

This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (<https://dspace.lboro.ac.uk/>) under the following Creative Commons Licence conditions.



For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

**Task Allocation and Consensus with Groups of Cooperating
Unmanned Aerial Vehicles**

by

Simon Hunt

A Doctoral Thesis

Submitted in Partial Fulfilment
of the requirements for the award of

Doctor of Philosophy
of
Department of Computer Science

15th January 2014

Copyright 2014 Simon James Hunt

Abstract

Task Allocation and Consensus with Groups of Cooperating Unmanned Aerial Vehicles

Simon Hunt

Loughborough University, 2014

Supervisor: Qinggang Meng

Supervisor: Chris Hinde

The applications for Unmanned Aerial Vehicles are numerous and cover a range of areas from military applications, scientific projects to commercial activities, but many of these applications require substantial human involvement. This work focuses on the problems and limitations in cooperative Unmanned Aircraft Systems to provide increasing realism for cooperative algorithms. The Consensus Based Bundle Algorithm is extended to remove single agent limits on the task allocation and consensus algorithm. Without this limitation the Consensus Based Grouping Algorithm is proposed that allows the allocation and consensus of multiple agents onto a single task. Solving these problems further increases the usability of cooperative Unmanned Aerial Vehicles groups and reduces the need for human involvement. Additional requirements are taken into consideration including equipment requirements of tasks and creating a specific order for task completion. The Consensus Based Grouping Algorithm provides a conflict free feasible solution to the multi-agent task assignment problem that provides a reasonable assignment

without the limitations of previous algorithms. Further to this the new algorithm reduces the amount of communication required for consensus and provides a robust and dynamic data structure for a realistic application. Finally this thesis provides a biologically inspired improvement to the Consensus Based Grouping Algorithm that improves the algorithms performance and solves some of the difficulties it encountered with larger cooperative requirements.

Acknowledgements

As this thesis draws closer to its end it seems appropriate now to spend some time looking back and giving thought to those who helped me reach this point. This Ph.D. has coincided with some of my lowest moments in my life but equally with the greatest moments of my life so far. I do not think I could have ever achieved this without the support and dedication of a number of people. Firstly I would like to thank both my supervisors Dr. Qinggang Meng and Prof. Chris J. Hinde, who have provided untold help to me throughout my research. They have helped support and motivate me when I was at my lowest and without them I certainly would not have made it to the end. Thanks to EPSRC for funding part of this research.

To my family for supporting me throughout and a particular thank you to my father Peter who pushed me so sternly but never stopped caring, had I listened to you more often perhaps I would be writing this a bit sooner. I must not forget the final family member, Molly the family dog, it is surprising how much clarity of thought walking the dog and talking to yourself can do.

The support my friends have given me especially those in the Loughborough Students Union Computer Society. For all the enjoyable steak nights we have had and the entertaining weekends at James France. A special thank you to Miles Sutcliffe for being there when I needed it with both good advice and good cooking.

My final and parting dedication goes my partner in life, Nadia Ong, you have made me more confident and helped push me through these final stages of this thesis. It does not matter what happens in the future because I know you will always be there for me.

Table of Contents

Abstract.....	i
Acknowledgements.....	iii
Table of Contents.....	v
List of Tables	viii
List of Figures	x
Chapter 1: <i>Introduction</i>	1
1.1 Introduction.....	1
1.2 Problem Statement.....	3
1.3 Objectives	5
1.4 Document Layout.....	5
1.5 Background	6
1.5.1 Unmanned Aerial Vehicles	6
1.5.2 Autonomous Systems.....	7
1.5.3 Multi-Agent Systems	11
1.5.4 Eusociality.....	13
Chapter 2: <i>Literature Review</i>	17
2.1 Centralised and Decentralised Systems	17
2.2 Task Allocation and Consensus	19
2.2.1 Task Allocation.....	19
2.2.2 Task Consensus.....	20
2.2.3 Consensus Based Auction Algorithm	25
Phase 1: The Auction Process.....	25
Phase 2: The Consensus Process.....	26
2.2.4 Consensus Based Bundle Algorithm	28
Phase 1: The Bundle Construction.....	29
Phase 2: Conflict Resolution.....	30
Scoring Functions	33
2.2.5 Task Allocation with Duo Cooperation Restraints	34
2.2.6 Task Allocation via Coalition Formation	36

2.3 Eusocial Animal Behaviours.....	38
2.4 Conclusions.....	42
Chapter 3: <i>Consensus Based Grouping Algorithm</i>	43
3.1 Problem.....	45
3.1.1 Single-Agent Task Assignment Problem.....	45
3.1.2 Multi-Agent Task Assignment Problem	46
3.1.3 Restricted Task Assignments	48
3.2 The Proposed CBGA	50
3.2.1 Local Data.....	50
3.2.2 Phase 1: Bundle Construction.....	52
3.2.3 Phase 2: Consensus	56
3.3 Performance	61
3.3.1 Methodology	61
3.3.2 Test Scenario.....	62
3.3.3 Multi-Agent Task Allocation.....	65
3.3.4 Restricted Task Assignments.....	78
3.4 Conclusions.....	83
Chapter 4: <i>Dynamic Multi-Agent Consensus Storage</i>	86
4.1 Problem.....	86
4.2 Algorithm.....	89
4.2.1 Local Data.....	89
4.2.2 Communication.....	92
4.3 Results.....	95
4.4 Conclusions.....	104
Chapter 5: <i>Biologically Inspired Improvements to the CBGA</i>	106
5.1 Problem.....	107
5.1.1 Team Improving Assignments	108
5.1.2 Task Quitting	110
5.2 Algorithm.....	110
5.2.1 Team Improving Assignments.....	111
5.2.2 Task Quitting	114
5.3 Performance	116

5.3.1 Test Scenario.....	116
5.3.2 Results.....	117
5.4 Conclusions.....	127
Chapter 6: <i>Conclusions and Future Work</i>	128
6.1 Conclusions.....	128
6.1.1 Key Contributions.....	128
6.1.2 Limitations	129
6.2 Future Work.....	129
Publications.....	132
References.....	133
Appendix A.....	143
A.1 Task Assignment and Consensus with the CBBA.....	143
A.2 Task Assignment and Consensus with the CBGA.....	147

List of Tables

Table 2.1: Consensus decision table for the CBBA (Choi et al., 2009).....	32
Table 3.1: Data stored on each agent i	51
Table 3.2: Data communicated by each agent i	51
Table 3.3: Winning agent matrix x_i for agent i storing the binary values for winning assignments between agents and tasks.....	52
Table 3.4: Icon key for agent path graphs.	63
Table 3.5: Winning assignment bids for the experiment displayed in Figure 3.10 where each task provides a fixed reward of $c_j = 100$ modified by the cost of travel calculated from the marginal score improvement (3.11).....	69
Table 4.1: Winning bid list and winning agent list used in the CBBA.....	89
Table 4.2: Winning bid matrix and winning agent matrix used for multi-agent assignments in CBGA.....	90
Table 4.3: Dynamic variable storage on agents, where X_{ikj} references the winning bid agent i believes agent k has made for task j	92
Table 4.4: Agent A1's winning bid matrix and the related communication message..	93
Table 4.5: Agent A1 able to add newly discovered agents to its knowledge base; a similar method can be used to add newly discovered tasks.....	94
Table A.1: Agent A1's initial winning agent list z_i and winning bid list y_i after the bundle construction phase but before any consensus.	144
Table A.2: Agent A2's winning agent list z_i and winning bid list y_i that is communicated to agent A1 and A3 after the bundle construction phase.	144
Table A.3: Agent A1's winning agent list z_i and winning bid list y_i after the second iteration of the bundle construction phase.....	145
Table A.4: Agent A2's winning agent list z_i and winning bid list y_i that is communicated to agent A1 and A3 after the second iteration of the bundle construction phase.	145

Table A.5: Agent A1's initial winning agent matrix x_i and winning bid matrix y_i after the bundle construction phase but before any consensus.	148
Table A.6: Agent A2's winning agent matrix x_i and winning bid matrix y_i that is communicated to agent A1 and A3 after the bundle construction phase.	148
Table A.7: Agent A1's winning agent matrix x_i and winning bid matrix y_i after the second iteration of the bundle construction phase.	149
Table A.8: Agent A2's winning agent matrix x_i and winning bid matrix y_i that is communicated to agent A1 and A3 after the second iteration of the bundle construction phase.	149
Table A.9: Agent A1's winning agent matrix x_i and winning bid matrix y_i after the third iteration of the bundle construction phase.	150
Table A.10: Agent A2's winning agent matrix x_i and winning bid matrix y_i that is communicated to agent A1 and A3 after the third iteration of the bundle construction phase. ...	150

List of Figures

Figure 1.1: ALFUS detailed model for agent autonomy (Huang et al., 2005) with an approximation on the autonomy level of the BigDog system and the Curiosity.	8
Figure 1.2: ALFUS summary model for defining autonomy levels in autonomous systems (Huang et al., 2005).	10
Figure 1.3: Hierarchical levels of autonomy algorithms (Lum, 2009).	11
Figure 1.4: Multi-agent system taxonomy (Farinelli et al., 2004).	12
Figure 1.5: Levels of social organisation for animal species.	14
Figure 1.6: The process of evolution by natural selection and random gene mutation.	15
Figure 2.1: Flowchart of basic consensus decision making process for a group of people.	22
Figure 2.2: Flowchart of basic consensus decision making for an agent in an agent based system.	23
Figure 2.3: Task allocation algorithm of the CBAA (Choi et al., 2009).	26
Figure 2.4: Consensus algorithm of the CBAA (Choi et al., 2009).	27
Figure 2.5: Task allocation algorithm for the CBBA (Choi et al., 2009).	30
Figure 2.6: Decentralised task elimination for agent i for the assignment and consensus of cooperative duo tasks (Choi et al., 2010).	35
Figure 2.7: Task allocation algorithm using a task quitting method for the allocation of bees throughout a hive (Johnson, 2009)	41
Figure 3.1: Bundle phase for CBGA.	56
Figure 3.2: Data sent and received between agent A and its neighbours, a link between two agents represents that those agents are within communication distance and thus can communicate data between each other.	57
Figure 3.3: Conflict resolution for the CBGA for multi-agent task.	58
Figure 3.4: Plot of agent paths and their assignments in X Y and Z.	63
Figure 3.5: Plot of agent paths and their assignments through time in X and Y.	64

Figure 3.6: Plot of agent paths and their assignments through time and position X.....	65
Figure 3.7: First experiment (left) has three agents completing fifteen tasks. Second experiment (right) has ten agents completing twenty tasks.....	66
Figure 3.8: First experiment (left) has four agents completing fifteen multi-agent tasks that require two agents each. The second experiment (right) has ten agents completing fifteen tasks that require three agents each.	67
Figure 3.9: Agent paths with position (X) over time (τ). Four agents complete three single-agent and three multi-agent tasks that require two agents each.	68
Figure 3.10: Agent paths with position (X) over time (τ). Eight agents complete two different types of multi-agent tasks. Task 1 requires three agents and task 2 requires six agents.	68
Figure 3.11: Comparison of average score and number of communication steps for consensus between the CBBA, CBGA and using both. Experiments contain twenty randomly generated tasks and multi-agent tasks require two agents.	69
Figure 3.12: Cross section of agents movement through time (τ) and the X axis to complete twenty multi-agent tasks. Each task requires two agents for completion.	70
Figure 3.13: Computational times for running each experiment in Figure 3.11 CBBA assigns only single-agent tasks, the CBGA assigns multi-agent tasks and mix requires both the CBGA and CBBA.	71
Figure 3.14: Percent score improvement by using the preferred duo cooperation (left) (Choi et al., 2010) and the CBGA (right) over the original CBBA where darker squares show lower or no improvement and lighter squares show a higher improvement.	72
Figure 3.15: Difference in percent score improvement by using the CBGA over the CBBA preferred duo cooperation where darker squares show a decline in score and lighter squares show an improvement in score.	73
Figure 3.16: Implementation of a sequential greedy algorithm to solve the multi-agent task assignment problem in a centralised system.....	74

Figure 3.17: Average assignment score achieved by the CBGA and the SGAMA completing 20 tasks with Nn agents (left) and 20 agents completing Nm tasks.....	75
Figure 3.18: Average assignment score achieved by the CBGA and the SGAMA when agent requirements L_j are increased with 10 agents and 20 tasks (left). Assignment of 10 agents to 20 tasks with each task requiring 10 agents (right).	76
Figure 3.19: Comparison of the average number of communication steps and number of bids required to come to a consensus as the number of agents required per task increases.	76
Figure 3.20: Effect on average communication steps and bid data as both the number of agents and required agents for multi-agent tasks increases.	77
Figure 3.21: Agent paths with position (X) over time (τ). First experiment (left) shows two different types of agents completing two different types of single-agent tasks. The second experiment (right) shows the same group of agents and tasks except the tasks are now multi-agent tasks requiring two agents each.	78
Figure 3.22: Two experiments both with multi-agent tasks that require two agents one of type A and one of type B.....	79
Figure 3.23: Comparison of average score and number of communication steps required to reach a consensus for tasks that require multiple different heterogeneous agents.	79
Figure 3.24: Agent's paths through time for 3 agent types (A,B,C) completing 3 different types of tasks (TA, TB, TC).....	80
Figure 3.25: Comparison of total score and number of communication steps for tasks that require previous tasks completed.....	82
Figure 3.26: Agents 'A' movement through time and the X axis. Tasks marked 'B' require the corresponding task 'T' to be completed first, similarly tasks marked 'C' require a corresponding task 'B' completed.....	82
Figure 4.1: The average amount of data sent in progressive steps through each iteration of the CBGA.....	96

