We again plot the distribution of these results for each capacity constraint (Figure 5.5). These plots again provide an interesting insight into the data and the behaviour of the algorithm. First, the extreme influence of a task execution capacity constraint of 1 is demonstrated with the costs for both local replanning and global reallocation erratic with regards to the number of unknown tasks. Second, for all other capacity constraints, including no capacity constraint, local replanning has near identical, and in some instances smaller, result distributions for 25% and 50% unknown tasks. However, in highly dynamic scenarios, global reallocation has a very clear benefit over local replanning. Overall, these results are in stark contrast to the results for the MiniMax team objective. In those prior results the benefits of global reallocation were clear, whereas, in these results, except in highly dynamic scenarios, global reallocation offers no clear advantage.

5.2.2 Computational Time Analysis

To fully contrast local replanning and global reallocation we also consider the amount of computational real-time robots require to generate an initial allocation of tasks, and then the accumulated time required to repeatedly replan or reallocate tasks. These experimental timing results are from a system with a 2.8GHz Intel Core i7 CPU, 8GB RAM, running

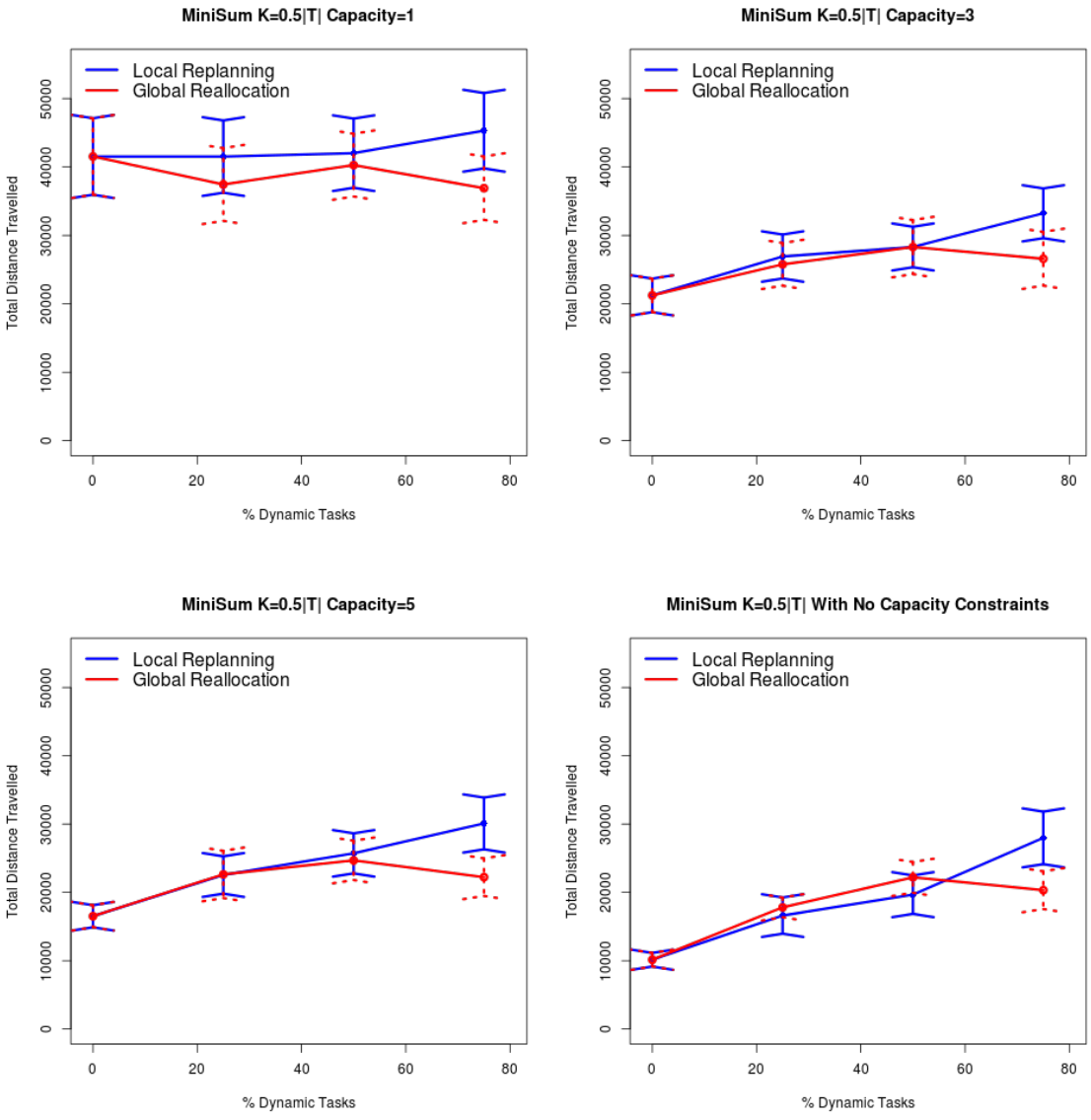


Figure 5.5: Distribution of Results for Task Insertion with the MiniSum Team Objective.

Table 5.3: Mean Initial Task Allocation Computation Time (s) for the MiniMax Team Objective.

Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	268	145	62	16
3	265	145	61	16
5	264	144	61	16
∞	242	137	60	16

Table 5.4: Mean Overall Cumulative Task Allocation Computation Time (s) for the MiniMax Team Objective.

		Local Replanning		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	268	173	90	41
3	265	172	89	42
5	264	171	88	42
∞	242	157	85	41
		Global Reallocation		
Capacity	All Known	25% Unknown	50% Unknown	75% Unknown
1	268	347	276	119
3	265	245	175	106
5	264	226	176	105
∞	242	206	174	105

Ubuntu 11.04 x64.

Table 5.3 shows the mean time required to cluster and auction an initial allocation of tasks for the MiniMax team objective. The results for the MiniSum team objective are not shown, however, they are near identical. Unsurprisingly, the fewer the number of tasks initially known, the faster the initial allocation of tasks. In the most dynamic situation the generation of an initial allocation is 15 times quicker than the baseline. This is an important result because, the quicker an initial allocation is generated, the sooner robots can begin executing tasks. We also note that the task capacity constraint has minimal influence on the time required to generate the initial allocations.

The overall cumulative time required for robots to replan or reallocate upon the insertion of all dynamic tasks into the system is given in Table 5.4. In the worst-case global reallocation of tasks is three times slower than local replanning. Of particular interest is that in all dynamic situations, local replanning was substantially quicker in computational time than the baseline. Also in highly dynamic situations, global reallocation completed all repeated auctions quicker than the single baseline auction.

Overall, taking into consideration mean distance and computational time results we can conclude that when robots seek to achieve the MiniMax team objective it is best

for robots to work together and globally reallocate tasks. However, when robots seek to achieve a MiniSum team objective, except in highly dynamic environments, global reallocation shows no clear improvements over local replanning. Additionally, as a result of its high computational times, in many situations global reallocation would offer few benefits.

5.3 Task Reallocation upon Robot Failure

In this section we consider the reallocation of a failed robot's assigned tasks to the remaining operating robots using SSI auctions. The problem of robot failure is an important consideration in the successful completion of a set of tasks in a distributed robotic system. For example, during task execution individual robots may fail due to malfunctioning equipment or running low on battery power [35]. As task reallocation upon robot failure has not previously been explored with SSI auctions we perform experiments using this more common auction routine and note that this is equivalent to SSC auctions with $k = |T|$.

In this scenario, we consider two different approaches for the reallocation of tasks among the remaining operating robots:

(a) *Partial reallocation* in which only the failed robot's uncompleted tasks are auctioned. This results in the remaining operating robots modifying their existing task execution plans to incorporate additional tasks.

(b) *Global reallocation* of the failed robot's uncompleted tasks and all uninitialised tasks across all operating robots. This an extension of the approach used in the preceding section.

While a global reallocation has greater computation and communication needs, more robot/task and inter-task synergies are considered and therefore it can be expected that this approach would produce lower team costs. However, surprisingly our empirical results (presented below) show that partial allocations produce final results that on average are equivalent to the results for global reallocation.

5.3.1 MRTA Problem Extension for Robot Failure

We now further extend the MRTA problem with collection and delivery (Section 5.1) to continue to provide valid solutions upon robot failure. When a robot fails we remove it from the set of operating robots: $R \leftarrow R \backslash \{r_{\text{fail}}\}$. As a consequence of this, if $T_{r_{\text{fail}}} \neq \emptyset$,

the previous complete solution to the problem no longer holds:

$$\cup_{r_i \in R} T_{r_i} \neq T$$

A new complete solution can be found by reassigning the set of tasks assigned to the failed robot $T_{r_{\text{fail}}}$ to the remaining operating robots. We wish to investigate if it is better for these remaining operating robots to keep their existing commitments or to additionally reallocate all uninitialised tasks in the system.

When a robot detects a malfunction or if it is running low on power it should inform other robots and then safely shutdown. To facilitate this, a failing robot broadcasts a message to all other operating robots containing its present location and the list of all uncompleted tasks assigned to it. Any uninitialised tasks are able to be immediately auctioned. However, tasks that are under execution when the robot fails must be modified. Because the robot has already visited the collection location of these tasks other robots must travel to the location of the failed robot and collect the task from it. To do this the collection location p_c of all initialised tasks $T_{r_{\text{init}}} \subseteq T_{r_{\text{fail}}}$ must be updated to the present location of the failed robot:

$$p_c \leftarrow p_{r_{\text{fail}}}$$

During reallocation robots continue executing their existing current task.

Partial Reallocation

A partial reallocation only auctions the task set assigned to the failed robot:

$$\bar{T} \leftarrow T_{r_{\text{fail}}}$$

The remaining operating robots calculate the bids for these tasks taking into consideration their existing task commitments. Using the cheapest insertion heuristic each robot's existing task execution plan is modified to include any additional task assignments. This approach allows robots to consider inter-task synergies between their existing commitments and tasks they are bidding for that may not have been considered during the previous allocation.

For instance, if in a previous allocation the task $t \in T_{r_{\text{fail}}}$ was assigned during the very first round of bidding no other robot would have been able to consider its synergy with other tasks. Partial reallocation also, generally, has a smaller communication overhead than a global reallocation of all tasks. In a distributed SSI auction the total number of messages sent between all robots is $|\bar{T}| * |R|^2$. The number of tasks for auction in a partial reallocation will always be $|T_{r_{\text{fail}}}| \leq |T|$.

Global Reallocation

A global reallocation considers all of the tasks from the failed robot and all uninitialised tasks across all remaining operating robots:

$$\bar{T} \leftarrow T_{r_{\text{fail}}} \cup \bigcup_i T_{r_i} \backslash T_{r_{\text{init}}}$$

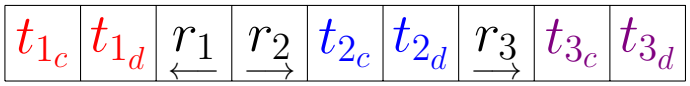
As before, each robot retains the tasks they have already picked up but are yet to deliver and incorporates the costs for the completion of these tasks when calculating bids for additional tasks. It can be expected that robots should be able to minimise costs the most using global reallocation as the change in the number of available robots and subsequent inter-task synergies can be better dealt with than under local replanning.

For example, consider a robot that fails with two initialised tasks and a capacity constraint of three tasks for all robots in the system. Another robot in the system has an existing plan that passes near the failed robot, however, at any one point in its existing plan it only has spare capacity for one additional task. In a partial allocation, this robot could modify its plan and pick up one of the failed robots tasks. The failed robot's second initialised task would then need to be allocated to a different robot, or the robot with the existing plan would need to find another location in its existing plan to return a second time to the failed robot to collect the second task. In a global reallocation, the remaining working robots can give up almost all of their existing tasks. This can result in the robot with limited spare capacity in its current plan giving up some of its existing task commitments allowing it to visit the failed robot once to collect both initialised tasks.

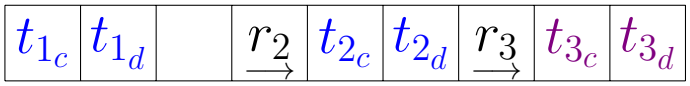
As another example, consider three robots travelling along a horizontal line. The first robot is travelling to the left, the middle robot to the right, and a third robot also travelling to the right (Figure 5.6). The first robot r_1 then fails. In a partial reallocation the middle robot r_2 would need to continue doing tasks to its right and then complete the tasks on its left. However, in a global reallocation the middle robot could give up its tasks to the right and travel to the left and complete the failed robot's tasks. In this situation you would expect that the global reallocation would result in a smaller task cost than the partial allocation.

5.3.2 Experiments

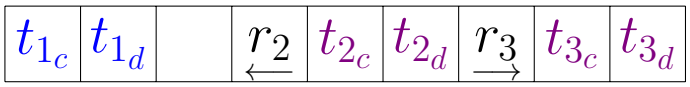
To contrast the differences between partial and global reallocations we simulate 25 configurations of our standard office-like environment. For each configuration we test with 10 identical robots, 60 tasks, and from two to eight robot failures. We compare these results to a baseline cost which is the cost to complete all tasks without any robot fail-



a: Initial Allocation (max $\lambda = 2$).



b: Partial Reallocation (max $\lambda = 7$).



c: Global Reallocation (max $\lambda = 6$).

Figure 5.6: Reallocation example with three robots and six tasks. r_1 is initially travelling to the left, r_2 and r_3 to the right. Tasks assigned to r_1 are in red, to r_2 in blue, and to r_3 in violet.

ures or reallocations of tasks. Robots fail at random intervals after arriving at a collection or delivery location. We test with the MiniMax and MiniSum team objectives and with capacity constraints of 1, 3, and 5.

MiniMax

We present the results for the MiniMax team objective in Table 5.5. The results table also includes the percentage increase in the distance travelled as the result of robot failure and reallocation of tasks compared to the baseline cost. These results are unexpected and show no clear advantage of global reallocations over partial reallocations. Our expectations were that global reallocation would outperform partial reallocation.

Of further note, in both reallocation approaches the total distance travelled decreases as the capacity constraint is increased. This is unsurprising as the larger the capacity constraint the more flexibility robots have in executing multiple tasks in unison. We also note, as the number of robots failing increases, the distance required for the remaining robots to travel increases.

To further analyse these results we looked at the distribution of the final costs for both reallocation techniques. Figure 5.7 shows one standard deviation around the mean for each of the capacities tested. Our first observation is that the results for both partial and global reallocations almost completely overlap. At no point does one technique offer a clear advantage over the other. Our second observation is that the standard deviation remains small in all but the extreme case of eight failed robots. We speculate this is due to robots with lower costs than the robot with the maximum cost taking on additional tasks from failed robots without impacting the overall maximum cost.

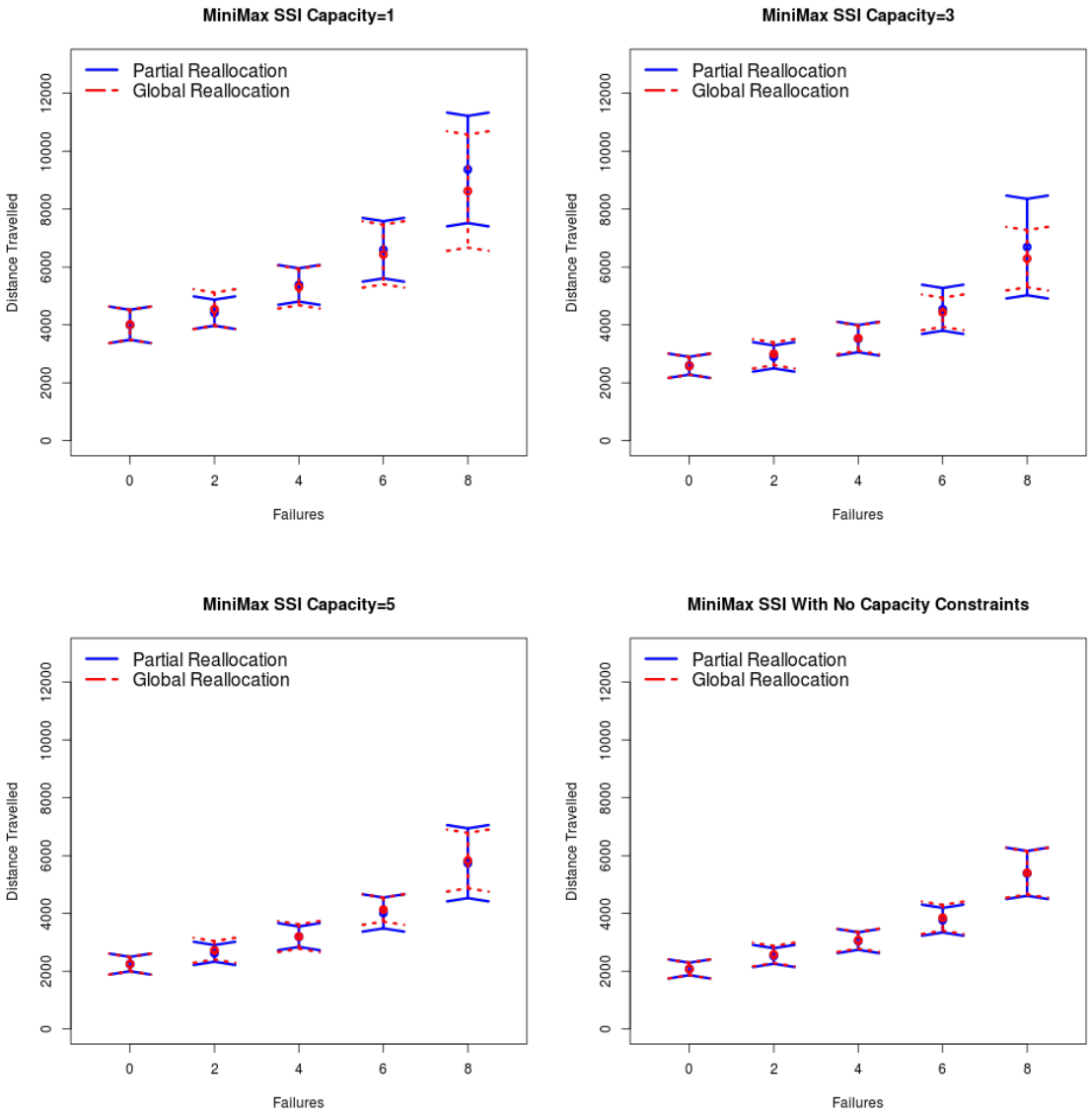


Figure 5.7: Distribution of Results for Robot Failures with the MiniMax Team Objective.

Table 5.5: Dynamic Robot Failure with the MiniMax Team Objective Results (percentage increase in cost after reallocation compared to initial cost in brackets).