Figure 4.2: Communication range limited to a distance of 1, overlapping circles and connected lines show connecting communication networks. Associated assignments for 10 agents completing 20 single-agent tasks (right).	97
Figure 4.3: Communication range limited to a distance of 2, connected communications shown by connected lines. Associated assignments for 10 agents completing 20 single-agent tasks (right).....	97
Figure 4.4: Communication range limited to a distance of 2.5, overlapping circles show connecting communication networks, blue link shows the networked agents. Associated assignments for 10 agents completing 20 single-agent tasks (right).	98
Figure 4.5: Communication range limited to a distance of 1, overlapping circles show connecting communication networks, blue link shows the networked agents. Associated assignments for 10 agents completing 10 multi-agent tasks where $L_j = 2$ (right). 99	99
Figure 4.6: Total Score and convergence time for consensus using the CBGA measured in the number of bid rounds required as the distance of communication increases.....	100
Figure 4.7: The effect on the number of communication steps and bids required in the CBGA for consensus as the distance of communication is increased.	100
Figure 4.8: Average data sent and received per agent to achieve a consensus for twenty tasks using the CBBA for single-agent tasks and the CBGA for two types of multi-agent tasks requiring two and three agents.....	101
Figure 4.9: Average data sent and received per agent for each assignment where a multi-agent task requiring three agents will require three assignments.....	102
Figure 4.10: The average number of conflicting bids per agent where a conflicting bid is defined as a bid for a task that is replaced by a better bid.	103
Figure 4.11: Comparison between the CBGA and the CBBA for duo tasks on the amount of data sent per agent to communicate an assignment at progressive iterations of each algorithm.	104

Figure 5.1: An example situation where individual assignment priority will result in a lower score than a team focused assignment.	109
Figure 5.2: Algorithm for determining valid task list h_{ij} in the CBGA.	112
Figure 5.3: Comparison of the bidding decision between individual focused bidding (left) and team focused bidding (right)	113
Figure 5.4: Task quitting algorithm for the CBGA.	115
Figure 5.5: Performance of the CBGA algorithm compared to the CBGA using Team focused bidding as the agent requirements (L_j) on tasks increases.	117
Figure 5.6: Performance with the result spread for the CBGA (left) and the CBGA with team focused bidding (right) with increasing agent requirement (L_j).	119
Figure 5.7: Assignment of 10 agents completing 20 tasks resulting in a score of 7295	119
Figure 5.8: Performance of the CBGA algorithm compared to the CBGA using task quitting where the number of agents is set to 10 (left) and when the number of agents is set to 8 (right) completing 20 multi-agent tasks.....	120
Figure 5.9: Performance of the CBGA using task quitting with the spread of results as the agent requirements (L_j) on tasks increases. Task quitting threshold δ set to 1.	121
Figure 5.10: Performance of the task quitting algorithm as the task quitting threshold δ is modified.	122
Figure 5.11: Performance of all variations of the CBGA with task quitting and team focused bidding, each simulation contains 10 agents and 20 tasks.....	122
Figure 5.12: Variance of results from assignments using the CBGA with both task quitting and team focused bidding.....	123
Figure 5.13: Assignment of 10 agents completing 20 tasks that require all 10 agents each the simulation setup is exactly the same as in Figure 5.7 but using task quitting to produce a score of 9391.....	124

Figure 5.14: Average assignment score achieved by the CBGA with biologically inspired improvements (CBGABI) and the SGAMA completing 20 tasks with Nn agents (left) and 20 agents completing Nm tasks.	125
Figure 5.15: Average assignment score achieved by the CBGABI and the SGAMA when agent requirements are increased with 10 agents and 20 tasks.	125
Figure 5.16: The effects of adding task quitting and team bidding to the CBGA for multi-agent tasks.	126
Figure A.1: Initial set up of the example simulation with a top down view of the agent starting positions and their communication network (left) and the view of task positions and their time windows (right).	143
Figure A.2: Initial assignments before any communication and consensus has taken place. Agent paths offset to allow easier viewing of agents on the same path.	144
Figure A.3: Assignments after the second iteration of the bundle phase. Agent paths offset to allow easier viewing of agents on the same path.	145
Figure A.4: Agent assignments at successive iterations of the CBBA. Agent paths offset to allow easier viewing of agents on the same path.	146
Figure A.5: Initial set up of the example simulation with a top down view of the agent starting positions and their communication network (left) and the view of task positions and their time windows (right).	147
Figure A.6: Initial assignments before any communication and consensus has taken place. Agent paths offset to allow easier viewing of agents on the same path.	147
Figure A.7: Assignments after the second iteration of the bundle phase. Agent paths offset to allow easier viewing of agents on the same path.	148
Figure A.8: Final assignment of agents, agent paths offset to allow easier viewing of agents on the same path.	150

Chapter 1: *Introduction*

This thesis investigates the problems with current task assignment and consensus algorithms for cooperative Unmanned Aircraft Systems (UAS). In particular, it addresses the limitations of consensus algorithms and develops extensions to existing algorithms to eliminate some of these limitations.

This chapter is an introduction to the main thesis. A brief background section describes aspects of autonomous systems and provides useful background information for the thesis. Some of the major contributions of this work are also presented along with the general layout and structure of the thesis.

1.1 INTRODUCTION

The subject of modern unmanned systems is challenging and engaging. Whilst much theoretical work from ground robotics can be directly applied to Unmanned Aerial Vehicles (UAV), there are still aspects that need to be accounted for as well as increased complexity. UAVs provide a larger scope for application than their ground counterparts which shows in their increasing use throughout the world (Fahlstrom & Gleason, 2012; Kreps & Kaag, 2012; Papadales & Downing, 2005). However, currently this is limited to military and research applications outside commercial airspace or with strict limits and surveillance. Unmanned systems encompass many areas of research such as machine learning, artificial intelligence, path planning, task allocation, control systems and many more.

UAVs are commonly visualised as robotic planes flying themselves around completing tasks on their own. In reality the operation of unmanned aircrafts around the world still requires significant human supervision. For an autonomous system to be beneficial it must be capable of operating in a wide variety of missions and be able to perform required tasks competently and consistently. For example, if the police force wants to use a UAV for surveillance of riots or crimes they cannot simply let a

UAV operate by itself (Gogarty & Hagger., 2008). Working in commercial airspace requires that all unmanned vehicles have a remote operator with a license to run the vehicle (McCarley & Wickens, 2005). Even then there are strict limits on its operation, but ignoring these points it would still be too dangerous for a UAV to run unsupervised given current knowledge. These missions are often complex with difficult operations due to the dynamic environments; a UAV must add no extra risk to any other users over and above that which a human pilot would have added.

The rise in the use of UAVs is becoming prevalent throughout the world. UAVs are finding valuable usage in performing military tasks that fall into the categories of the dull, dirty and dangerous (Schneiderman, 2012) (Hirschfeld et al., 1993) (Connelly et al., 2006). Dull missions can be thought of as those which require repetitive and tedious actions over long periods of time where human error can easily creep in from exhaustion. Dirty tasks will often require complicated processes and decision making that would be better suited to the precision of a computer. As expected some tasks contain potentially harmful activities that are dangerous in their nature or if performed incorrectly would be fatal to human pilots. UAVs take advantage of the better sustained alertness of machines over that of human pilots in addition to lower political and human cost if the mission fails. Lower risk and higher confidence in mission success are two strong motivators for continued expansion of UAS.

The applications for UAVs are numerous and cover a range of areas from military applications, scientific projects to commercial activities. As technology advances the future of UAVs looks increasingly towards civilian activities (Martinez-Val & Perez, 2009) (Campoy et al., 2009). Common applications include surveillance of power lines or pipes (Jiang et al., 2013) (Larrauri et al., 2013) (Dewi, 2005), disaster monitoring (Tuna et al., 2012) (Maza et al., 2011) and search and rescue operations (Tomic et al., 2012) (Zhao et al., 2012) (Lin & Goodrich, 2009). With

improvements in the cooperative abilities of UAVs there is a movement towards an increase in the use of multiple UAVs for complex tasks like tracking and surveillance (Li et al., 2012) (Chen & Wu, 2012) (Hirsch et al., 2012) (Hirsch et al., 2011). As the applications for UAVs increase so too does their need to cooperate to perform bigger and more complex tasks. Applications are wide and varied and can apply to both military and civil areas, many applications spanning both sections. For example a search and rescue application can be very useful for both military and civil areas. The monitoring of a disaster area for civil use can be used in military applications by monitoring a bomb site for example. A large portion of research into UAVs is being dedicated to increasing their cooperative abilities (Zhu et al., 2013) (Nakamura et al., 2013) to solve new complex problems or provide better and faster solutions to existing problems.

1.2 PROBLEM STATEMENT

Creating a UAV to cover all situations and problems is difficult due to hardware and software limitations (Pastor et al., 2007) and so it is far easier to use UAVs dedicated to solving a precise problem. A reconnaissance UAV needs to be lightweight and mobile whilst being able to carry state-of-the-art photography and video equipment (Kontogiannis & Ekaterinaris, 2013), limiting any additional equipment it can carry. However upon doing this the UAV's ability to solve a wide variety of tasks in a dynamic environment is reduced. With a diverse selection of UAVs that can form teams and work together to complete tasks the limitation of any one UAV can be mitigated. Using multiple UAVs will improve the efficiency with which a number of tasks can be performed by completing tasks in parallel. A system that allows heterogeneous agents to assign and complete tasks together increases the flexibility of the system, which is an aspect of producing higher autonomy (Müller, 2012).

Of particular interest within the area of UAV cooperation is the Task Assignment Problem (TAP) which assigns a finite number of agents to complete a finite number of tasks as efficiently as possible. This problem can be solved with a centralized or decentralized solution but current research investigates decentralized solutions that are more feasible for real world adoption. Many researchers have solved the TAP using auction algorithms (Sujit & Beard, 2007) (Hoeing et al., 2007) where agents make bids for tasks and receive assignments based on their bids by a single auctioneer. One such solution that makes use of auction algorithms is the Consensus Based Auction Algorithm (CBAA) (Choi et al., 2009). The CBAA brings agents to a consensus on the allocation of tasks by enforcing agreement upon the solution rather than the information set. Whilst task allocation for an individual agent is relatively simple, the difficulty comes with consensus between all agents when using a decentralized algorithm. The CBAA succeeds in giving a conflict free solution that has superior convergence and performance to other auction algorithms performing at a polynomial-time that scales well with the size of the network and/or the number of tasks. However the CBAA focuses only on single agents completing single tasks or in the case of the Consensus Based Bundle Algorithm (CBBA) a single agent completing multiple tasks. Research has been done on using the CBBA with heterogeneous agents to create heterogeneous teams that can perform complex missions in real-time dynamic environments (Ponda et al., 2010) (Whitten et al., 2011). An extension has already been made to the CBBA that accounts for cooperative tasks; however this extension is limited to ‘Required Duo Tasks’ (Choi et al., 2010) that require cooperation from two agent types. The work proposed in this research aims to create a theoretically unlimited requirement on cooperation. This research introduces multi-agent tasks that can be solved by multiple heterogeneous UAVs working together, where modifications must be made to current task allocation and consensus algorithms to account for these new constraints.

1.3 OBJECTIVES

This research will investigate the challenges associated with the assignment of multi-agent tasks to groups of cooperating agents. The objectives of this research are as follows:

- Investigate solutions to the multi-agent task assignment problem.
- Develop a solution to handle multi-agent task assignments with a focus on robustness, performance and scalability.
- Improve task assignment algorithms to function with complex task requirements such as equipment and task dependencies.
- Improve the existing data structure in the CBBA for use in a dynamic environment with heterogeneous agents
- Investigate the potential use of animal behaviours in multi-agent systems

The proposed algorithm will provide a solution to the assignment and consensus of a decentralised group of agents representing UAVs. The solution will provide a conflict free solution with a similar performance to the CBBA despite the additional complexity of the assignments. The performance of the algorithm can be measured from the quality of assignments created based on a scoring function as well as the computational run time of the algorithm. Furthermore the algorithm will provide a framework for introducing complex requirements on tasks including equipment and task dependencies. Finally by investigating the cooperative behaviours in animal species improvements can be created to increase the quality of assignments and remove some issues found in the assignment algorithms.

1.4 DOCUMENT LAYOUT

This thesis is divided into several chapters. The rest of Chapter 1 outlines the research area and some useful background knowledge. Chapter 2 describes the previous work and knowledge relevant to the research topic including the Consensus Based Auction and Bundle Algorithm which are the founding algorithms for this

work. Chapter 3 details the development and results of the Consensus Based Grouping Algorithm developed from the Consensus Based Bundle Algorithm to achieve greater cooperative behaviours in performing complex tasks. This chapter also includes additional limitations on the task requirements including equipment and task dependencies. During the development of the grouping algorithm a new dynamic bid storage system is developed to reduce communication and increase consistency between UAVs, this research is presented in Chapter 4. Biologically inspired improvements are made to the CBGA to improve its performance, the development and results of these improvements are documented in Chapter 5. Finally Chapter 6 closes the report with comments about the work so far, and the future work and applications this research can lead to.

1.5 BACKGROUND

To fully understand this research it will be necessary to include some background information on relevant topics. The topics are required to understand the literature, current work and the research.

1.5.1 Unmanned Aerial Vehicles

A UAV is an aircraft that flies without a human crew on board. The abbreviation UAV has been expanded in some cases to Unmanned Aircraft Vehicle System (UAVS). In the United States, the Federal Aviation Administration has adopted the generic class UAS to reflect the fact that these are not just aircraft, but systems, including ground stations and other elements. To distinguish UAVs from missiles, a UAV is defined as a reusable, un-crewed vehicle capable of controlled, sustained, level flight. This definition separates them from cruise missiles, which are not considered UAVs, because, like many other guided missiles, the vehicle itself is a weapon that is not reused.