Capacity	Failures	Initial λ	Partial Realloc λ	Global Realloc λ
1	2	4005	4424 (10.5%)	4545 (13.5%)
1	4	4005	5383 (34.4%)	5309 (32.6%)
1	6	4005	6596 (64.7%)	6435 (60.7%)
1	8	4005	9370 (134.0%)	8624 (115.4%)
3	2	2591	2897 (11.8%)	3004 (15.9%)
3	4	2591	3527 (36.1%)	3541 (36.6%)
3	6	2591	4539 (75.1%)	4437 (71.2%)
3	8	2591	6691 (158.2%)	6290 (142.7%)
5	2	2249	2619 (16.4%)	2725 (21.1%)
5	4	2249	3196 (42.1%)	3201 (42.3%)
5	6	2249	4016 (78.5%)	4134 (83.8%)
5	8	2249	5738 (155.1%)	5831 (159.2%)
∞	2	2001	2438 (21.8%)	2482 (24.0%)
∞	4	2001	2923 (46.1%)	2949 (47.4%)
∞	6	2001	3611 (80.5%)	3682 (84.0%)
∞	8	2001	5137 (156.7%)	5159 (157.4%)

MiniSum

The results for the MiniSum team objective are presented in Table 5.6. The total summated cost for the partial and global reallocations includes the distanced travelled for all robots including the distance travelled until failure for failed robots. These results are also unexpected. In all but one set of parameters tested, partial reallocations resulted in lower overall distance sums than for global reallocations. Again there are trends that, as the capacity constraint is increased, the cost decreases and, as the number of failed robots increases, the cost increases.

We again plot the distributions of the two reallocation techniques. Figure 5.8 is a plot of the distribution of one standard deviation around the mean for the capacity constraint of one. This plot shows a different distribution to that of the MiniMax team objective results. One can observe in this plot that, as the number of failed robots increases, the standard deviation becomes much larger. This indicates that in some of the office/task/robot configurations tested, the final costs remained low despite the large number of robot failures, however, in other configurations the final costs became extremely large. When the number of robot failures remains less than four there is very little difference in means and distributions between partial and global reallocations. However, when the capacity is one, as the number of robot failures becomes large there is a clear benefit in using partial reallocations.

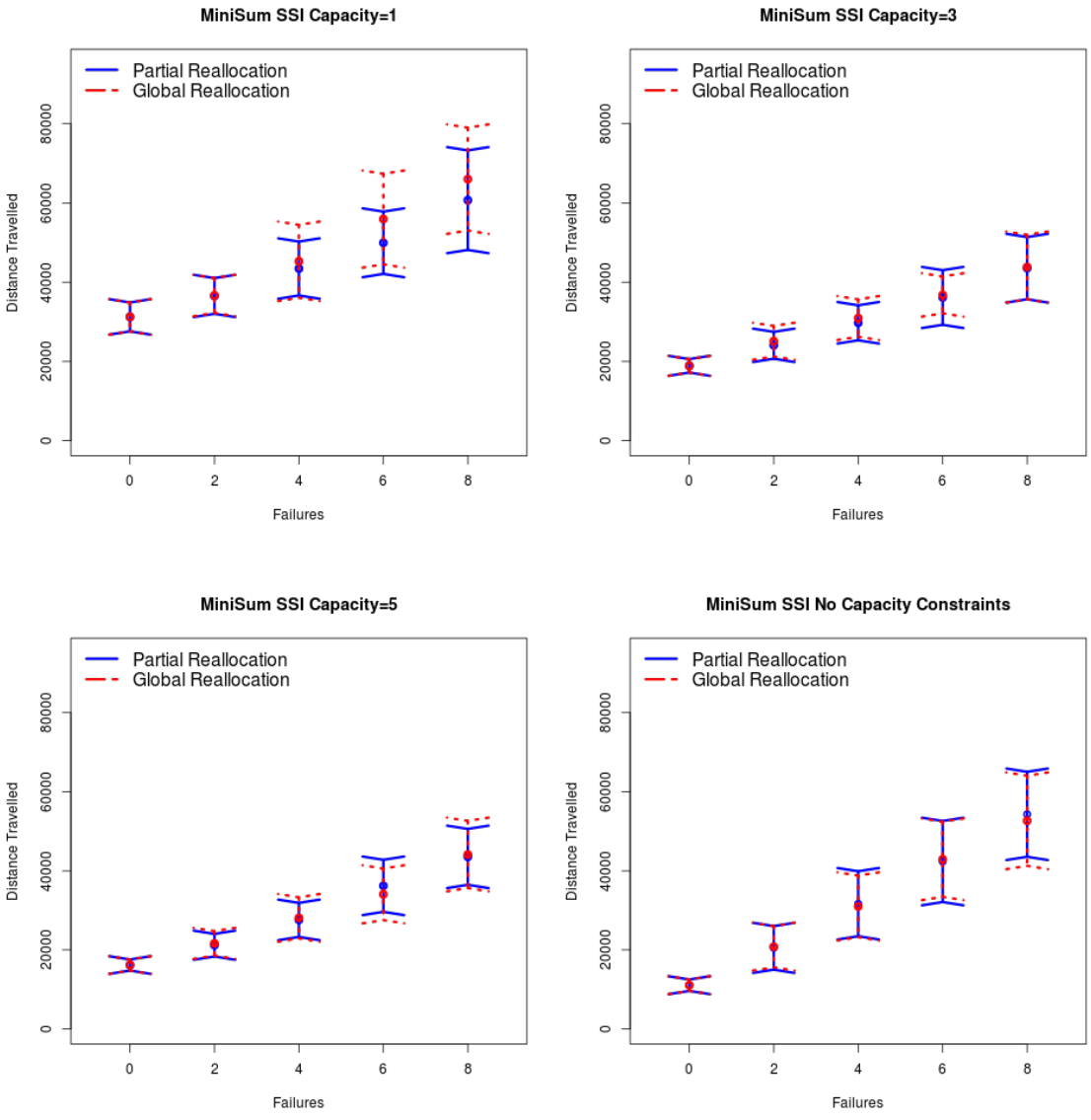


Figure 5.8: Distribution of Results for Robot Failures with the MiniSum Team Objective.

Table 5.6: Dynamic Robot Failure with the MiniSum Team Objective Results (percentage increase in cost after reallocation compared to initial cost in brackets).

Capacity	Failures	Baseline λ	Partial Realloc λ	Global Realloc λ
1	2	31262	36551 (16.9%)	36672 (17.3%)
1	4	31262	43473 (39.1%)	45293 (44.9%)
1	6	31262	49973 (59.9%)	55958 (79.0%)
1	8	31262	60731 (94.3%)	66042 (111.3%)
3	2	18914	24092 (27.4%)	25119 (32.8%)
3	4	18914	29773 (57.4%)	30972 (63.8%)
3	6	18914	36163 (91.2%)	36812 (94.6%)
3	8	18914	43563 (130.3%)	43848 (131.8%)
5	2	16191	21207 (31.0%)	21666 (33.8%)
5	4	16191	27599 (70.5%)	28121 (73.7%)
5	6	16191	36218 (123.7%)	34074 (110.5%)
5	8	16191	43542 (168.9%)	44143 (172.6%)
∞	2	10628	19682 (85.2%)	19887 (87.1%)
∞	4	10628	30574 (187.7%)	29859 (181.0%)
∞	6	10628	40588 (281.9%)	41278 (288.4%)
∞	8	10628	52248 (391.6%)	50929 (379.2%)

Computation Time

Finally, we consider the overall computation time required for generating the initial allocation and for reallocation. Table 5.7 presents the mean timings for the MiniSum team objective (we omit the MiniMax team objective data as the results are nearly identical). The results in this table show that the time required for partial reallocation is much lower than for global reallocation. We note that for the initial allocation the capacity constraint has almost no impact on the time required. For partial reallocation, as the capacity constraint increases there is a smaller increase in the time taken. In global reallocation, this trend is not as apparent, except in scenarios with no capacity constraints. As the number of failed robots increases, both reallocation techniques require more computation time. In particular, the time required for global reallocation grows at a very rapid rate. Overall, from this data and the previous results we can conclude that partial reallocations are a viable technique for handling robot failure. Their resultant costs are at least equal to global reallocation and they have much faster computation times.

Table 5.7: Mean Computation Time (s) for Handling Robot Failure with the MiniSum Team Objective (percentage increase in time after reallocation compared to initial time in brackets).

Capacity	Failures	Initial	Partial Realloc	Global Realloc
1	2	209	250 (19.3%)	330 (57.6%)
1	4	209	265 (25.9%)	439 (110%)
1	6	209	280 (33.3%)	510 (143%)
1	8	209	302 (44.3%)	579 (176%)
3	2	211	261 (23.8%)	339 (60.9%)
3	4	211	284 (36.1%)	419 (101%)
3	6	211	314 (48.4%)	493 (134%)
3	8	211	340 (61.3%)	553 (162%)
5	2	213	269 (25.3%)	343 (59.4%)
5	4	213	307 (44.2%)	433 (103%)
5	6	213	358 (68.7%)	509 (140%)
5	8	213	398 (86.7%)	615 (189%)
∞	2	240	382 (59.3%)	434 (80.8%)
∞	4	238	536 (125%)	563 (136%)
∞	6	238	688 (190%)	747 (214%)
∞	8	236	834 (253%)	878 (271%)

5.3.3 Discussion

Our experiment results are unexpected and appear to contradict previous results on reallocation of tasks using auctions. We can classify previous work into two groups, the first being work that present algorithms for task reallocation [10, 23, 43, 81], and the second dealing with task reallocation upon new task insertion [106, 124]. We are unaware of previous work comparing partial and global reallocation of tasks using repeated auctions.