Historically, UAVs were simple drones (remotely piloted aircraft), but autonomous control is increasingly being employed in UAVs. Their operations are generally limited to military applications primarily due to limits and regulations on their use in commercial airspace (Dalamagkidis et al., 2012). Currently, military UAVs perform reconnaissance as well as attack missions. While many successful drone attacks have been reported, they are also prone to collateral damage and/or erroneous targeting, as with many other weapon types. UAVs are also used in a small but growing number of civil applications but are often limited to private land and outside of public airspace.

Currently, common UAS have basic autonomous flight capabilities such as waypoint following (Chao et al., 2010), altitude and airspeed hold, and automated launch and retrieval (Garratt et al., 2007). Although the capabilities of these systems are formidable, the key distinction is that these are integrated systems rather than merely aircraft. They require a support infrastructure which contains ground stations, communication links, and human operators. Often the system requires at least two human operators for flight operations: one to manage flight paths and the avionics systems, and another to operate the camera system. Personnel required for launch, retrieval, and ground tasks may further increase this number. Although these systems succeed in shielding the human operator from the dangers present in these missions, the amount of human interaction required for these autonomous systems exceeds that of their manned counterparts.

1.5.2 Autonomous Systems

The majority of this work falls in the category of autonomous systems. This is an ambiguous term as it can refer to certain types of networks or indeed any system which has decision making abilities. This thesis works with the definition of autonomous systems for robotics where autonomous is defined as operating without

outside control or existing independently. These systems contain one or more robotic devices such as vehicles, aircraft or simpler devices like arms. Within the system these devices are referred to as agents. For these agents to be considered autonomous they are expected to perform a desired task in potentially dynamic environments without continuous human guidance. There are different levels of autonomy, for example, some factory robots are considered autonomous but within the strict confines of their environment resulting in what would be considered a low level of autonomy. Compared to the high level of autonomy the Mars rover ‘Curiosity’ has, which is necessary due to the delay between the system and the controller; this is more important for tasks where speed is essential and the reaction time of controllers is not adequate, as experienced when the rover first landed on Mars (Cruzen & Thompson., 2013).

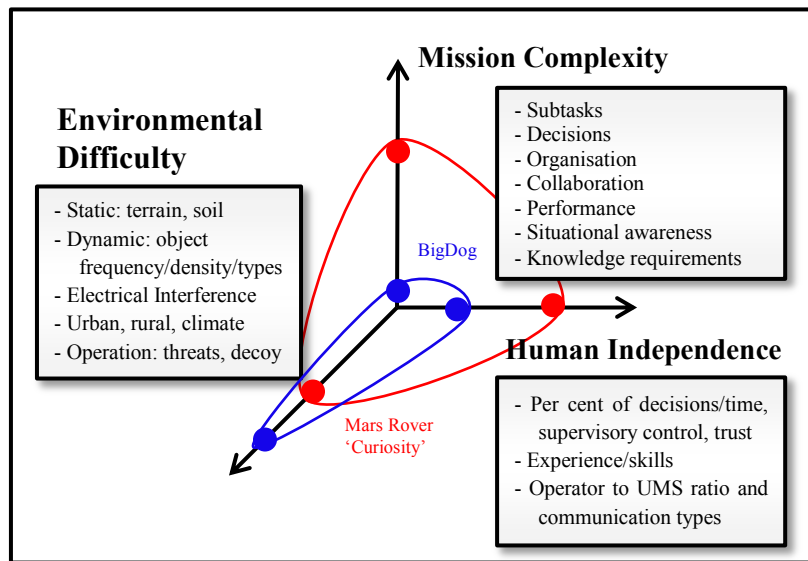


Figure 1.1: ALFUS detailed model for agent autonomy (Huang et al., 2005) with an approximation on the autonomy level of the BigDog system and the Curiosity.

As there is such a broad range of autonomy levels it is useful to define a method to determine the approximate autonomy of a given system. The autonomy levels for unmanned systems (ALFUS detailed model) (Huang et al., 2005), takes into

consideration the mission complexity, environmental difficulty and human independence as shown in Figure 1.1. Task complexity and adaptability to environment are among the key aspects as well as the collaboration with human operators, such as their levels of involvement and types of interaction. It also takes into account performance factors (Huang et al., 2005) including mission success rate, response time and precision. Figure 1.1 plots a possible autonomy level for the Mars rover Curiosity and the rough terrain quadruped robot BigDog. Curiosity would score highly in all areas requiring many autonomous systems to function, even simple tasks become more complex when commands take over 2.5 hours to receive and acknowledge (Lutz, 2011). Comparatively the BigDog system is overall less autonomous but contains higher levels of autonomous algorithms for navigating difficult and dynamic terrain, such as path planning, navigation and stabilising algorithms (Wooden et al., 2010). The overall concept for measuring each attribute is shown in Figure 1.2, where as a system relies less on human supervision and is able to deal with dynamic environments and missions, its level of autonomy increases.

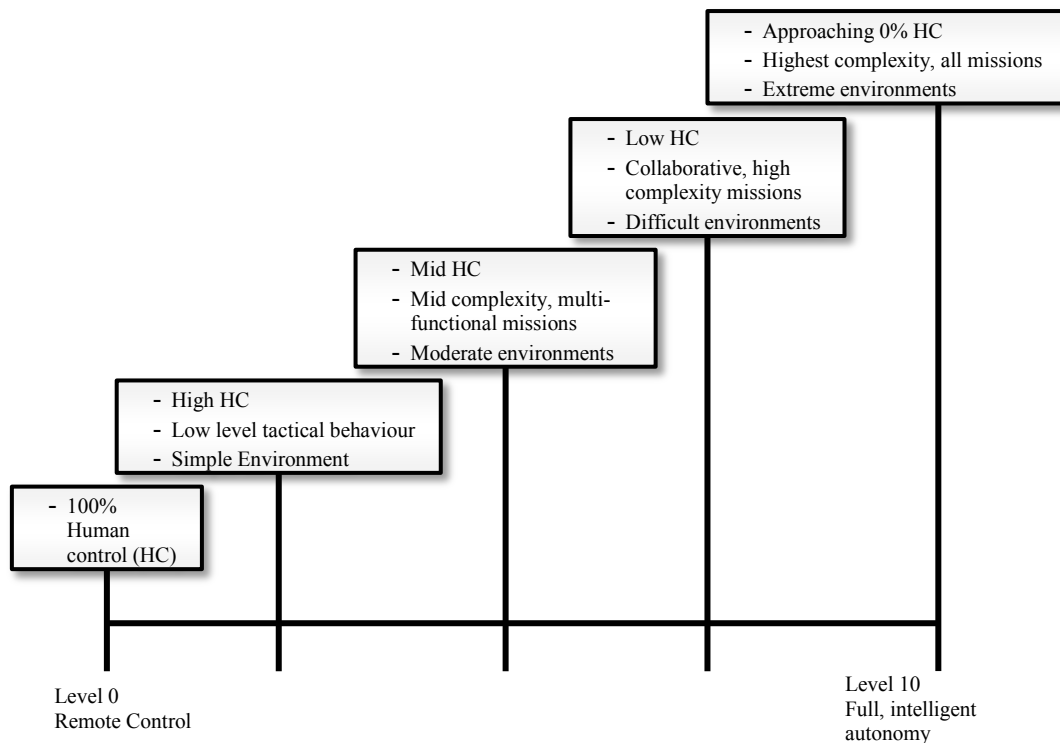


Figure 1.2: ALFUS summary model for defining autonomy levels in autonomous systems (Huang et al., 2005).

As well as the overall autonomy of a system, the underlying algorithms that control the agents can be similarly classified. Figure 1.3 shows the three levels for autonomy algorithms; strategic, tactical and dynamics and control (Lum, 2009). The strategic phase usually deals with tasks such as mission planning and task allocation. These algorithms are considered “low bandwidth” because compared to the other levels they run infrequently. The tactical phase deals with specialised responsibilities dealing with short term goals for instance path following or target observation. Finally the dynamics and control level deals with the inner control systems that often run continuously such as state stabilisation, an autonomous aircraft may need to constantly adjust its flight equipment to account for changing winds. Any high autonomy agent will require algorithms from all three levels. This hierarchical

structure of autonomy and the challenges it involves has been studied by groups like Passino et al (Antsaklis et al., 1989).

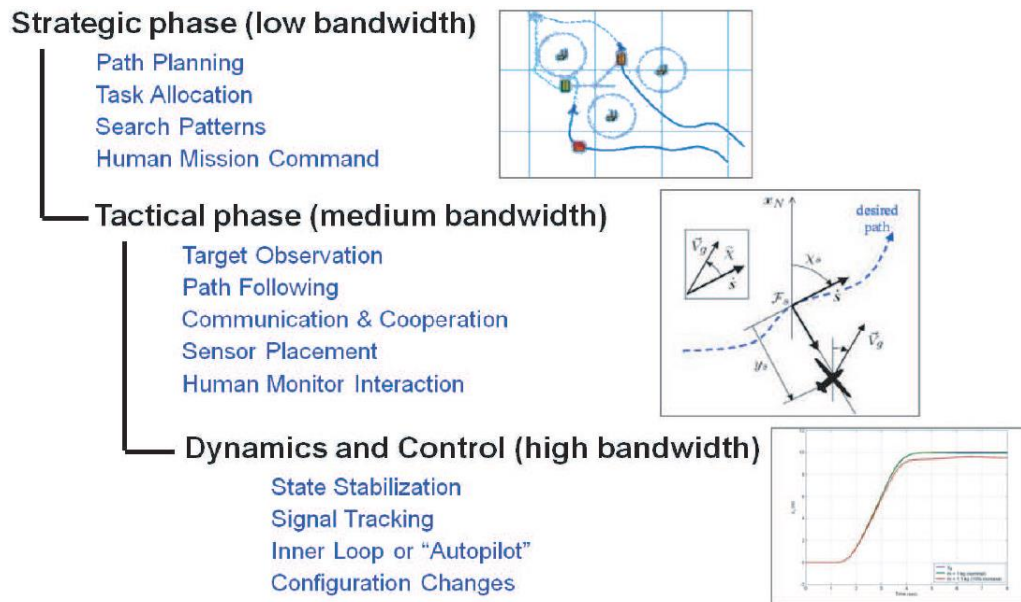


Figure 1.3: Hierarchical levels of autonomy algorithms (Lum, 2009).

Often, the terms “autonomous systems” and “autonomous algorithms” are used inter-changeably. The distinction made in this work is that autonomous algorithms are the routines which manage specific tasks without human interactions and autonomous systems are comprised of many sub-systems including hardware and autonomous algorithms. The autonomous system may refer to a single actual agent or the entire team and infrastructure to manage a group of agents.

1.5.3 Multi-Agent Systems

A multi-agent system can simply be a system that contains multiple intelligent agents. This field has become a very important research area within artificial intelligence and robotics, allowing the development and analysis of sophisticated AI problem-solving and control architectures (Innocenti et al., 2010). With the increasing

complexity and intelligence in robotics, cooperation has become a fundamental feature of multi-agent systems. The work reported in this thesis is focused on in cooperative systems; Figure 1.4 shows a proposed classification of multi-agent systems (Farinelli et al., 2004). It categorises a multi-agent system based on how coordinated the system is, showing how much coordination is achieved by the system.

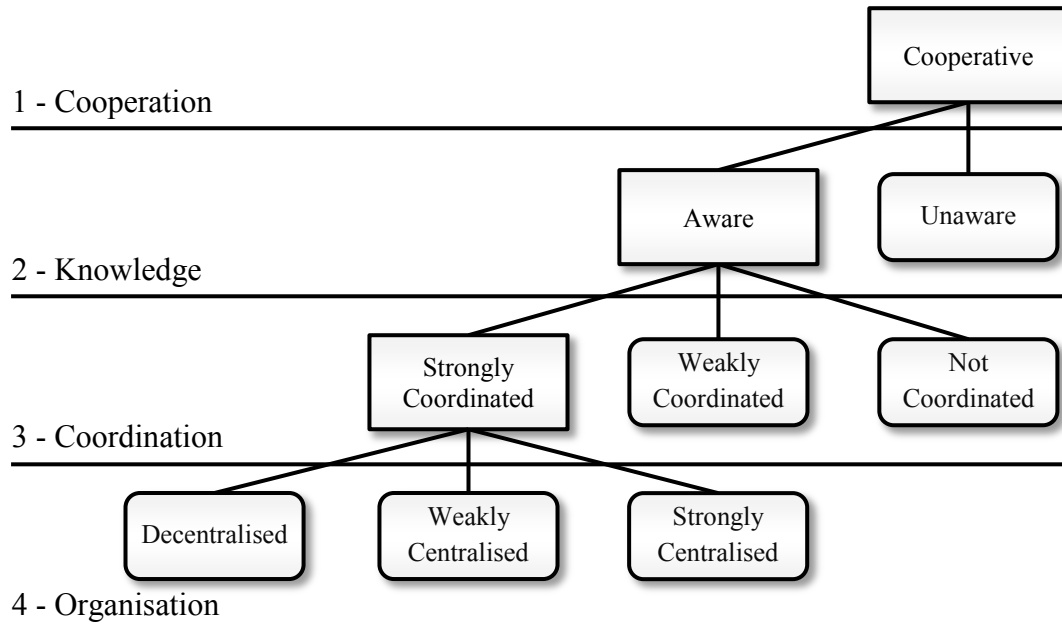


Figure 1.4: Multi-agent system taxonomy (Farinelli et al., 2004).

The first level, cooperation, distinguishes between cooperative systems and non-cooperative systems. It defines whether the agents cooperate to accomplish a task. A cooperative system is one where the agents work together to complete a global task. Competition is usually merged in with cooperative systems when using multi-agent systems.

The second level deals with agent knowledge; this defines the type of knowledge agents have about their team mates. Aware agents have some knowledge about the other agents in their team, for instance, their location or current activity. Unaware agents have no knowledge about their team mates and act only on their own knowledge. Unaware systems are simpler than aware systems, however it must be

noted that knowledge is not equivalent to communication. A multi-agent system can be aware even though there is no direct communication among the agents.

Coordination describes the actions of agents when cooperating on a task. Strong coordination means agents act to achieve a task based on the actions of the other agents such that the whole group becomes coherent. There are different ways the agents can take into account the actions of the others and it comes down to the cooperation protocol which is a set of rules that the agents follow in order to interact with each other in the environment. Weak coordination will produce agents that act to achieve the team goal but without considering what the other agents are doing.

The final level is the organisation level, which comprises of the structure and organisation of the multi-agent system. A centralised system has a leader that organises the other agents to complete the goal; the other agents act in accordance with what the leader tells them to do. A weak centralised system still uses a leader to complete the goal however in this system the leader can change or have multiple leaders. A distributed or decentralised system is comprised of agents that together organise themselves, and there is no central area where all decisions are made.

This research is about creating cooperative, coordinated and, most importantly, decentralised agents that can solve the problems provided and come to a conflict free consensus that is focused on maximising the group performance. Additionally this research will explore the effects of differing levels of knowledge on the agents ranging from full global knowledge and communication to only allowing local knowledge, with information communicated between agents.

1.5.4 Eusociality

Evolution is a process that gradually changes species over millions of years to adapt and survive in their environment. The goal of any animal species is to survive and reproduce, where evolution refines each species to complete those goals by

removing weaknesses and adapting their bodies and behaviours to the environment. With the power of adaption that comes from evolution scientists have made use of many areas of biology to create robust and adaptive systems, for instance using evolution itself in the form of genetic algorithms for machine learning (Goldberg & Holland, 1988) or the creation of robotics inspired by biology (Liu & Sun, 2012).

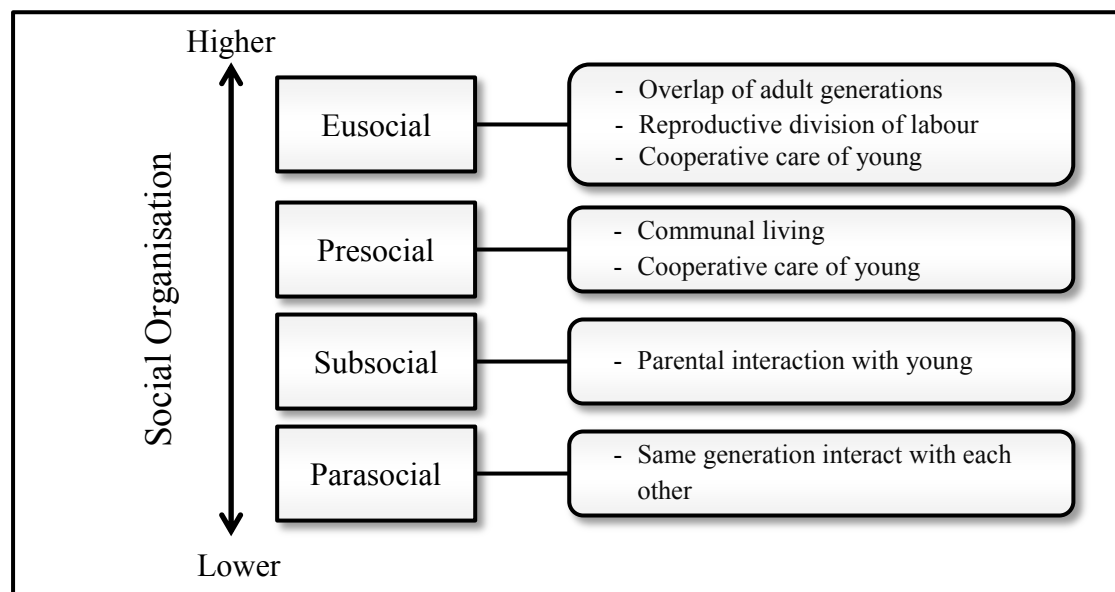


Figure 1.5: Levels of social organisation for animal species.

The emergence of cooperation and altruism were once thought of as impossible through evolution by natural selection, but scientists have shown conditions under which reciprocity cooperation can evolve (Axelrod & Hamilton, 1981). With the powerful filtering of weaknesses and strengths by evolution cooperative animal species make excellent tools to extract cooperative algorithms or beneficial concepts. Social animals that often show aspects of cooperative behaviour can be split into different levels of sociality; eusocial, presocial, subsocial and parasocial.

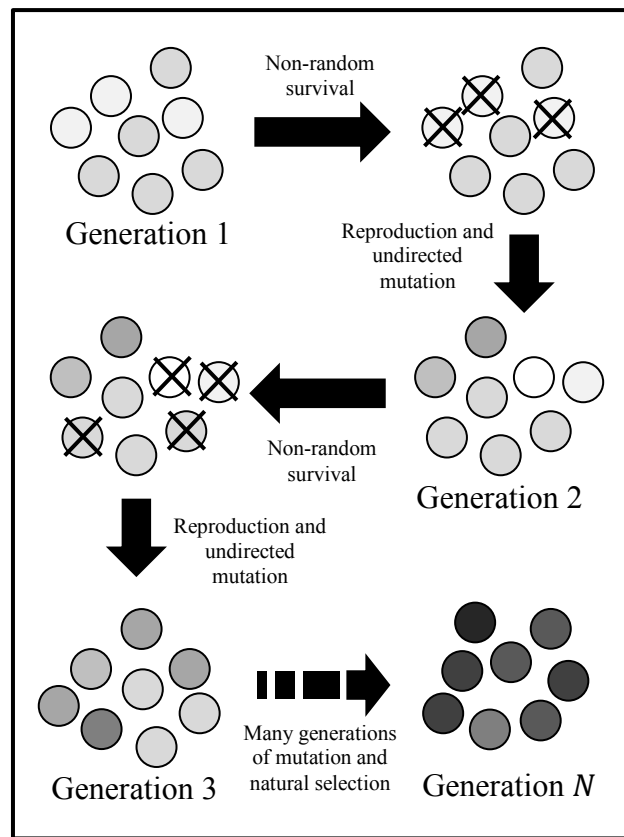


Figure 1.6: The process of evolution by natural selection and random gene mutation.

Figure 1.5 shows the various levels of social organisation with humans categorised as presocial and being preceded by various types of insects at the eusocial level. Eusociality is defined by a number of characteristics; cooperative care of the young, overlapping generations of adults within the group and a division of labour into reproductive groups and non-reproductive groups (Plowes, 2010). For a species to be considered eusocial they must possess each of these characteristics with many species failing at the division of labour. Eusocial species make excellent resources for studying cooperative animal behaviours because of the method of their reproduction and subsequent evolution. Species evolve by passing on their genes where a gene is the biological code for a characteristic that an animal possesses, such as an insect's colour. Should that specific characteristic aid the animal in its survival then it is more likely to survive and reproduce thus propagating that gene down the generations until

it becomes a predominant gene in the species as shown in Figure 1.6. For non-eusocial species the genes that are passed on are stored in each individual animal creating a selective pressure towards individual survival. Furthermore, whilst these species may cooperate each individual does so for selfish reasons, because cooperation increases the individual's odds for survival. In comparison eusocial species have their genes passed on only by one member in a group such as the queen ant in an ant colony. This form of reproduction means that for the majority of individuals, their only goal is the survival of the colony and specifically the queen; an individual ant is willing to sacrifice its life if it means the queen survives and reproduces.

When analysing cooperative behaviours it seems that eusocial species would provide more suitable results, as the only concern of a group of cooperative robots is achieving the goal set out, and that sacrifice of individual performance for the greater good of the team is a desirable trait.

Chapter 2: Literature Review

The work presented in this thesis covers many fields of research. Many of these subjects are well studied and extensive literature exists regarding relevant methods and techniques. This chapter highlights some of the related work done by others in specific areas related to this research. The limitations of these approaches are not detailed here. Instead, as the corresponding topic is discussed in the thesis, the deficiencies of these methods are outlined which illustrate the advantages and thus, the need for the methods developed in this dissertation.

2.1 CENTRALISED AND DECENTRALISED SYSTEMS

Centralisation comes from network theory; centralised networks are a type of network where all users connect to a central server, which is the acting agent for all communications. When applied to multi-agent systems all relevant sensors and agent information required in decision making are sent to a central hub or a single agent acting as the leader of the group. This central point in the multi-agent system collects the relevant situational awareness of all agents such as location or health status. With this information decisions can be made and communicated to the rest of the agents in the network. Task allocation deals strictly with centralised systems.

Centralisation makes decision making easier with complete system knowledge but in reality there is far too much information required. In practice, consideration of factors such as communication bandwidth, interference and delay is needed. This leads to using decentralised systems where the decision making is split across all agents. Each agent in the system will make its own decisions rather than a collective leader or central hub. However, decentralised systems must solve another problem: consensus. With each agent making its own decisions it is very easy for conflicts to arise as multiple agents attempt the same task. To avoid conflict in the system consensus must be reached between the agents as to who will do which tasks.

Within the field of robot formation control, the team formation can be controlled as a centralised system where there is monitoring and control of all robots to place them in specifically desired positions, or by a decentralised system when there is no supervisor and feedback is only detected by the relative position of each robot in respect to their neighbours. The centralised formation control can be a good strategy for a small team of robots, when it is implemented using a single computer and a single sensor to monitor and control the whole team. However, when considering a team with a large number of robots, the need for a greater computational capacity and a large communication bandwidth could make it advisable to use a decentralised formation control. Yamaguchi et al. present a distributed control scheme and shows simulations for final static formations (Yamaguchi et al., 2001). Fierro et al. proposes a hierarchical control structure that allows the switching of controllers in order to have a stable formation, based on sensing their relative positions to neighbouring robots, under a strategy of distributed control (Fierro et al., 2002).

Research by Claes and Holvoet shows that with a multi-agent exploration model, the overhead introduced by the decentralised system is higher (Claes & Holvoet, 2011). When comparing communication in decentralised and centralised systems this is often to be expected. The researchers argue that a decentralised system offers benefits of scalability. Even though the total message count in the decentralised system is higher than that of the centralised one, it spreads the communication overhead more evenly over the system. The centralised system, on the contrary, focuses all messages along with most computations in one central location. A decentralised method allows more freedom and autonomy for multi-agent systems and gives a robust solution that is required when dealing with autonomous multi-agent systems.

2.2 TASK ALLOCATION AND CONSENSUS

2.2.1 Task Allocation

With an increasing focus on the cooperative use of multi-robot systems, multi-robot coordination has received significant attention. Cooperative multi-robot systems allow for the execution of an increased number, variety and complexity of tasks. With this the importance of multi-robot task allocation (MRTA) has emerged. The advantages of self-organising groups of robots have led to MRTA becoming a key research issue in its own right. As cooperative multi-robot systems are created, the question “which robot should execute which task?” is inevitably asked (Bellingham et al., 2001).

The assignment problem or task allocation problem is a combinatorial optimisation problem that attempts to find the least-cost solution between two disjoint sets (Lo, 1998). In its general form, the problem is as follows:

“There are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the total cost of the assignment is minimized.”

There is a set of agents $I \equiv \{1, \dots, N_n\}$ and a set of tasks $J \equiv \{1, \dots, N_m\}$. An agent has a cost associated with it for completing each task. Let C_{ij} be the non-negative cost of assigning the i^{th} agent to the j^{th} task. The objective is to assign each task to one agent in such a way as to minimize the overall cost of completing all the tasks. Define a binary variable X_{ij} where $X_{ij} = 1$ to indicate agent i is assigned to task j . Otherwise $X_{ij} = 0$. Then the total cost of the assignment is equal to $\sum X_{ij} * C_{ij}$

C_{ij} for $i = 1$ to $n, j = 1$ to m . A valid task allocation must satisfy the following constraints:

- A task allocation must be correct. For each agent $i \in I$ must be assigned to no more than one task.
- A task allocation must be complete. For each task $j \in J$ must have exactly one agent assigned.

For an assignment to be efficient the task allocation must be valid and the cost is minimized, i.e.

$$Cost = \min(\sum X_{ij} * C_{ij} \text{ for } i = 1 \text{ to } n, j = 1 \text{ to } m) \quad (2.1)$$

Once the assignment is solved, agents will be assigned tasks such that the cost of completing each task is minimised. Many solutions exist for the task allocation problem (Wright, 1990) (Burkard et al., 2009) including practical solutions for use in UAVs (Jin et al., 2003) (Richards et al., 2002), however these solutions are centralised requiring a central agent or hub to perform the assignments. In this way, solving the task assignment problem for a decentralised system creates additional problems.

2.2.2 Task Consensus

Task allocation requires a central hub or a designated leader in order for a group of agents to decide on the correct allocation. For the task allocation problem to be solved in a decentralised system where each agent makes their own decisions about their assignments, it requires that agents come to a consensus on assignments. Task consensus is the process of agents communicating their desired assignments to each other and agreeing or coming to a consensus on the correct assignments.

Although group consensus can be found in many social animal species (Conradt & Roper, 2005), coming to a consensus is a difficult process even for humans who have the ability to easily adapt to unforeseen problems. Consensus decision-making is a group decision making process that attempts to provide a unanimous decision for all participants of the group. There are many different methods groups can use to come to a consensus (Schweiger et al., 1986) but the following concepts form some unifying principles for consensus decision making.