In this chapter's introduction we stated that dynamic task insertion is a very similar problem. Naïvely, one can assume that adding a new task to the set of tasks: $T \leftarrow T \cup \{t_{new}\}$ and removing a robot from the set of robots: $R \leftarrow R \backslash \{r_{fail}\}$ would affect the task allocation problem in a similar manner, as the complete solution: $\cup_{r_i \in R} T_{r_i} = T$, relies on both R and T (Theorem 2.1).

However, a key difference between dynamic task insertion and robot failure is the location of the tasks for reallocation. Most dynamic task insertion approaches assume that the task location is random. In contrast, when a robot fails, despite the failure occurring at random, the tasks for auction are not randomly distributed in the environment. They are generally geographically close and also contain tight inter-task synergies.

Additionally, in a set of tasks, there may be certain tasks that are undesired by all robots. These tasks are often in locations that are far from other tasks and therefore have high costs, or in the case of robot failure, in locations that are opposite to the remaining

operating robots' directions of travel. During reallocation, robots seek to minimise the global cost through minimisation of their local costs. Generally this works well, however, there are instances where a robot gives up an undesired task in order to commit to additional lower cost tasks, and in later bidding rounds become forced to recommit to their undesired task. In these instances this causes the overall team costs to increase more than if the undesired task was always retained by the original robot and the lower cost tasks distributed to other robots.

Overcoming this problem is difficult, for example, modifying the bidding process to allocate tasks with large costs early on may improve some scenarios. However, previous studies show, this approach has its own local minima and it is also difficult to identify which scenarios are best approached through local cost minimisation and those best approached through minimising maximum task costs [120].

In two examples below, one for the MiniMax (Figure 5.9) and one for the MiniSum (Figure 5.10) team objective, we show this problem occurring in the scenario of robot failure. In both scenarios, the cost to complete one task gradually becomes so great that both robots chose not to bid for this specific task until it is the only task remaining. Because of this the cost to complete this task relative to each robot's current allocation becomes even greater than if it had been allocated prior to less expensive tasks. The reason that this problem doesn't occur in partial reallocations is the retention of previously allocated tasks ensures that the costs to complete these undesired tasks are taken into consideration when bidding for additional tasks.

In both of these examples we assume a capacity constraint of 1.

MiniMax Example

The layout for our MiniMax team objective example is given in Figure 5.9. In this layout all robots are initially travelling to the right. The initial task allocations and path costs are:

$$\lambda_{r_1}(T_{r_1} \leftarrow \{\langle t_{1_c}, t_{1_d}\rangle, \langle t_{2_c}, t_{2_d}\rangle\}) = 11$$
$$\lambda_{r_2}(T_{r_2} \leftarrow \{\langle t_{3_c}, t_{3_d}\rangle, \langle t_{4_c}, t_{4_d}\rangle\}) = 8$$
$$\lambda_{r_3}(T_{r_3} \leftarrow \{\langle t_{5_c}, t_{5_d}\rangle, \langle t_{6_c}, t_{6_d}\rangle\}) = 8$$

The maximum cost of any one of the initial paths is $\max_{r \in R} \lambda_r(T_r) = 11$. Now assume that one time step has elapsed. At this moment robot r_2 fails: $R \leftarrow R \backslash \{r_2\}$. All robots have moved one cell towards their first collection location, robot r_1 is now located at the task collection point t_{1_c} and robot r_3 at point t_{5_c}.

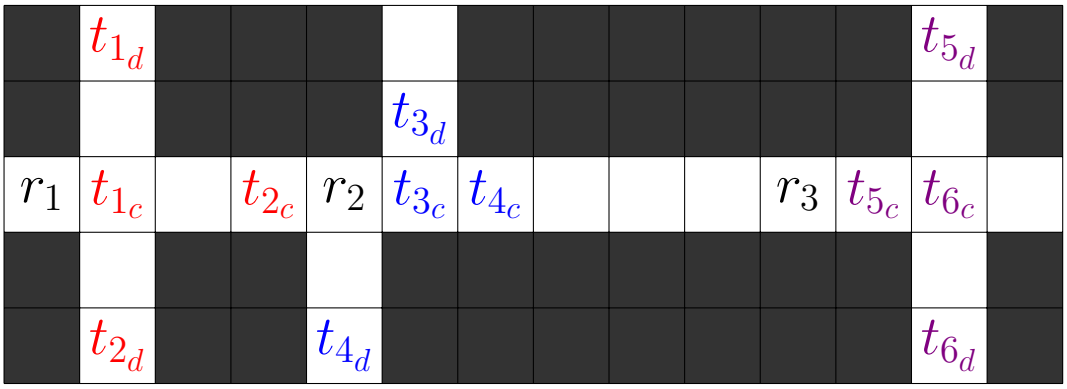


Figure 5.9: MiniMax Team Objective Task Reallocation Example (tasks initially assigned to r_1 are in red, to r_2 in blue, and to r_3 in violet).

We first consider a partial reallocation. Robot r_2's tasks are offered for auction:

$$\bar{T} \leftarrow T_{r_2} = \{\langle t_{3_c}, t_{3_d}\rangle, \langle t_{4_c}, t_{4_d}\rangle\}$$

In Table 5.8 we present the bid calculations for each auction round for a reallocation of these tasks to robots r_1 and r_3. In the first round, robot r_1 is awarded task $\langle t_{3_c}, t_{3_d}\rangle$ and in the following auction round, the cost of completing this additional task is factored into bid calculations. Subsequently, in the second round, robot r_3 is awarded task $\langle t_{4_c}, t_{4_d}\rangle$.

The final task allocations and total path costs are:

$$\lambda_{r_1}(T_{r_1} \leftarrow \{\langle t_{1_c}, t_{1_d}\rangle, \langle t_{3_c}, t_{3_d}\rangle, \langle t_{2_c}, t_{2_d}\rangle\}) = 17$$
$$\lambda_{r_2}(T_{r_2} \leftarrow \{t_{3_c}\}) = 1$$
$$\lambda_{r_3}(T_{r_3} \leftarrow \{\langle t_{5_c}, t_{5_d}\rangle, \langle t_{6_c}, t_{6_d}\rangle, \langle t_{4_c}, t_{4_d}\rangle\}) = 20$$

The maximum cost of any one of the paths after partial reallocation is $\max_{r\in R}\lambda_r(T_r) = 20$.

Now consider a global reallocation. The tasks for auction are:

$$\bar{T} \leftarrow \{\langle t_{2_c}, t_{2_d}\rangle, \langle t_{3_c}, t_{3_d}\rangle, \langle t_{4_c}, t_{4_d}\rangle, \langle t_{6_c}, t_{6_d}\rangle\}$$

Robot r_1 continues executing its first task $\langle t_{1_c}, t_{1_d}\rangle$ and robot r_3 continues executing task $\langle t_{5_c}, t_{5_d}\rangle$ during the reallocation. Table 5.9 presents the bid calculations for each auction round. In the first round robot r_3 wins task $\langle t_{6_c}, t_{6_d}\rangle$. In the second round robot r_1 wins task $\langle t_{3_c}, t_{3_d}\rangle$. In the third auction round robot r_1 wins task $\langle t_{4_c}, t_{4_d}\rangle$. In the final round of the auction robot r_1 wins its third additional task $\langle t_{2_c}, t_{2_d}\rangle$ through simple tie breaking (N.B. the maximum cost would remain the same if task $\langle t_{2_c}, t_{2_d}\rangle$ was awarded to robot r_3).

Table 5.8: Partial Reallocation SSI Auction with the MiniMax team objective for example given in Figure 5.9 (winning bids and assignments in bold).

Round	r_1 Bids Calculated	r_1 Bid Submitted	r_3 Bids Calculated	r_3 Bid Submitted
1	$\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_3, 16\rangle$ $\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 20\rangle$	$\boldsymbol{\beta_{\mathbf{r_1}}^{\mathbf{t_3}}}$	$\beta_{r_3}^{t_3} \leftarrow \langle r_3, t_3, 17\rangle$ $\beta_{r_3}^{t_4} \leftarrow \langle r_3, t_4, 19\rangle$	$\beta_{r_3}^{t_3}$
2	$\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 22\rangle$	$\beta_{r_1}^{t_4}$		$\boldsymbol{\beta_{\mathbf{r_3}}^{\mathbf{t_4}}}$
$\lambda_{r_i}(T_{r_i})$		17		20

Table 5.9: Global Reallocation SSI Auction with the MiniMax team objective for example given in Figure 5.9.