Agreement Seeking

Consensus decision-making is a process that seeks agreement across the group. Individuals may initially have different choices but must ultimately agree on one decision through the process of consensus decision-making. Failure to unanimously agree on a decision would be defined as a conflict, whereas the decision making process should come to a conflict free solution.

Inclusive

All members of the group should provide input into the decision making process such that conflicting information or decisions can be removed or dealt with.

Collaborative

Consensus decision-making is a collaborative process where the group comes to an agreed decision. All members of the group contribute towards the decision making process without one central leader making decisions for the group.

Cooperative

Participants in the consensus process should attempt to reach the best decision possible for the group taking into account information from all members. At times the best decision for the group may not be the optimum decision for an individual.

Although the process of consensus between two humans is complicated, it can be broken down into some key steps. These basic steps provide a method for formulating the decision making process as seen in Figure 2.1.

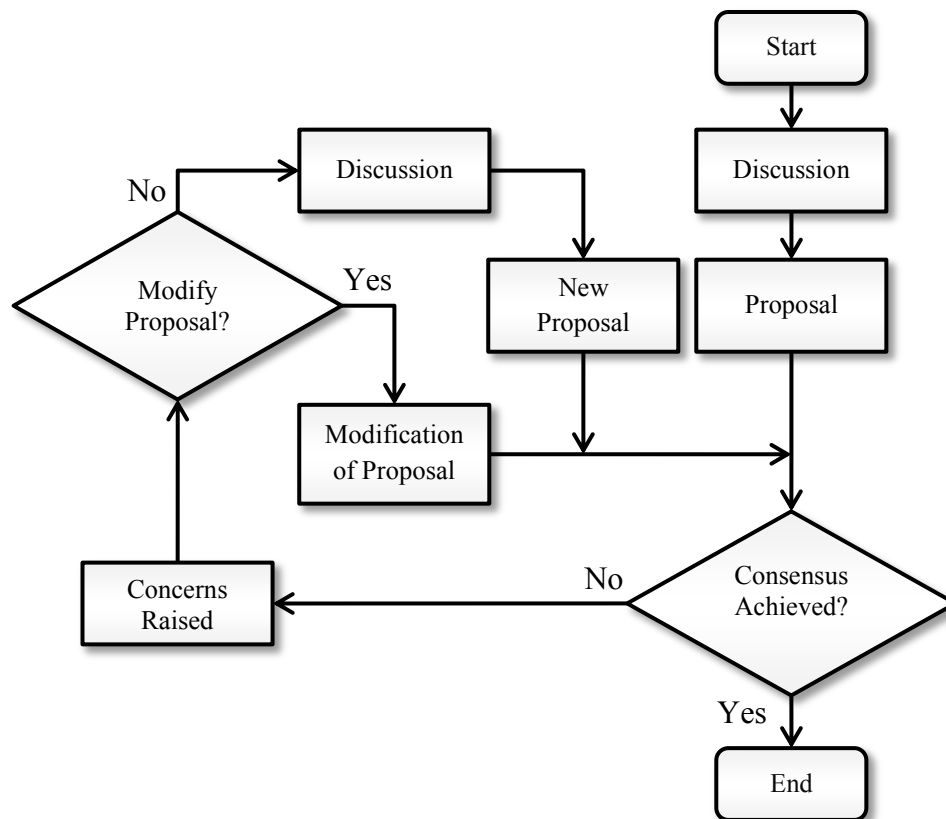


Figure 2.1: Flowchart of basic consensus decision making process for a group of people.

Discussion of the problem: The problem is discussed with the goal of identifying the current situation; the goal is for each member involved to gather as much information as they can such that they can form their own decisions on the matter.

Formation of a proposal: Based on the discussion a formal decision is proposed either for the group as a whole or for a specific individual's decision.

Call for consensus: Each member determines whether they agree with the proposed decision.

Identification and addressing of concerns: If a consensus is not achieved problems with the proposal are raised and points of conflict explained.

Modification of the proposal: Depending on the degree of conflict with the original decision either a modification of the decision can be proposed or the group rejects the decision and returns to discussion of the problem.

Although the decision making process between a group of humans is more complicated, containing much back and forth discussion, the basic idea can be translated into an agent based group decision making process. The key steps shown in Figure 2.2 outlines what stages must be incorporated into an agent based consensus.

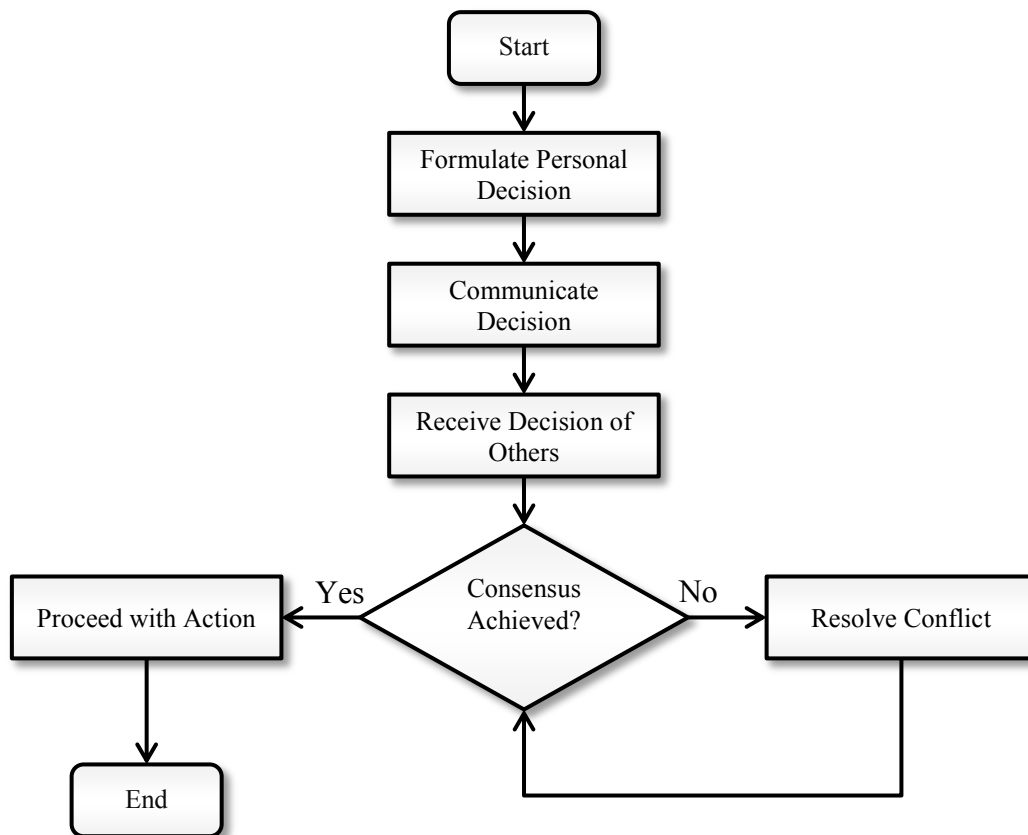


Figure 2.2: Flowchart of basic consensus decision making for an agent in an agent based system.

In multi-robot systems a decentralised solution for the task allocation problem would provide a more suitable solution for a real world application. This is even more valid for UAS where operational distances can cover large areas that could leave a central hub out of communication range or provide a single point of failure in a leader system. A decentralised solution for the MRTA would provide a robust and dynamic solution for a self-organising autonomous system. For decentralized systems, cooperating agents often require a globally consistent situational awareness (Ren et al., 2007). In a dynamic environment with sensor noise and varying network topologies, maintaining a consistent situational awareness throughout the group can be very difficult. Consensus algorithms are used in these cases to enable the group to converge on some specific information set before generating a plan (Beard & Stepanyan, 2003). Examples of typical information sets could be detected target positions, target classifications, and agent positions. Various consensus approaches have been shown to guarantee convergence over many different dynamic network topologies (Hatano & Mesbahi, 2004) (Wu, 2006) (Tahbaz-Salehi & Jadbabaie, 2006).

A variety of consensus based task allocation algorithms have been developed that provide provably conflict free solutions with superior convergence properties and performance (Choi et al., 2009). Further, these algorithms have been extended to provide robust solutions for specific situations and requirements (Ponda et al., 2010) (Di Paola et al., 2011) such as time constraints (Mercker et al., 2010) and dynamic uncertain environments (Bertuccelli et al., 2009). The consensus based algorithms provide excellent frameworks to develop solutions to the multi-agent task assignment problem.

2.2.3 Consensus Based Auction Algorithm

The Consensus Based Auction Algorithm (CBAA) solves single assignment problems by using both auction and consensus algorithms in a decentralized system (Choi et al., 2009). The algorithm contains two phases that alternate until assignments and consensus are achieved. The first phase of the algorithm is the auction process, while the second is a consensus algorithm that is used to converge on a winning solution. The CBAA, by iterating between the two phases can exploit the benefits of both auction and consensus algorithms. Robustness and computational efficiency are achieved from the auction algorithm whilst the decentralized consensus algorithm can exploit network flexibility and converge on a conflict free solution. The CBAA has been shown to provide a conflict free, feasible solution, which previous algorithms were unable to account for.

Phase 1: The Auction Process

The first phase of the algorithm is the auction process. Here, each agent i places a bid on a task j asynchronously with the rest of the agents. All agents store and update 2 vectors of length N_m where N_m is the number of tasks in the simulation, both are initialized as zero vectors. The first vector x_i records the task list for agent i where $x_{ij} = 1$ if agent i has been assigned to task j , and 0 if not. The second vector is the winning bids list y_i which keeps an as up-to-date as possible estimate of the highest bid made for each task thus far, this vector is used primarily in phase 2. Using $c_{ij} \geq 0$ as the bid that agent i places for task j , and $h_i \in \{0,1\}^{N_m}$ be the availability vector whose j^{th} entry is 1 if task j is available to agent i . The list of valid tasks h_i can be generated using

$$h_{ij} = \text{II}(c_{ij} > y_{ij}), \forall j \in J \quad (2.2)$$

where $\Pi(\cdot)$ is the indicator function that is 1 if the argument is true and 0 otherwise. Algorithm 1 in Figure 2.3 shows the procedure of agent i 's phase 1 at iteration t where one iteration consists of a single run of phase 1 and phase 2. With a decentralized system each agent's iteration count can be different allowing each agent to run with different iteration periods. An unassigned agent i , which can be defined as having $\sum_j x_{ij} = 0$, first computes a valid task list h_i . If there are valid tasks, it then selects a task J_i giving it the maximum score based on the current list of winning bids (line 7 of algorithm 1, Figure 2.3), and updates its task x_i and winning bids list y_i accordingly. If it is the case that an agent has already been assigned a task $\sum_j x_{ij} \neq 0$, this selection process is skipped and the agent moves to phase 2.

Algorithm 1: CBAA Phase 1 for agent i at iteration t

```

1: procedure SELECT TASK ( $c_i, x_i(t-1), y_i(t-1)$ )
2:    $x_i(t) = x_i(t-1)$ 
3:    $y_i(t) = y_i(t-1)$ 
4:   if  $\sum_j x_{ij}(t) = 0$  then
5:      $h_{ij} = \Pi(c_{ij} > y_{ij}(t)), \forall j \in J$ 
6:     if  $h_i \neq 0$  then
7:        $J_i = \operatorname{argmax}_j c_{ij} \cdot h_{ij}$ 
8:        $x_{i,J_i}(t) = 1$ 
9:        $y_{i,J_i}(t) = c_{i,J_i}$ 
10:    end if
11:  end if
12: end procedure

```

Figure 2.3: Task allocation algorithm of the CBAA (Choi et al., 2009).

Phase 2: The Consensus Process

The second phase of the CBAA is the consensus section of the algorithm. Here agents make use of a consensus strategy to converge on the list of winning bids and use that list to determine the winner in the bidding. This allows agents to converge on a conflict free solution over all tasks.

Phase 2 involves communicating the winning bid list for an agent to all other agents within communication range. $G(\tau)$ is a symmetric adjacency matrix showing communication links between agents where $g_{ik}(\tau) = 1$ if a link exists between agents i and k at time τ , and 0 otherwise. Agents i and k are said to be neighbours if such a link exists. It is assumed that every node has a self-connected edge; in other words, $g_{ii}(\tau) = 1, \forall i$.

During each iteration of phase 2 of the algorithm, agent i receives the list of winning bids y_i from every neighbour in range. The procedure of phase 2 is shown in algorithm 2 (Figure 2.4) when agent i 's t^{th} iteration corresponds to τ in real time. The consensus is performed on the winning bids list y_i based on the winning bids lists received from each neighbour y_k for all agents k such that $g_{ik} = 1$ in a way that agent i replaces y_{ij} values with the largest value between itself and its neighbours (line 4). Additionally an agent will lose its assignment if it finds that it is outbid by others for the task it had selected (line 6).

If two agents place the same winning bid for a task the winner cannot be determined randomly because each agent decides the winner independently and knowledge must be coherent across the group. A number of solutions exist for this problem; one such solution is to communicate an agent's unique identification number along with the bid data and using it to break any ties.

Algorithm 2: CBAA Phase 2 for agent i at iteration t

```

1: SEND  $y_i$  to  $k$  with  $g_{ik}(\tau) = 1$ 
2: RECEIVE  $y_k$  from  $k$  with  $g_{ik}(\tau) = 1$ 
3: procedure UPDATE TASK( $g_i(\tau), y_{k \in \{k | g_{ik}(\tau)=1\}}(t), J_i$ )
4:    $y_{ij}(t) = \max_k g_{ik}(\tau) \cdot y_{kj}(t), \forall j \in J$ 
5:    $z_{i,J_i} = \operatorname{argmax}_k g_{ik}(\tau) \cdot y_{k,J_i}(t)$ 
6:   if  $z_{i,J_i} \neq i$  then
7:      $x_{i,J_i}(t) = 0$ 
8:   end if
9: end procedure

```

Figure 2.4: Consensus algorithm of the CBAA (Choi et al., 2009).

The CBAA converges on a conflict free solution to the single assignment problem with provable score performance of at least 50% of the value of the optimal solution (Choi et al., 2009). The optimal score is the maximum score achievable from valid assignments of agents to tasks. With perfect information the optimal score for the single-agent task assignment problem can be calculated using the implicit coordination algorithm (Alighanbari, 2004). Assuming agents have accurate knowledge of their situational awareness the CBAA provides the same performance score of a sequential greedy algorithm a centralised solution for the task assignment problem.

2.2.4 Consensus Based Bundle Algorithm

The major downside to the CBAA is that whilst at a specific time in the simulation each agent can select the optimal task for it to complete, it does not take into account future selections. When a number of tasks are located close to each other a single agent can perform all the tasks rather than sending an agent to each task. Researchers addressed the problem by grouping assignments into bundles for bidding (Shehory et al., 1998) (Berhault et al., 2003) (Andersson et al., 2000) (De Vries & Vohra, 2003) providing the multi-assignment problem where each agent bids for multiple tasks. Each assignment combination or bundle was treated as a single item for bidding which led to complicated winner selection methods. The CBAA was extended to the multi-assignment problem developing the Consensus Based Bundle Algorithm (CBBA) (Choi et al., 2009). In the CBBA each agent has a list of tasks potentially assigned to it, but the auction process is carried out at the task level rather than at the bundle level as previous algorithms had been. Similar to the CBAA the CBBA contains two distinct phases for controlling the allocation and consensus of tasks.

Phase 1: The Bundle Construction

During the first phase an agent internally builds up a single bundle containing all the tasks it plans to complete and updates it as the assignment process progresses. Each agent continually adds tasks to its bundle until it is incapable of adding any others. Agents carry two lists of tasks: the bundle b_i , with a path p_i . Tasks are added to the end of an agents bundle in the order of their assignment; while the path contains the order in which those tasks will be completed. The cardinality of b_i and p_i cannot be greater than the maximum assignment size, without any limitation of assignment size the cardinality is equal to the number of tasks N_m . Using $S_i^{P_i}$ as the total reward value for agent i performing the tasks along the path p_i , if a task j is added to the bundle b_i , it incurs the marginal score improvement of

$$c_{ij}(b_i) = \begin{cases} 0, & \text{if } j \in b_i \\ \max_{n \leq |p_i|} S_i^{P_i \theta_n \{j\}} - S_i^{P_i}, & \text{otherwise} \end{cases} \quad (2.3)$$

Where $|\cdot|$ denotes the cardinality of the list, and θ_n denotes the operation that inserts the second list right after the n^{th} element of the first list. A task is inserted into the current path at all possible locations to find the greatest score improvement. The first phase of the CBBA is summarized in algorithm 3 in Figure 2.5. Each agent carries four vectors: a winning bid list y_i , a winning agent list z_i , a bundle b_i and the corresponding path p_i . The difference between x_i and z_i is that in the CBBA an agent needs to know not only if it is outbid on the task it selects but who is assigned to each task as well; this enables better assignments based on more sophisticated conflict resolution rules.

Algorithm 3: CBBA Phase 1 for agent i at iteration t :

```

1: procedure BUILD BUNDLE( $z_i(t-1), y_i(t-1), b_i(t-1)$ )
2:    $y_i = y_i(t-1)$ 
3:    $z_i = z_i(t-1)$ 
4:    $b_i = b_i(t-1)$ 
5:    $p_i = p(t-1)$ 
6:   while  $|b_i| < L_t$  do
7:      $c_{ij} = \max_{n \leq |p_i|} S_i^{P_i \theta_n \{j\}} - S_i^{P_i}, \forall j \in J \setminus b_i$ 
8:      $h_{ij} = \Pi(c_{ij} > y_{ij}), \forall j \in J$ 
9:      $J_i = \operatorname{argmax}_j c_{ij} \cdot h_{ij}$ 
10:     $n_{i,J_i} = \operatorname{argmax}_n S_i^{P_i \theta_n \{J_i\}}$ 
11:     $b_i = b_i \theta_{\text{end}} \{J_i\}$ 
12:     $p_i = p_i \theta_{n_{i,J_i}} \{J_i\}$ 
13:     $y_{i,J_i}(t) = c_{i,J_i}$ 
14:     $z_{i,J_i}(t) = i$ 
15:  end while
16: end procedure

```

Figure 2.5: Task allocation algorithm for the CBBA (Choi et al., 2009).

Phase 2: Conflict Resolution

Similarly to the CBAA, the CBBA runs a consensus phase to remove agents bidding for the same task and unify group knowledge. In the case of the CBAA agents made bids for single tasks, if they received a higher winning bid for that task from one of their neighbours they would release it and re-assign themselves to another task. However the CBBA deals with multiple assignments where bids are made based on their current bundle of tasks. If an agent loses an assignment they must not only release the task in question but also any tasks scheduled to be completed after that task, where the marginal score values for the proceeding tasks are no longer valid. But this method of releasing tasks makes convergence more complicated as other agents might have made incorrect observations about the maximum bids. To prevent this, information about when agents last communicated with each other must also be transferred so in this way agents can work out if bid data for an agent is out of date.

In the consensus phase of the CBBA three vectors are communicated. The winning bid list y_i , the winning agent list z_i and s_i containing the last update time an

agent had with all other agents. When assignments are communicated the time vector s_i is calculated as

$$s_{ik} = \begin{cases} \tau_r, & \text{if } g_{ik} = 1 \\ \max_{m \in \{m: g_{im}\}} s_{mk}, & \text{otherwise} \end{cases} \quad (2.4)$$

where τ_r is the previous message reception time when an agent is in communication range, otherwise the agent uses the last update time of one of its neighbours who has had communication with the target.

With the addition of the time vector s_i an agent can determine when its bid data is out of date by comparing the update time for the agent it has received communication from. For example, if two agents i and k both think agent m is assigned to task j such that $z_{ij} = z_{ik} = m$ but that $y_{ij} \neq y_{ik}$ meaning that each agent has a different winning bid value for m , by comparing s_{im} with s_{km} agents can determine whose bid is out of date and update their information.

When agent i receives bid data from agent k about task j there are three possible actions agent i can take

- 1) *Update*: $y_{ij} = y_{kj}$, $z_{ij} = z_{kj}$
- 2) *Reset*: $y_{ij} = 0$, $z_{ij} = \emptyset$
- 3) *Leave*: $y_{ij} = y_{ij}$, $z_{ij} = z_{ij}$

Table 2.1 outlines the decision rules for all combinations of bid comparisons when receiving a communication message. The first column contains who the agent k , the sender, believes is assigned to task j . The second column contains who the receiver, agent i , thinks is assigned to the task. Finally depending on the combination of sender to receiver assignments, the final column displays the action the receiver will take depending on the agents involved, the bids placed and the update time of

each agent's data. As agents iterate between the two phases they will gradually converge on a conflict free solution, an overview of the CBBA converging on a conflict free assignment can be seen in Appendix A.1.

When the number of assignments an agent is allowed is limited to 1, the CBBA will produce the same result as the CBAA, so in this respect the CBBA can also guarantee 50% optimality for the single-assignment problem. Additionally because the multi-assignment problem can be treated as a single assignment problem with an additional combinatorial number of agents the minimum 50% performance guarantee can also be applied to the multi-assignment problem (Choi et al., 2009).

Table 2.1: Consensus decision table for the CBBA (Choi et al., 2009).

Sender's (agent k 's) z_{kj}	Receiver's (agent i 's) z_{ij}	Receiver's Action (default: leave)
k	i	<i>if $y_{kj} > y_{ij} \rightarrow \text{update}$</i>
	k	<i>update</i>
	$m \notin \{i, k\}$	<i>if $s_{km} > s_{im}$ or $y_{kj} > y_{ij} \rightarrow \text{update}$</i>
	<i>none</i>	<i>update</i>
i	i	<i>leave</i>
	k	<i>reset</i>
	$m \notin \{i, k\}$	<i>if $s_{km} > s_{im} \rightarrow \text{leave}$</i>
	<i>none</i>	<i>leave</i>
$m \notin \{i, k\}$	i	<i>if $s_{km} > s_{im}$ or $y_{kj} > y_{ij} \rightarrow \text{update}$</i>
	k	<i>if $s_{km} > s_{im} \rightarrow \text{update}$ else $\rightarrow \text{reset}$</i>
	m	<i>if $s_{km} > s_{im} \rightarrow \text{update}$</i>
	$n \notin \{i, k, m\}$	<i>if $s_{km} > s_{im}$ and $s_{kn} > s_{in} \rightarrow \text{update}$ if $s_{km} > s_{im}$ and $y_{kj} > y_{ij} \rightarrow \text{update}$ if $s_{kn} > s_{in}$ and $s_{im} > s_{km} \rightarrow \text{reset}$</i>
	<i>none</i>	<i>if $s_{km} > s_{im} \rightarrow \text{update}$</i>
<i>none</i>	i	<i>leave</i>
	k	<i>update</i>
	$m \notin \{i, k\}$	<i>if $s_{km} > s_{im} \rightarrow \text{update}$</i>
	<i>none</i>	<i>leave</i>

Scoring Functions

The CBBA provides a conflict free assignment on the assumption that the scoring function it uses satisfies a diminishing marginal gain (DMG). The marginal score improvement of a task described in (2.1) shows that the score improvement for agent i doing task j is $c_{ij}(b_i)$ but that this score is dependent on the current bundle b_i . If the scoring function satisfies DMG it can be said that the value of a task does not increase as other tasks are added to the set before it. This can be formally described as

$$c_{ij}(b_i) \geq c_{ij}(b_i \theta_{end} b), \forall b_i, b \in (J \cup \{\emptyset\})^{N_m} \quad (2.5)$$

where \emptyset denotes an empty task and θ_{end} is the function that adds the second list after the first list. The value of an assigned task c_{ij} with the bundle b_i does not gain any increased value when another task is added to the bundle $b_i \theta_{end} b$. Many reward functions in autonomous search and exploration robotics are consistent with DMG (Bertuccelli et al., 2009). The CBBA uses a time-discounted reward that satisfies DMG as follows

$$S_i^{p_i} = \sum \lambda^{\tau_i^j(p_i)} \bar{c}_j \quad (2.6)$$

where λ is the time discount rate, $\tau_i^j(p_i)$ is the estimated time of arrival for agent i travelling along with the path p_i to arrive at task location j and \bar{c}_j is the fixed reward for performing task j . This creates a time discounted reward where performing a task later will result in a reduced reward (Alighanbari, 2004). In search and rescue scenarios where uncertainty grows with time, the time discounted reward models the reduced expected reward for visiting a location later rather than earlier. Distance is not factored into the cost because travel time is sufficient at modelling the discounted reward and still satisfies the triangular inequality for distance between task locations such that

$$\tau_i^j(p_i \theta_n \{k\}) \geq \tau_i^j(p_i), \forall n, \forall k \quad (2.7)$$

results in an agent taking longer to travel between tasks thus arriving at each task later than if the agent travelled over a shorter path. This inequality further discounts the score value such that for all non-negative \bar{c}_j the scoring function $S_i^{p_i}$ does satisfy DMG. With this assumption, the CBBA provides a decentralised task-allocation algorithm addressing the multi-assignment problem to produce a conflict free solution.

2.2.5 Task Allocation with Duo Cooperation Restraints

An extension to the CBBA developed part of a solution to the multi-agent task allocation problem, extending the algorithm to deal with “duo tasks” that are defined as tasks requiring one or two agents. The algorithm was extended to solve missions with heterogeneous networked agents, where tasks are given a specific number of agents required for their completion. There are quantifiable advantages to using multiple UAVs for tasks that could be undertaken with a single UAV. For instance, search and rescue operations can be done with a single UAV but multiple UAVs would speed the process up among other advantages (Bernard et al., 2011) unlike computer parallelisation which rarely achieves twice the speed for twice the computing. In other situations a task might require multiple UAVs where a single UAV would not be sufficient, for example, two UAVs carrying heavy building material together (Willmann et al., 2012). This leads to three types of tasks defined as the following:

- 1) Solo Task (J_S): Referred to as single-agent tasks are tasks that require one agent to complete them. Additional agents assigned to the task are unacceptable providing no increase in the score of the assignment and would be classed as producing a conflict in the assignment.

- 2) Preferred Duo Tasks (J_{PD}): A task that can be completed by either one or two agents. The assignment of a single agent is acceptable and provides the same reward as a solo task. Assigning two agents to the task provides a greater reward.
- 3) Required Duo Tasks (J_{RD}): Similar to the Preferred Duo Tasks however a single assignment is unacceptable and provides no reward. Instead these tasks require two agents of differing types to complete the task.

Algorithm 4: Decentralized task elimination for agent i

```

1: Initialise invalid tasks set:  $J_{IV} = \emptyset$ .
2: Initialise outer-loop iteration count:  $t_{out} = 0$ .
3: while  $J_{IV} = \emptyset$  OR  $t_{out} < 1$  do
4:   Eliminate invalid tasks:  $c_{ij1} = 0, c_{ij2} = 0, \forall j \in J_{IV}$ .
5:    $t_{out} = t_{out} + 1$ .
6:   while CBBA not converged do
7:     CBBA bundle construction phase.
8:     CBBA conflict resolution phase.
9:     Check CBBA convergence.
10:  end while
11:  Identify invalid tasks:  $J_{IV} \equiv \{j \in J_{RD} | (z_{ij1} = NULL) XOR (z_{ij2} = NULL)\}$ 
12: end while

```

Figure 2.6: Decentralised task elimination for agent i for the assignment and consensus of cooperative duo tasks (Choi et al., 2010).