Round	r_1 Bids Calculated	r_1 Bid Submitted	r_3 Bids Calculated	r_3 Bid Submitted
1	$\beta_{r_1}^{t_2} \leftarrow \langle r_1, t_2, 10\rangle$ $\beta_{r_1}^{t_3} \leftarrow \langle r_1, t_3, 9\rangle$ $\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 13\rangle$ $\beta_{r_1}^{t_6} \leftarrow \langle r_1, t_6, 17\rangle$	$\beta_{r_1}^{t_3}$	$\beta_{r_3}^{t_2} \leftarrow \langle r_3, t_2, 18\rangle$ $\beta_{r_3}^{t_3} \leftarrow \langle r_3, t_3, 13\rangle$ $\beta_{r_3}^{t_4} \leftarrow \langle r_3, t_4, 15\rangle$ $\beta_{r_3}^{t_6} \leftarrow \langle r_3, t_6, 7\rangle$	$\boldsymbol{\beta_{\mathbf{r_3}}^{\mathbf{t_6}}}$
2		$\boldsymbol{\beta_{\mathbf{r_1}}^{\mathbf{t_3}}}$	$\beta_{r_3}^{t_2} \leftarrow \langle r_3, t_2, 22\rangle$ $\beta_{r_3}^{t_3} \leftarrow \langle r_3, t_3, 17\rangle$ $\beta_{r_3}^{t_4} \leftarrow \langle r_3, t_4, 19\rangle$	$\beta_{r_3}^{t_3}$
3	$\beta_{r_1}^{t_2} \leftarrow \langle r_1, t_2, 16\rangle$ $\beta_{r_1}^{t_4} \leftarrow \langle r_1, t_4, 15\rangle$	$\boldsymbol{\beta_{\mathbf{r_1}}^{\mathbf{t_4}}}$		$\beta_{r_3}^{t_4}$
4	$\beta_{r_1}^{t_2} \leftarrow \langle r_1, t_2, 22\rangle$	$\boldsymbol{\beta_{\mathbf{r_1}}^{\mathbf{t_2}}}$		$\beta_{r_3}^{t_2}$
$\lambda_{r_i}(T_{r_i})$		23		8

The final task allocations and total path costs are:

$$\lambda_{r_1}(T_{r_1} \leftarrow \{\langle t_{1_c}, t_{1_d}\rangle, \langle t_{3_c}, t_{3_d}\rangle, \langle t_{4_c}, t_{4_d}\rangle, \langle t_{2_c}, t_{2_d}\rangle\}) = 23$$
$$\lambda_{r_2}(T_{r_2} \leftarrow \{t_{3_c}\}) = 1$$
$$\lambda_{r_3}(T_{r_3} \leftarrow \{\langle t_{5_c}, t_{5_d}\rangle, \langle t_{6_c}, t_{6_d}\rangle\}) = 8$$

The maximum cost of any one of the paths after global reallocation is $\max_{r \in R} \lambda_r(T_r) = 23$. Which, in this example, is higher than the cost for a partial reallocation.

When one looks at the bidding calculations, the last task allocated $\langle t_{2_c}, t_{2_d}\rangle$ was only chosen as the task to bid on by either robot in the very last round of bidding. Despite being the second best option for robot r_1 in the first round of bidding, due to the changing costs in inter-task synergies as more tasks were allocated, this task was never considered the best option by any robot until they were forced to commit. Finally, robot r_1 recommits to this task which now has a greater cost than earlier due to the additional tasks to which the robot has committed. In comparison, the partial reallocation auction for this example,

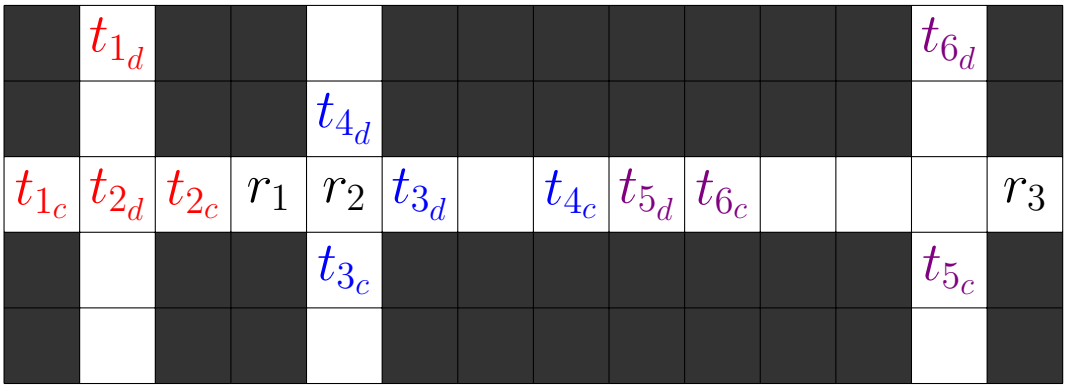


Figure 5.10: MiniSum Team Objective Task Reallocation Example (tasks initially assigned to r_1 are in red, to r_2 in blue, and to r_3 in violet).

never had the problem of task $\langle t_{2_c}, t_{2_d} \rangle$ as robot r_1 retained it from its original allocation and considered the costs of this task when bidding for other tasks.

MiniSum Example

An example of this for the MiniSum team objective is given in Figure 5.10. The initial task allocations and paths are:

$$\lambda_{r_1}(T_{r_1} \leftarrow \{\langle t_{2_c}, t_{2_d} \rangle, \langle t_{1_c}, t_{1_d} \rangle\}) = 6$$
$$\lambda_{r_2}(T_{r_2} \leftarrow \{\langle t_{3_c}, t_{3_d} \rangle, \langle t_{4_c}, t_{4_d} \rangle\}) = 9$$
$$\lambda_{r_3}(T_{r_3} \leftarrow \{\langle t_{5_c}, t_{5_d} \rangle, \langle t_{6_c}, t_{6_d} \rangle\}) = 13$$

The robot r_1 begins by travelling to the left, robot r_2 downwards and robot r_3 will travel to the left to deliver task $\langle t_{5_c}, t_{5_d} \rangle$ and then turn around to complete task $\langle t_{6_c}, t_{6_d} \rangle$. The summated total cost of the initial paths is $\sum_{r \in R} \lambda_r(T_r) = 28$.

Again we assume that one time step has elapsed and all robots have moved one cell towards their first collection location. At this point robot r_1 fails: $R \leftarrow R \backslash \{r_1\}$. When this occurs, robot r_2 is located at the task collection point t_{3_c} and robot r_3 is one cell away from task collection point t_{5_c}.

First we consider a partial reallocation. Robot r_1's tasks are offered for auction:

$$\bar{T} \leftarrow \{\langle t_{1_c}, t_{1_d} \rangle, \langle t_{2_c}, t_{2_d} \rangle\}$$

In Table 5.10 we present the bid calculations for each auction round for a reallocation of these tasks to robots r_2 and r_3. During the first auction round robot r_2 wins task $\langle t_{2_c}, t_{2_d} \rangle$. In the second auction round robot r_2 again wins and is awarded task $\langle t_{1_c}, t_{1_d} \rangle$.

Table 5.10: Partial Reallocation SSI Auction with the MiniSum team objective for example given in Figure 5.10 (winning bids and assignments in bold).

Round	r_2 Bids Calculated	r_2 Bid Submitted	r_3 Bids Calculated	r_3 Bid Submitted
1	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 8\rangle$ $\beta_{r_2}^{t_2} \leftarrow \langle r_2, t_2, 4\rangle$	$\boldsymbol{\beta_{r_2}^{t_2}}$	$\beta_{r_3}^{t_1} \leftarrow \langle r_3, t_1, 21\rangle$ $\beta_{r_3}^{t_2} \leftarrow \langle r_3, t_2, 15\rangle$	$\beta_{r_3}^{t_2}$
2	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 4\rangle$	$\boldsymbol{\beta_{r_2}^{t_1}}$		$\beta_{r_3}^{t_1}$
$\lambda_{r_i}(T_{r_i})$		17		13

Table 5.11: Global Reallocation SSI Auction with the MiniSum team objective for example given in Figure 5.10.

Round	r_2 Bids Calculated	r_2 Bid Submitted	r_3 Bids Calculated	r_3 Bid Submitted
1	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 8\rangle$ $\beta_{r_2}^{t_2} \leftarrow \langle r_2, t_2, 4\rangle$ $\beta_{r_2}^{t_4} \leftarrow \langle r_2, t_4, 6\rangle$ $\beta_{r_2}^{t_6} \leftarrow \langle r_2, t_6, 9\rangle$	$\boldsymbol{\beta_{r_2}^{t_2}}$	$\beta_{r_3}^{t_1} \leftarrow \langle r_3, t_1, 11\rangle$ $\beta_{r_3}^{t_2} \leftarrow \langle r_3, t_2, 7\rangle$ $\beta_{r_3}^{t_4} \leftarrow \langle r_3, t_4, 5\rangle$ $\beta_{r_3}^{t_6} \leftarrow \langle r_3, t_6, 6\rangle$	$\beta_{r_3}^{t_4}$
2	$\beta_{r_2}^{t_1} \leftarrow \langle r_2, t_1, 4\rangle$ $\beta_{r_2}^{t_4} \leftarrow \langle r_2, t_4, 6\rangle$ $\beta_{r_2}^{t_6} \leftarrow \langle r_2, t_6, 18\rangle$	$\boldsymbol{\beta_{r_2}^{t_1}}$		$\beta_{r_3}^{t_4}$
3	$\beta_{r_2}^{t_4} \leftarrow \langle r_2, t_4, 6\rangle$ $\beta_{r_2}^{t_6} \leftarrow \langle r_2, t_6, 18\rangle$	$\beta_{r_2}^{t_4}$		$\boldsymbol{\beta_{r_3}^{t_4}}$
4		$\beta_{r_2}^{t_6}$	$\beta_{r_3}^{t_6} \leftarrow \langle r_3, t_6, 11\rangle$	$\boldsymbol{\beta_{r_3}^{t_6}}$
$\lambda_{r_i}(T_{r_i})$		11		23

The final task allocations and total path costs are:

$$\lambda_{r_1}(T_{r_1} \leftarrow \{t_{2_c}\}) = 1$$
$$\lambda_{r_2}(T_{r_2} \leftarrow \{\langle t_{3_c}, t_{3_d}\rangle, \langle t_{4_c}, t_{4_d}\rangle, \langle t_{2_c}, t_{2_d}\rangle, \langle t_{1_c}, t_{1_d}\rangle\}) = 17$$
$$\lambda_{r_3}(T_{r_3} \leftarrow \{\langle t_{5_c}, t_{5_d}\rangle, \langle t_{6_c}, t_{6_d}\rangle\}) = 13$$

The final summated path cost after partial reallocation is $\sum_{r \in R} \lambda_r(T_r) = 31$.

Now we consider a global reallocation. The tasks for auction are:

$$\bar{T} \leftarrow \{\langle t_{1_c}, t_{1_d}\rangle, \langle t_{2_c}, t_{2_d}\rangle, \langle t_{4_c}, t_{4_d}\rangle, \langle t_{6_c}, t_{6_d}\rangle\}$$

Robot r_2 continues executing task $\langle t_{3_c}, t_{3_d}\rangle$ and robot r_3 continues executing task $\langle t_{5_c}, t_{5_d}\rangle$ during the reallocation. In Table 5.11 we present the bid calculations for each auction round of the reallocation. In the first two auction rounds, robot r_2 is awarded tasks $\langle t_{2_c}, t_{2_d}\rangle$ and $\langle t_{1_c}, t_{1_d}\rangle$ respectively. In the third auction round both robots bid on task

$\langle t_{4_c}, t_{4_d} \rangle$. Robot r_3 wins the auction round and is allocated this task with the lowest bid which results in a different task allocation to the previous partial reallocation. In the final auction round robot r_3 additionally wins task $\langle t_{6_c}, t_{6_d} \rangle$.

The final task allocations and total path costs are:

$$\lambda_{r_1}(T_{r_1} \leftarrow \{t_{2_c}\}) = 1$$
$$\lambda_{r_2}(T_{r_2} \leftarrow \{\langle t_{3_c}, t_{3_d} \rangle, \langle t_{2_c}, t_{2_d} \rangle, \langle t_{1_c}, t_{1_d} \rangle\}) = 11$$
$$\lambda_{r_3}(T_{r_3} \leftarrow \{\langle t_{5_c}, t_{5_d} \rangle, \langle t_{4_c}, t_{4_d} \rangle, \langle t_{6_c}, t_{6_d} \rangle\}) = 23$$

The final summated path cost after global reallocation is $\sum_{r \in R} \lambda_r(T_r) = 35$ which, for this example, is higher than that of partial reallocation.

Again, in this example, a task $\langle t_{6_c}, t_{6_d} \rangle$ that was previously allocated to the remaining operating robot r_3 becomes undesired. In this instance, the two tasks from the failed robot are the first two tasks allocated in the reallocation. However, the robots that these tasks are allocated to do this at the expense of increasing the inter-task synergies to their previous commitments. This is particularly true for robot r_3 in this instance.

We have now shown for both team objectives possible scenarios in which partial reallocations can generate lower costs than global reallocation. In both scenarios, one task that was previously assigned to a remaining operating robot becomes problematic due to high inter-task synergies in global reallocations. While undesired tasks can occur in any task reallocation scenario, it is particularly problematic in the scenario of a failed robot as the tight-knit inter-task synergies of the failed robot's tasks are often more highly desired by other robots than their own previous commitments.

5.4 Summary

In this chapter we have built on our previous work on SSC auctions and demonstrated their effectiveness in task allocation with collection and delivery in dynamic environments. For the scenario of dynamic task insertion our empirical analysis considered the trade-off in performance between local replanning and global reallocation. The key results for this showed that, despite global reallocation generally producing better results than local replanning, there is a large cost in computational time.

For the scenario of task reallocation upon robot failure we focused on two techniques for the reallocation of tasks: partial reallocation which considers only a subset of the total tasks in the system; and, global reallocation which considers almost all tasks in the system. Our empirical evaluations show, that despite global reallocation considering more inter-task synergies, partial reallocations on average performed at least as well.

This result was surprising and contradicted the results for dynamic task insertion which is a similar task reallocation problem. To understand the differences in these dynamic scenarios we generated two simple scenarios in which partial reallocation produces lower cost solutions than global reallocation and discussed the differences in inter-task synergies where tasks are randomly inserted versus tasks requiring reallocation being located in a geographically close area. In particular, we highlighted the problem of undesired tasks and the impact the costs of completing these tasks relative to each robot's other task commitments has on the overall team cost.

Chapter 6

Conclusion

In this thesis we have developed SSC auctions which quickly generate low cost solutions to MRTA problems. Although, there are many existing distributed auction-like approaches for solving this type of problem, few have considered grouping tasks with high inter-task synergies while still ensuring fast solution generation times. Additionally, through repeated cluster formation and auctioning during task execution, our approach allows for solutions to be improved post-initial allocation and react to changes in dynamic environments.

In the related work chapter we outlined four different types of auctions for MRTA problems:

1. Parallel auctions which are a simple approach for distributed and rapid solution generation for MRTA problems. Generally, they produce high cost, and at worse, unbounded cost solutions as they fail to consider inter-task synergies when constructing bids.

2. Sequential auctions which improve on parallel auctions through bid calculations for additional tasks including the cost of completing previously allocated tasks. This approach lowers the team cost as some inter-task synergies are considered. However, the order in which tasks are offered for auction heavily influences the bid calculations and the resulting overall solution costs.

3. SSI auctions which improve this further by allowing individual robots to select any unallocated task of its choosing to place a bid on, and one task is awarded per auction round. Although SSI auctions have provable solution bounds they continue to suffer from problems where tasks that one robot should, for optimality, complete are allocated to different robots. To this end, a number of extensions have been proposed that seek to improve the solution quality by considering different bidding and winner determination strategies.

4. Combinatorial auctions which generate optimal solutions in one auction round are an alternative auction approach. Although optimal solutions are highly desired, in systems with limited resources, the cost involved in calculating these solutions makes them impractical for use in most environments.

A number of authors have previously suggested [102] or studied [21, 139, 31] task clustering as an approach for overcoming these weaknesses in single item auction approaches. In each approach there is a trade-off between the number of inter-task synergies considered, the quality of the solution and the time required to obtain a solution.

For SSC auctions, our approach was to address problems with inter-task synergies arising in single-item auctions and to ensure that SSC auctions ran in a similar time to SSI auctions. As fewer auctions rounds are required in SSC auctions the time gained is used to generate the initial task clusters. Our approach also allows for clusters of different sizes — this differs from other approaches which set fixed cluster sizes (e.g. SSI auctions with bundles [57]) — allowing clusters with many tasks of high inter-task synergies to be formed, rather than arbitrarily limiting the size of clusters.

The key results from our development and empirical study of SSC auctions are:

- SSC auctions produce lower cost solutions than standard SSI auctions and SSI auctions with bundles, and run in equivalent time to standard SSI auctions in static environments with capacity constraints (Section 3.1.8).

- Repeated SSC auctions with dynamic task clusters allow solution costs to be gradually reduced during the execution of tasks (Section 3.2.5).

- SSC auctions with agglomerative clustering are statistically equivalent to SSI auctions in static environments with no capacity constraints (Sections 4.3.1 and 4.3.2).

- For the MiniMax team objective modifying the SSC auction rules to allow the priority allocation of large task clusters produces statistically significant lower team costs compared to standard SSI auctions without capacity constraints (Section 4.4.1).

- In dynamic environments with online tasks requiring collection and delivery, a global reallocation of tasks through repeated auctions greatly reduces the overall team costs compared to local replanning (Section 5.2.1).

- In dynamic environments with robot failure, using SSI auctions to perform a partial reallocation of subsets of the total tasks to be completed outperforms full reallocation of all uncompleted tasks (Section 5.3.2).

6.1 Future Work

The work in this thesis has formed a foundation on which future work that encompasses task clustering and auction mechanisms can build on. There are four broad areas that future work falls under: task clustering algorithms, solution quality measures, extended problem domains, and robot capabilities. In the following subsections we offer suggestions in each of these areas of possible further research focus.

6.1.1 Task Clustering Approaches

Throughout this thesis we have used clustering algorithms that require a pre-defined k value for the number of clusters to form. While we offered a discussion of the merits of possible alternative approaches in Section 4.5, experimental evaluation of these and other approaches may further improve the quality of solutions generated by SSC auctions. Additionally, clustering algorithms that include knowledge of robot locations and likely inter-cluster associations (i.e. the likelihood that two clusters are assigned to the same robot), may produce lower cost MRTA solutions.

Another area of possible consideration is a pre-clustering analysis of the structure of the task set. In some clustering algorithms, the quality of clusters formed is dependent on the distribution and structure of the items being clustered. A pre-clustering analysis may reveal which algorithms are likely to produce the most suitable clusters for a given problem, which in turn, may allow the system to dynamically choose the most appropriate clustering algorithm accordingly.

6.1.2 Solution Quality Measures

In Chapter 3 we presented some basic formal analysis of the bounds of SSC auctions. However, these lack strong formal rigour. A more formal foundation of clustering properties and their relation to auctions would be of benefit. In particular, understanding the relationships between tasks, clusters, robots and the auction process may aid in developing a system which is guaranteed to produce near optimal solutions in minimal time.

In the context of empirical evaluation, a more rigorous study of the relation between the time taken to form clusters and allocate tasks and the cost of the solution would be of interest. For instance, if task clustering approaches that incorporated additional domain knowledge, as discussed above, had a large increase in cluster formation time, how would this trade-off against the improved solution execution time?

6.1.3 Extended Problem Domains

In Section 2.2.1 we listed a number of extensions to the standard MRTA problem definition, and in Chapter 5 we studied two dynamic environment scenarios with tasks requiring collection and delivery. While these two scenarios are relatively common problems faced in MRS domains, there are more complex extensions of these which may be of interest in future work.