The assignment algorithm for cooperative duo tasks is displayed in Figure 2.6 and uses a method of decentralised task elimination to remove and re-distribute agents amongst the remaining valid tasks. The inside loop of the algorithm (lines 6-10) runs the regular CBBA with additional task restriction such as an agent cannot assign itself to both the leader and follow role for a duo task. Once the inner CBBA has converged all agents have the same situational awareness about the status of each task because of the conflict free properties of the CBBA. The process of task elimination is used to remove any tasks that either have no assignments or an incorrect number of assignments for the required duo tasks. With the invalid tasks removed the CBBA is

run again with the restricted pool of tasks, this allows agents with spare time to assign to the second slot of preferred duo tasks providing an improved solution.

The algorithm successfully allows the assignment of duo tasks and provides an improvement in score over the CBBA when using preferred duo tasks. However the algorithm is limited to two agent requirements and the use of vectors for the winning bid and agent lists causes problems where agents must be explicitly forbidden from assigning themselves to each and every other part of a duo task they are assigned to. This system would provide difficulties expanding the algorithm for agent requirements greater than two.

2.2.6 Task Allocation via Coalition Formation

A method of multi-agent cooperation involves the assignment of tasks to groups of cooperating agents called coalitions (Shehory & Kraus, 1998) (Lau & Zhang, 2003) (Amgoud, 2005). This situation involves tasks that can be split up into many sub-tasks that may not be satisfied by a single agent. This problem takes a set of agents $I \equiv \{1, \dots, N_n\}$ and a set of tasks $J \equiv \{1, \dots, N_m\}$ where agents must work out how best to form coalitions so as to maximise the overall reward from completing tasks. Each task j contains a fixed number k of subtasks l with each subtask requiring a specific capability value a_{jl} and providing a reward of p_j . Every agent i has an associated vector $C_i = [c_{j1}, c_{j2}, \dots, c_{jk}]$, where c_{jl} is the capability value of the agent performing a sub task l . Coalitions are defined as groups of agents working towards a common goal each coalition is given a value based on the sum of the capabilities of the group. Agents in coalitions can then work together to complete various tasks and subtasks.

Task allocation via coalition formation follows three general phases:

- 1) Generate the collation structures; here the agents form a collation in order to coordinate at completing a task or set of tasks.
- 2) Discuss the structure amongst the agents to determine which one is most suitable.
- 3) Distributing the sub-tasks over the agents of the coalition.

The structure of coalitions depends on the specific problem; it might be that tasks are independent or that agents must belong to only one or multiple coalitions. It is assumed that agents are co-operative and interested in maximising the overall score of the system, therefore the objective function is the sum of all fulfilled tasks as seen in (2.8) where x_j is the non-negative integer for high-demand meaning a task can be repeated as many times as resources will allow or a boolean value where the task can only be completed once. C^l is the total resources available for the subtask l

$$\begin{aligned}
 &\text{Maximize} && \sum_{j=1}^{N_m} p_j x_j \\
 &\text{Subject to} && \sum_{j=1}^{N_m} a_{jl} x_j \leq C^l, l = 1, 2 \dots k
 \end{aligned} \tag{2.8}$$

Various solutions to this problem exist depending on the specific task settings, consider that each task can only be completed once, agents have limited resources and the reward is fixed. A standard greedy approach does not provide a good solution as the capabilities of each agent are limited. Instead coalitions are constructed iteratively by maximising the coalition value (Lau & Zhang, 2003). This approach has two steps that are iterated between, firstly the coalition capability values are computed for each possible coalition, then secondly the coalition with the highest value among all tasks is formed and the task that is assigned to the coalition removed from the task list. This process is alternated until no more tasks are left or a coalition with sufficient value cannot be formed.

Task allocation via coalition formation allows agents to pool resources together and complete tasks with their related sub-tasks that a single agent would be unable to do. However, coalition formation assumes global communication and whilst it can function with some delay the quality of assignments is reduced as delay increases. In addition the population of agents cannot change during the formation process and would not function with unknown agents that are discovered as the assignment proceeds. Finally many solutions for achieving coalition formation assume global knowledge of the goals (Shehory & Kraus, 1998). Whilst each task can be defined as a multi-agent task, it is always broken down into single-agent sub tasks such that coalitions are formed to complete the multi-agent task but agents are still only being assigned to single-agent tasks.

2.3 EUSOCIAL ANIMAL BEHAVIOURS

Eusocial animals, like the majority of ant species, a number of bee species and a few wasp species have some similarities to that of robotic cooperative systems. Unlike most animals, eusocial species focus on the group rather than the individual. An ant, for instance, has evolved to put the success of the colony ahead of itself, in the case where for example ants have been shown to use self-sacrificial defences to protect the nest (Tofilski et al., 2008). Similarly with a cooperative system an individual agent should focus on maximising the performance of the group as a whole rather than its own performance. Ant nests allocate specific workers to specific tasks without any central or hierarchical control (Anderson & Ratnieks, 1999). Whilst the task allocation is individual centric and the decision is made by an individual it must still be beneficial to the group. Some decisions will reduce an agent's contribution but overall increase the team's performance, the allocation algorithm must account for both loss of time and score by not fully allocating multi-agent tasks. Detrain and

Deneubourg (Detrain & Deneubourg, 2006) show how if-then rules embedded in ant behaviours, however simple in their logic, ultimately produce efficient group-level responses for objectives like resource acquisition and risk avoidance. Further, that these behavioural rules coupled with self-organising processes provide a robust and efficient method for problem solving. A difficulty encountered with multi-agent tasks is that agents can become confined to tasks that no other agents plan to assist with. When a multi-agent task has insufficient assignments the task cannot be completed and will not score, thus wasting the contribution of agents assigned.

Bees are another eusocial species that show collective intelligence in the organisation and allocation of tasks for the survival of the colony. Bees perform task partitioning where a task is split up into a number of steps that are performed by multiple bees (Arathi & Spivak, 2001) where each bee has a specific part of the overall task to achieve. This focuses the hive on the task and its division rather than the individual performing the task. As part of a “hygienic behaviour” worker bees remove diseased brood cells from the hive, if left the infection would spread and destroy the colony. This requires two operations, the removal of the cap on the cell followed by removing the diseased brood. Individual bees will focus either on removing the cap or removing the cell and together will complete the tasks rather than each individual removing a cap then removing the diseased cell. Bee colonies show complex cooperative behaviours for the organisation and allocation of workers in the hive. Multiple systems have been proposed that show how bee colonies come to collective decisions in tasks such as the allocation of workers to nectar sources with changing environmental conditions (Seeley et al., 1991) (Biesmeijer & de Vries, 2001) (Cox & Myerscough, 2003). The similarities between the multiple systems are that global coordination of the workers happens despite individual bees relying on local information. The self-organising model of these colonies shows the amplification of random noise into structured patterns and that collective problem

solving capabilities can emerge when individuals have limited information processing abilities (Deneubourg et al., 1991). With a great deal of research focused on the foraging abilities of honey bees, Johnson examined the self-organisation of the internal hive where bees would perform over 15 tasks varying in exigency, often localised to specific regions of the hive (Johnson, 2009). Johnson proposed an agent based self-organisation model that explained the fluid task-allocation dynamics observed in the hive. Using a form of task-quitting the bees are able to track changes in task demand at the group level whilst individually using local information. As bees become insensitive to certain stimuli for a period of time after quitting a task it allowed bees to redistribute the colony resources to high demand areas. Figure 2.7 shows the developed task-quitting algorithm, where agents can be at one of three behavioural states; working, patrolling or inactive. Whilst in a specific behavioural state the bees will perform tasks related to it, where a working bee is either busy or not. At each time step, bees either quit or remain in their current behaviour state based on a quitting probability. The quitting probability was developed such that the agent bees would stay in a state on average as long as was empirically observed in the bee hives. Johnson was able to show that frequent task quitting can allow colonies to track variation in task demand in a changing environment. This process allows bees to re-assign themselves to high priority areas and would be useful in solving the problem of agents being assigned to multi-agent tasks that are not reaching the correct requirements.

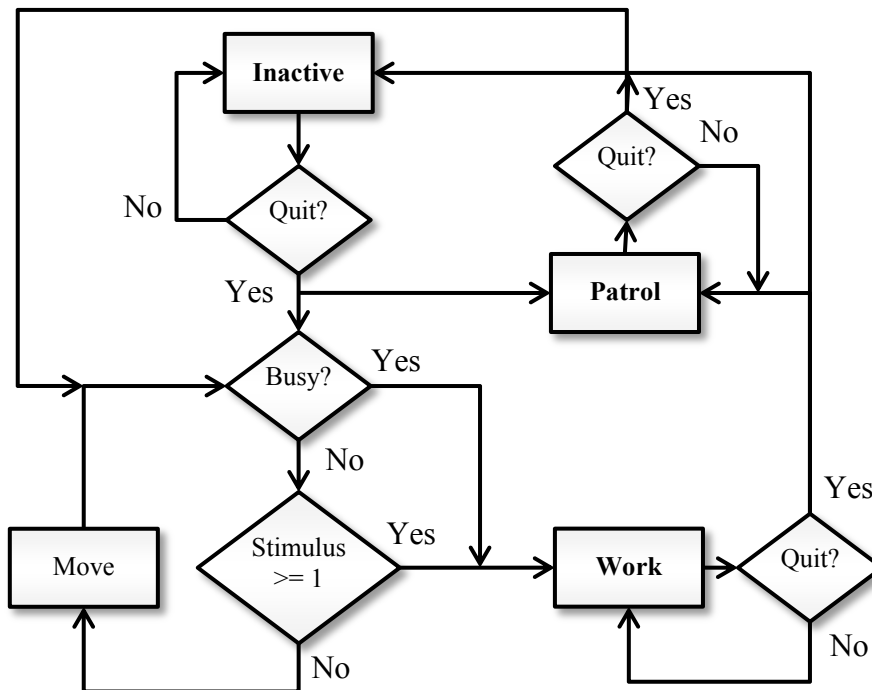


Figure 2.7: Task allocation algorithm using a task quitting method for the allocation of bees throughout a hive (Johnson, 2009). At each time step, bees either remain in their behavioural state or quit based on a quitting probability. When a bee quits its current activity it chooses randomly one of the other two states (Work, Patrol, and Inactive). Working bees are either busy or not busy, busy bees complete tasks in their location, when there are no tasks at a bees location the bee is not busy and instead moves randomly to find work.

An environmental change in a bee hive such as an increase in the temperature creates an increase in new assignments to the task of “fanning” that cools down the hive and larva. Johnson (Johnson, 2009) showed that using this method of frequent task quitting resulted in a similar change in assignments between the real colonies and the simulated ones. Additionally Johnson showed that by decreasing the probability of bees quitting their current task the allocation of bees to the tasks with the highest demand was greater but that the hives response to changing task demands, such as environmental changes, was much lower. Essentially the bee’s use of task quitting causes a sacrifice in work output but it increases the hives ability to deal with and adapt to a changing environment.

2.4 CONCLUSIONS

This Chapter provided a detailed description of current solutions for the single-agent task assignment problem. Furthermore, the limitations of these solutions for multi-agent task assignments were discussed. Whilst the CBBA provides a conflict free solution with 50% optimality for the single-agent task assignment problem, it has limited capabilities in solving the multi-agent assignments as well as difficulties scaling the solution for increasing agent requirements. Behaviours observed in eusocial species show promise at providing stronger assignments in task allocation algorithms. Using the CBBA a dynamic and scalable solution to the multi-agent task assignment problem can be provided that removes the limitations of previous solutions. To achieve these goals new assignment and consensus algorithms will need to be developed for multi-agent tasks where single agent tasks can continue to be handled by the CBBA. Furthermore to allow the new algorithm to function with dynamically added tasks or agents, as well as handle heterogeneous agents, will require a modification of how the CBBA structures its assignment data. These changes to the CBBA will increase the complexity of tasks that agents can deal with as well as provide potential practical applications in the future.

Chapter 3: *Consensus Based Grouping Algorithm*

This chapter addresses the problem of multi-agent task assignments for UAVs, which are defined as tasks that require multiple agents. The algorithm is an extension of the Consensus Based Bundle Algorithm that converges to a conflict free, feasible solution which previous algorithms were unable to account for. Furthermore the algorithm takes into account heterogeneous agents and task dependencies such that groups of UAVs with differing equipment or sensors can self-organise in order to complete series of complex tasks.

The CBBA (Choi et al., 2009) was created to solve an extension of the TAP where agents queue up tasks they will complete: individual agents take available tasks and compute every permutation given their current queue of tasks or “bundle”, where the highest rewarded permutation becomes their bid for that task. In this way, agents continually remove and revise new tasks as other agents find they can create a more valuable sequence with that task. Thus, the CBBA gives a conflict free solution with a guaranteed 50% optimality to the multi-assignment problem.

Extensions of the problem can be developed that simulate realistic situations by designing complex tasks with stricter requirements. The consensus algorithm needs to be developed in order to handle these new tasks, including tighter task selection and higher cooperative decision making. The requirements that are examined are multi-agent, equipment requirements and task dependencies, where a multi-agent task is defined as one which requires more than one agent to complete, an example of which would be using two UAVs to carry construction material (Willmann et al., 2012). A task that requires specific equipment would require unique agents; Merino et al (Merino et al., 2005) looked at using multiple heterogeneous agents for cooperative fire detection. Task dependencies are defined as tasks that

require other tasks to be completed before they can start, creating a list of tasks that must be completed in a certain order.