One such extension and question is whether knowledge of task locations, robot locations and direction can be used to determine the likelihoods that partial or full reallocations will result in overall lower costs. Additional knowledge of the environment may also factor into decisions about task selection, for instance, in a highly hostile environment, robot failure may be very high, and as such not all tasks in the system may be able to be completed. In this scenario, robots may select tasks to bid on based on the likelihood of completing the highest number of tasks and ensuring self preservation rather than simply trying to lower the overall team cost.

Finally, the trade-off between competing constraints may also be of interest in complex task domains. For instance, in a rideshare scenario, passengers desire to get from a pickup location to a drop-off location as quickly as possible. Whereas, the company's objective may be to minimise distance travelled and therefore energy used.

6.1.4 Robot Capabilities

To date, all empirical evaluations have been simulation based and not validated on physical robot systems. Although simulation is common in many multi-robot auction systems [126, 8, 62, 122, 138, 57, 58, 75, 80, 81], validation on real robots is highly desired. A possible suitable platform for such experimentation is the newly formed RoboCup Logistics League.[1]

Additionally, empirical evaluation with teams of heterogeneous robots and tasks would be of interest. Tasks requiring specific robot capabilities would require appropriate changes to cluster formation algorithms and this would result in disparities in bidding and task allocations based on the abilities of individual robots to complete specific tasks, rather than simply on their location and current task commitments.

[1] http://www.robocup-logistics.org/

References

[1] T. Allard and S. Shekh. Hierarchical multi-agent distribution planning. In *Proceedings of the Australasian Joint Conference Australasian Conference on Artificial Intelligence*, pages 755–766, 2012.

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 49–60, 1999.

[3] G. Ball and D. Hall. ISODATA, a novel method of data analysis and pattern classification. Technical report, Stanford research institute, 1965.

[4] F. Bellifemine, A. Poggi, and G. Rimassa. JADE–a FIPA-compliant agent framework. In *Proceedings of the International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM)*, volume 99, pages 97–108, 1999.

[5] J. L. Bentley. Fast algorithms for geometric traveling salesman problems. *INFORMS Journal on Computing*, 4(4):387–411, 1992.

[6] G. Berbeglia, J. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15(1):1–31, 2007.

[7] G. Berbeglia, J. Cordeau, and G. Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8–15, 2010.

[8] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. M. Griffin, and A. J. Kleywegt. Robot exploration with combinatorial auctions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1957–1962, 2003.

[9] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[10] S. C. Botelho and R. Alami. M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1234–1239, 1999.

[11] C. Boutilier, M. Goldszmidt, and B. Sabata. Sequential auctions for the allocation of resources with complementarities. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 527–523, 1999.

[12] E. Broadbent, R. Q. Stafford, and B. A. MacDonald. Acceptance of healthcare robots for the older population: Review and future directions. *International Journal of Social Robotics*, 1(4):319–330, 2009.

[13] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011.

[14] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. A. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica (Slovenia)*, 30(1):3–31, 2006.

[15] S. A. Chien, A. Barrett, T. A. Estlin, and G. Rabideau. A comparison of coordinated planning methods for cooperating rovers. In *Proceedings of the International Conference on Autonomous Agents*, pages 100–101, 2000.

[16] G. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6:791–812, 1958.

[17] T. Dagaeff, F. Chantemargue, and B. Hirsbrunner. Emergence-based cooperation in a multi-agent system. In *Proceedings of the European Conference on Cognitive Science (ECCS)*, pages 91–96, 1997.

[18] P. Davidsson, L. Henesey, L. Ramstedt, J. Törnquist, and F. Wernstedt. Agent-based approaches to transport logistics. In *AAMAS Workshop on Agents in Traffic and Transportation*, 2004.

[19] W. S. DeSarbo and V. Mahajan. Constrained classification: the use of a priori information in cluster analysis. *Psychometrika*, 49(2):187–215, 1984.

[20] M. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, pages 115–122, 2000.

[21] M. Dias and A. Stentz. Opportunistic optimization for market-based multirobot control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2714–2720. IEEE, 2002.

[22] M. B. Dias and A. Stentz. A comparative study between centralized, market-based, and behavioral multirobot coordination approaches. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2279–2284, 2003.

[23] M. B. Dias, M. Zinck, R. Zlot, and A. Stentz. Robust multirobot coordination in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3435–3442, 2004.

[24] 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, 2006.

[25] M. B. Dias, R. Zlot, M. Zinck, J. P. Gonzalez, and A. Stentz. A versatile implementation of the traderbots approach for multirobot coordination. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, 2004.

[26] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman. On cooperation in multi-agent systems. *The Knowledge Engineering Review*, 12(3):309–314, 1997.

[27] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):29–41, 1996.

[28] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):1–21, 2002.

[29] J. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[30] A. Ekici, P. Keskinocak, and S. Koenig. Multi-robot routing with linear decreasing rewards over time. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 958–963, 2009.

[31] M. Elango, S. Nachiappan, and M. K. Tiwari. Balancing task allocation in multi-robot systems using k-means clustering and auction based mechanisms. *Expert Systems with Applications*, 38(6):6486–6491, 2011.

[32] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[33] V. Faber. Clustering and the continuous k-means algorithm. *Los Alamos Science*, 22:138–144, 1994.

[34] A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(5):2015–2028, 2004.

[35] A. Farinelli, P. Scerri, and M. Tambe. Building large-scale robot systems: Distributed role assignment in dynamic, uncertain domains. In *Proceedings of the Workshop on Representations and Approaches for Time-Critical Decentralized Resource, Role and Task Allocation*, 2003.

[36] K. Fischer, J. P. Müller, and M. Pischel. Cooperative transportation scheduling: An application domain for dai. *Applied Artificial Intelligence*, 10(1):1–34, 1996.

[37] K. Fischer, J. P. Müller, M. Pischel, and D. Schier. A model for cooperative transportation scheduling. In *Proceedings of the International Conference on Multiagent Systems (ICMAS)*, pages 109–116, 1995.

[38] J. Forlizzi and C. F. DiSalvo. Service robots in the domestic environment: a study of the roomba vacuum in the home. In *Proceedings of the ACM SIGCHI/SIGART Conference on Human-Robot Interaction (HRI)*, pages 258–265, 2006.

[39] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325–344, 2000.

[40] V. Frías-Martínez, E. Sklar, and S. Parsons. Exploring auction mechanisms for role assignment in teams of autonomous robots. In *Proceedings of RoboCup 2004: Robot Soccer World Cup VIII*, pages 532–539, 2004.

[41] K. Ganesh and T. T. Narendran. Cloves: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. *European Journal of Operational Research*, 178(3):699–717, 2007.

[42] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, volume 1, pages 317–323, 2003.

[43] B. P. Gerkey and M. J. Matarić. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics*, 18(5):758–768, 2002.

[44] M. Golfarelli, D. Maio, and S. Rizzi. Multi-agent path planning based on task-swap negotiation. In *Proceedings of the UK Planning and Scheduling SIG Workshop*, pages 69–82. Citeseer, 1997.

[45] M. Golfarelli, D. Maio, and S. Rizzi. A task-swap negotiation protocol based on the contract net paradigm. Technical report, Universitá di Bologna, 1997.

[46] M. Golfarelli and S. Rizzi. Spatio-temporal clustering of tasks for swap-based negotiation protocols in multi-agent systems. In *Proceedings International Conference on Intelligent Autonomous Systems (IAS)*, pages 172–179. Venice, Italy, 2000.

[47] J. Hartigan and M. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society*, 28(1):100–108, 1979.

[48] H. H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings of the National Conference on Artificial Intelligence*, pages 22–29, 2000.

[49] H. Iba. Emergent cooperation for multiple agents using genetic programming. In *Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 32–41, 1996.

[50] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[51] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *Knowledge Engineering Review*, 8:223–223, 1993.

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

[53] S. Kalam, M. Gani, and L. Seneviratne. A game-theoretic approach to non-cooperative target assignment. *Robotics and Autonomous Systems*, 58(8):955–962, 2010.

[54] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[55] S. Keshmiri and S. Payandeh. Multi-robot, dynamic task allocation: a case study. *Intelligent Service Robotics*, 6:1–18, 2013.

[56] S. Koenig, C. A. Tovey, M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, A. Meyerson, and S. Jain. The power of sequential single-item auctions for agent coordination. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1625–1629, 2006.

[57] S. Koenig, C. A. Tovey, X. Zheng, and I. Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1359–1365, 2007.

[58] S. Koenig, X. Zheng, C. A. Tovey, R. B. Borie, P. Kilby, V. Markakis, and P. Keskinocak. Agent coordination with regret clearing. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 101–107, 2008.

[59] R. C. Kohout and K. Erol. In-time agent-based vehicle routing with a stochastic improvement heuristic. In *Proceedings of the National Conference on Artificial Intelligence*, pages 864–869, 1999.

[60] W. J. Krzanowski and Y. Lai. A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Biometrics*, 44:23–34, 1988.

[61] T. Kurita. An efficient agglomerative clustering algorithm using a heap. *Pattern Recognition*, 24(3):205–209, 1991.

[62] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt. Simple auctions with performance guarantees for multi-robot task allocation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 698–705, 2004.

[63] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, pages 343–350, 2005.

[64] E. L. Lawler, J. K. Lenstra, A. R. Kan, and D. B. Shmoys. *The traveling salesman problem: a guided tour of combinatorial optimization*, volume 3. Wiley Chichester, 1985.

[65] M. K. Lee, S. B. Kiesler, J. Forlizzi, S. S. Srinivasa, and P. E. Rybski. Gracefully mitigating breakdowns in robotic services. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 203–210, 2010.

[66] K. Lerman, C. V. Jones, A. Galstyan, and M. J. Mataric. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[67] L. Liu and D. A. Shell. Multi-level partitioning and distribution of the assignment problem for large-scale multi-robot task allocation. In *Robotics: Science and Systems*, 2011.

[68] L. Liu and D. A. Shell. Large-scale multi-robot task allocation via dynamic partitioning and distribution. *Autonomous Robots*, 33(3):291–307, 2012.

[69] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.

[70] K. S. Macarthur, R. Stranders, S. D. Ramchurn, and N. R. Jennings. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[71] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. California, USA, 1967.

[72] M. Mes, M. van der Heijden, and A. van Harten. Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1):59–75, 2007.

[73] G. W. Milligan. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, 45(3):325–342, 1980.

[74] Y. Mohammadi, A. Tazari, and M. Mehrandezh. A new hybrid task sharing method for cooperative multi agent systems. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 2045–2048. IEEE, 2005.

[75] A. R. Mosteo and L. Montano. Comparative experiments on optimization criteria and algorithms for auction based multi-robot task allocation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3345–3350, 2007.

[76] R. Müller. *Combinatorial Auctions*, chapter Tractable Cases of the Winner Determination Problem, pages 319–336. The MIT Press, 2006.

[77] B. Mutlu and J. Forlizzi. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In *Proceedings of the*

ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 287–294, 2008.

[78] G. Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 56(5):836–863, 1968.

[79] M. Nanjanath, A. J. Erlandson, S. Andrist, A. Ragipindi, A. A. Mohammed, A. S. Sharma, and M. L. Gini. Decision and coordination strategies for robocup rescue agents. In *Proceedings of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 473–484, 2010.

[80] M. Nanjanath and M. L. Gini. Dynamic task allocation for robots via auctions. In *ICRA*, pages 2781–2786, 2006.

[81] M. Nanjanath and M. L. Gini. Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems*, 58(7):900–909, 2010.

[82] B. Nebel, C. Dornhege, and A. Hertle. How much does a household robot need to know in order to tidy up? In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[83] L. E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1):5–14, 2008.

[84] S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[85] M. Pavone, J. C. Castillo-Rogez, J. Hoffman, I. Nesnas, and N. J. Strange. Spacecraft/rover hybrids for the exploration of small solar system bodies. In *IEEE Proceedings*, volume 2425, 2013.

[86] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[87] D. Puig, M. A. García, and L. Wu. A new global optimization strategy for coordinated multi-robot exploration: Development and comparative evaluation. *Robotics and Autonomous Systems*, 59(9):635–653, 2011.

[88] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized coordination in robocup rescue. *The Computer Journal*, 53(9):1447–1461, 2010.

[89] S. Ray and R. H. Turi. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the International Conference on Advances in Pattern Recognition and Digital Techniques*, pages 137–143, 1999.

[90] P. Rousseeuw and L. Kaufman. Clustering by means of medoids. *Statistical data analysis based on the L1-norm and related methods*, page 405, 1987.

[91] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[92] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, NJ, 2 edition, 2003.

[93] D. Sáez, C. E. Cortés, and A. Núñez. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computers & Operations Research*, 35(11):3412–3438, 2008.

[94] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 256–262, 1993.

[95] T. Sandholm. Contract types for satisficing task allocation: I theoretical results. In *Proceedings of the AAAI spring symposium: Satisficing models*, pages 68–75, 1998.

[96] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artifical Intelligence*, 135(1-2):1–54, 2002.

[97] T. Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 90–97, 2000.

[98] T. Sandholm and S. Suri. Bob: Improved winner determination in combinatorial auctions and generalizations. *Artifical Intelligence*, 145(1-2):33–58, 2003.

[99] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Cabob: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1102–1108, 2001.

[100] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Cabob: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, 2005.

[101] S. Sariel, T. Balch, and J. Stack. Empirical evaluation of auction-based coordination of auvs in a realistic simulated mine countermeasure task. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 197–206. Springer, 2006.

[102] S. Sariel and T. R. Balch. Efficient bids on task allocation for multi-robot exploration. In *Proceedings of the International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 116–121, 2006.

[103] S. Sariel, T. R. Balch, and N. Erdogan. Robust multi-robot cooperation through dynamic task allocation and precaution routines. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics*, pages 196–201, 2006.

[104] S. Sariel-Talay, T. Balch, and N. Erdogan. Multiple traveling robot problem: A solution based on dynamic task selection and robust execution. *IEEE/ASME Transactions on Mechatronics*, 14(2):198–206, 2009.

[105] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 727–734, 2005.

[106] A. Schoenig and M. Pagnucco. Evaluating sequential single-item auctions for dynamic task allocation. In *Proceedings of the Australasian Joint Conference Australasian Conference on Artificial Intelligence*, pages 506–515, 2010.

[107] J. R. Searle. Collective intentions and actions. *Intentions in communication*, page 401, 1990.

[108] G. P. Settembre, P. Scerri, A. Farinelli, K. P. Sycara, and D. Nardi. A decentralized approach to cooperative situation assessment in multi-robot systems. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 31–38, 2008.

[109] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

[110] R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*, 11(1):61–70, 1981.

[111] S. L. Smith and F. Bullo. Monotonic target assignment for robotic networks. *IEEE Transactions on Automatic Control*, 54(9):2042–2057, 2009.

[112] P. Sneath and R. Sokal. *Numerical taxonomy: The principles and practice of numerical classification.* WH Freeman, 1973.

[113] A. Solanas and M. A. García. Coordinated multi-robot exploration through unsupervised clustering of unknown space. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 717–721, 2004.

[114] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[115] J. Song and A. Regan. Approximation algorithms for the bid construction problem in combinatorial auctions for the procurement of freight transportation contracts. *Transportation Research Part B: Methodological*, 39(10):914–933, 2005.

[116] D. Steinley. Local optima in k-means clustering: what you don't know may hurt you. *Psychological Methods*, 8(3):294, 2003.

[117] A. W. Stroupe, A. Okon, M. L. Robinson, T. Huntsberger, H. Aghazarian, and E. T. Baumgartner. Sustainable cooperative robotic technologies for human and robotic outpost infrastructure construction and maintenance. *Autonomous Robots*, 20(2):113–123, 2006.

[118] C. A. Sugar and G. M. James. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98(463):750–763, 2003.

[119] C. Sung, N. Ayanian, and D. Rus. Improving the performance of multi-robot systems by task switching. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2999–3006, 2013.

[120] G. Thomas and A. B. Williams. Sequential auctions for heterogeneous task allocation in multiagent routing domains. In *Systems, Man and Cybernetics*, pages 1995–2000. IEEE, 2009.

[121] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[122] C. Tovey, M. Lagoudakis, S. Jain, and S. Koenig. The generation of bidding rules for auction-based robot coordination. *Multi-Robot Systems.*, III:3–14, 2005.

[123] V. Trianni, C. Ampatzis, A. L. Christensen, E. Tuci, M. Dorigo, and S. Nolfi. From solitary to collective behaviours: Decision making and cooperation. In *Advances in Artificial Life*, pages 575–584, 2007.

[124] J. Van Duin, L. Tavasszy, and E. Taniguchi. Real time simulation of auctioning and re-scheduling processes in hybrid freight markets. *Transportation Research Part B: Methodological*, 41(9):1050–1066, 2007.

[125] L. Vig and J. A. Adams. Multi-robot coalition formation. *IEEE Transactions on Robotics*, 22(4):637–649, 2006.

[126] A. Viguria, I. Maza, and A. Ollero. Set: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3344, 2007.

[127] M. J. Wooldridge. *An introduction to multiagent systems*. Wiley, 2009.

[128] L. Wu, M. A. Garcia, D. Puig, and A. Sole. Voronoi-based space partitioning for coordinated multi-robot exploration. *Journal of Physical Agents*, 1(1):37–44, 2007.

[129] L. Wu, D. Puig Valls, and M. Á. García García. Balanced multi-robot exploration through a global optimization strategy. *Journal of Physical Agents*, 4(1):35–44, 2010.

[130] K. M. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1160–1165, 2008.

[131] Y. Xu, P. Scerri, K. P. Sycara, and M. Lewis. Comparing market and token-based coordination. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1113–1115, 2006.

[132] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[133] K. Zhang, E. G. Collins Jr., and A. Barbu. A novel stochastic clustering auction for task allocation in multi-robot teams. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3300–3307, 2010.

[134] K. Zhang, E. G. Collins Jr., and A. Barbu. An efficient stochastic clustering auction for heterogeneous robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4806–4813, 2012.

[135] K. Zhang, E. G. Collins Jr., and D. Shi. Centralized and distributed task allocation in multi-robot teams via a stochastic clustering auction. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(2):21, 2012.

[136] X. Zheng and S. Koenig. K-swaps: Cooperative negotiation for solving task-allocation problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 373–379, 2009.

[137] X. Zheng and S. Koenig. Generalized reaction functions for solving complex-task allocation problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 478–483. AAAI Press, 2011.

[138] X. Zheng, S. Koenig, and C. A. Tovey. Improving sequential single-item auctions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2238–2244, 2006.

[139] R. Zlot and A. Stentz. Complex task allocation for multiple robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1515–1522, 2005.

[140] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3016–3023, 2002.