Two solutions for the multi-agent task allocation problem (Choi et al., 2010 ; Manisterski et al., 2006) both have their limitations that make them unsuitable. Firstly, the creators of the CBBA extended their algorithm for heterogeneous cooperation (Choi et al., 2010); this extension solved duo cooperation constraints where a simulation would contain two agent types that solve three different types of tasks. Solo tasks required one type of agent; preferred duo tasks scored greater for the assignment of two different agents and required duo tasks needed one of each agent type. However, this solution is limited to two agents and the proposed solution here will allow any number of agent requirements to be assigned to tasks. Secondly, another solution to the multi-agent problem (Manisterski et al., 2006) used a central solver to group related tasks into a set and assign enough agents to complete them. However, using a centralized algorithm will not provide a robust and feasible solution for real world applications. This chapter provides a solution for the decentralised assignment of multi-agent tasks that require any number of agents for their completion. Solving this problem can increase the cooperation of UAVs to an improved autonomous operational level further reducing the need for human interaction. To achieve this, agents need to develop an increased awareness of what other agents are planning more so than is required for the CBBA. Agents must plan their own schedules around that of others and come to complex agreements on task order. As the complexity of decision making increases so too does the requirement for information needed to make a decision and the underlying communication required (Nodine et al., 2001). Using the framework set up by the CBBA the algorithm is extended to account for the existing limitations; this extension leads us to the Consensus Based Grouping Algorithm (CBGA).

3.1 PROBLEM

The CBBA is limited to single-agent tasks and is unable to handle further restrictions on which agents can complete those tasks; the duo-task extension is similarly limited to multi-agent tasks for two agents. This chapter develops an algorithm that can deal with and provide a conflict free solution to the following restraints.

- Tasks require 1 to n agents
- Tasks require specific equipment or sensors
- Tasks can have an order of completion
- Tasks have a time window in which they must be started

Agents will need to form groups containing the correct equipment before being able to complete a task. Additionally tasks can require a specific order of completion. The CBGA will provide a conflict free solution to this problem with some small assumptions on the network connectivity and scoring scheme.

3.1.1 Single-Agent Task Assignment Problem

The single-agent task assignment problem is a combinatorial optimization problem that tries to find the least-cost solution between two disjoint sets. There is a set of agents $I \equiv \{1, \dots, N_n\}$ and a set of tasks $J \equiv \{1, \dots, N_m\}$. The objective of the task assignment problem is to find a conflict free matching set of agents to tasks that maximises a global reward. With a valid assignment each agent $i \in I$ can be assigned up to a maximum of M_m tasks and each task $j \in J$ must have no more than one agent assigned.

An agent has a reward associated with it for completing a task. Let c_{ij} be the non-negative reward of assigning the i^{th} agent to the j^{th} task. The objective is to assign each task to one agent in such a way as to maximise the overall reward from

completing all the tasks. There is a binary variable x_{ij} where $x_{ij} = 1$ to indicate agent i is assigned to task j , otherwise $x_{ij} = 0$. The global reward or assignment score is the sum of all local rewards, where each local reward is the function of tasks assigned to an agent. Then the local reward generated by agent i is equal to (3.1) where $c_{ij}(p_i)$ is the path dependant reward of agent i completing task j on the path p_i .

$$\sum_{j=1}^{N_m} c_{ij}(p_i)x_{ij} \quad (3.1)$$

For an assignment to be efficient the task allocation must be valid and the reward is maximised as (3.2).

$$\begin{aligned} & \max \sum_{i=1}^{N_n} \left(\sum_{j=1}^{N_m} c_{ij}(p_i)x_{ij} \right) \\ & \text{subject to} \sum_{j=1}^{N_m} x_{ij} \leq M_m, \forall i \in I \\ & \sum_{i=1}^{N_n} x_{ij} \leq 1, \forall j \in J \\ & x_{ij} \in \{1,0\}, \forall (i,j) \in I \times J \end{aligned} \quad (3.2)$$

The single assignment problem emerges when $M_m = 1$ which can be solved with the CBAA. If $M_m > 1$ then multi-assignments are allowed creating the multi-assignment problem with a solution provided by the CBBA, although the CBBA also provides the same solution as the CBAA for the single assignment problem.

3.1.2 Multi-Agent Task Assignment Problem

The task assignment problem is extended to cover the addition of multi-agent tasks. Each agent i can be assigned to multiple tasks as part of the CBBA, conversely each task j can similarly have multiple agents assigned to it. Agents will now need to store

a matrix of bid data that will allow agents to track multiple winners for a multi-agent task where the task requires multiple agents. The winning agent matrix will now take the form $x_{ikj} = 1$, which translates to agent i thinks agent k is assigned to task j with a winning bid value of y_{ikj} .

Tasks now contain an agent requirement L_j that specifies how many agents are required for the task j . With multiple agents potentially being assigned to a task the algorithm will not limit each task j to a single assignment. Instead for a task assignment to be valid (3.3) must be true for a given task j , where x_{ij} determines an agent's own knowledge about its assignments and avoids double counting assignments where $x_{imj} = x_{kmj}$ with a conflict free solution.

$$\sum_{i=1}^{N_n} x_{ij} = L_j \quad (3.3)$$

It is assumed that $M_m = N_m$ such that agents will have no limit on the number of tasks they can assign themselves to. Combining the task assignment problem from (3.2) with the restriction for multi-agent tasks in (3.3) for an assignment to be efficient the task allocation must be valid and the reward is maximised as

$$\begin{aligned} \max \quad & \sum_{i=1}^{N_n} \left(\sum_{j=1}^{N_m} c_{ij}(p_i) x_{ij} \cdot z_j \right) \\ \text{subject to} \quad & \sum_{j=1}^{N_m} x_{ij} \leq N_m, \forall i \in I \\ & \sum_{i=1}^{N_n} x_{ij} \leq L_j, \forall j \in J \\ & L_j > 0, \forall j \in J \\ & x_{ikj} \in \{1,0\}, \forall (i, k, j) \in I \times I \times J \end{aligned} \quad (3.4)$$

where z_j is the binary value for task validity equal to 1 when a task is valid or 0 when a task is invalid depending on the task requirements. A multi-agent task can be

determined valid using (3.3) by having the correct number of assignments. If the task has less than the required number the requirements have not been met so the task is considered as having failed providing no score. A failed task is not considered a conflict and no cost is associated with failing other than the indirect cost of not receiving a reward for the assignments.

3.1.3 Restricted Task Assignments

In addition to the constraints in (3.4) further requirements and restrictions are added to tasks that must be satisfied, each task will require specific types of agents or a specific set of equipment before the assignment can be considered valid. There is a list of N_q pieces of equipment $E \equiv \{e_1, e_2, \dots, e_{N_q}\}$ found in the assignments where ae_i is a list of equipment that agent i has such that $ae_i \subseteq E$. Similarly task j requires a specific list of equipment te_j where $te_j \subseteq E$, if $te_j = \emptyset$ then it is assumed any agent can bid on the task. When $ae_i \cap te_j \neq \emptyset$ agent i can bid on task j because it contains at least one piece of equipment required. A valid assignment is worked out using

$$te_j \setminus \{ae_i | x_{ij} = 1\} = \emptyset, \forall i \quad (3.5)$$

where “\” is the set complement that returns the list of equipment in te_j that is not found in the current assignment $\{ae_i | x_{ij} = 1\}$. When (3.5) is true task j has a valid assignment with the correct equipment. If the equipment list ae_i for each agent is limited to a single piece of equipment such that $|ae_i| = 1, \forall i$ then the problem is limited to agents of different types and that $|te_j| = L_j$. In this case each task requires multiple agents of a specific type.

When making assignments with task planning from the CBBA tasks are only available for bidding when all requirements have been met, agents should be able to plan all tasks in advance. When tasks have time restrictions for their completion it

becomes imperative to assign as many tasks as possible at the beginning of the simulation, waiting until after the pre-requisites of a task have been completed could lead to no time being available or agent close enough to complete it. However if agents assume an assigned task will be completed they can prepare ahead of time to complete the follow up task and it does not necessarily have to be the same agent performing the following task.

Therefore a set of task prerequisites P_j are created for each task j such that $P_j \subset J$ and that $j \notin P_j$. The set P_j contains which tasks must be completed before the related task j can be attempted. When $P_j = \emptyset$ task j has no prerequisites and availability is limited to the highest bidder as before. Assuming that the existence of a bid for a preceding task will result in the completion of that task then a task with prerequisites is valid when

$$(x_{im} > 0), \forall i \in I, \forall m \in P_j \quad (3.6)$$

thus when (3.6) is true agent i can bid on task j where all prerequisite tasks m have valid assignments. With multiple prerequisite tasks the order of completion does not matter except in the case that those tasks also have their own prerequisite. However, these pre-conditions are assumed to be unchanging such that by completing another task or fulfilling a prerequisite task does not add, remove or change other pre-conditions. In reality the task of attacking a target might require a task prerequisite to find the target in an area, if the target is unfound then the following attack task is now unnecessary. Tasks and their prerequisite tasks are assumed to be static such that the conditions set out at the beginning of a simulation for each task are not changed during the simulation.

Finally each task has a start and end time in which the task must begin, an agent can calculate valid tasks for assignment using

$$\tau_i^j(p_i) < end_j \quad (3.7)$$

where $\tau_i^j(p_i)$ is the estimated arrival time of agent i at task j along the path p_i . If the agent can arrive before the end time end_j then it can be assigned to the task. Additionally it cannot finish a task until after the start time with each task taking a length of time to complete.

3.2 THE PROPOSED CBGA

The Consensus Based Grouping Algorithm is a solution to the multi-agent task assignment problem where tasks can require multiple agents before they can be completed. Agents make bids on valid tasks and send this data to their neighbours. All agents receive bids from their neighbours and validate that data with their own removing conflicts and converging on a single global solution as assignment data propagates through the networked agents. The CBGA is split similarly to the CBBA into two phases; first the bundle construction phase where agents fill their bundle with tasks they will complete, secondly the consensus phase where agents come to an agreement on which agents are participating in each task.

3.2.1 Local Data

Table 3.1 displays the data each agent stores during a simulation to perform assignment and consensus on tasks. The data that agent i sends to agent k when they are within communication distance $g_{ik}(\tau) = 1$ are displayed in Table 3.2. An agent i sends all the winning bids that it knows about where $y_{ikj}, \forall k, \forall j$ contains every combination of agent to task bids that agent i has. Similarly the winning agent matrix x_i is communicated which can be seen in Table 3.3. Along with this agent update times are sent allowing agents to know how old information is that they are being sent. Because all winning bid data is sent to an agent's neighbour winning bids will eventually

propagate to every agent connected in the communication network even though two agents might be outside each other's direct communication range. The equipment each agent has is assumed to be known by other agents, in a team of cooperating UAVs this information would already be known or in the case of discovering new agents the information would be communicated on first contact. Similarly the equipment requirement on tasks is assumed to be known.

Table 3.1: Data stored on each agent i .

Stored Data	Symbol	Description
Bundle	b_i	List of tasks the agent has currently assigned to itself. Ordered based on when tasks were added to the agent's assignments.
Path	p_i	Similar to the bundle a list of tasks the agent has currently assigned itself. Ordered by the order in which an agent will complete the tasks.
Winning Bid Matrix	y_i	Matrix containing the winning bid each agent has made to each task according to agent i .
Winning Agent Matrix	x_i	Matrix containing the winning assignments where 1 means an agent is assigned to a task otherwise 0.
Agent Update Times	s_i	List of last update time from each agent in time t .
Equipment List	ae_i	List of equipment an agent has or the agents type when limited to one instance.

Table 3.2: Data communicated by each agent i .

Communicated Data	Symbol	Description
Winning Agents	x_i	The agents that agent i thinks are assigned to each task.
Winning Bids	y_i	The winning bids matrix that agent i thinks have been made for each task
Agent Update Times	s_i	List of last update time from each agent in time t .

Table 3.3: Winning agent matrix x_i for agent i storing the binary values for winning assignments between agents and tasks. The winning bid list y_i is stored in exactly the same except that it contains the winning bid.

		Winning Agent Matrix (x_i)				
		Tasks				
		1	2	3	...	j
Agents	1	x_{i11}	x_{i12}	x_{i13}	...	x_{i1j}
	2	x_{i21}	x_{i22}	x_{i23}	...	x_{i2j}

	i	x_{ii1}	x_{ii2}	x_{ii3}	...	x_{iij}

3.2.2 Phase 1: Bundle Construction

In phase 1 each agent constructs a bundle of tasks $b_i \in (J \cup \{\emptyset\})^{N_m}$ and the ordered path for those tasks $p_i \in (J \cup \{\emptyset\})^{N_m}$. Bundle and path construction works similarly to the development in the CBBA (Choi et al., 2009) but with the new task restrictions limiting the valid tasks for bidding. During the bundle phase an agent builds up a bundle of tasks it plans to complete by calculating the marginal improvement of each task and selecting the task with the greatest improvement. After adding the best task to its bundle it repeats the process for the rest of the tasks, continuing until no more tasks are valid or all tasks have been added.

An agent determines which task it will add next to the bundle by calculating the marginal score improvement of a task. Each task provides a fixed reward \bar{c}_j for each agent, multi-agent tasks provide the same reward for every agent assigned therefore creating higher rewards for such tasks. As with the CBBA an agent places a bid on a task based on the marginal score improvement it can achieve given the agent's current bundle. Because tasks are given time windows for their completion a time discounted reward for the entire simulation is not viable, a task should not supply a reduced reward because it starts later than another. Instead the time discounted

reward should apply from the start time of the task not the simulation time. Thus the reward for an agent i completing task j is worked out as

$$reward_j = \bar{c}_j e^{-\lambda(\tau_i^j(p_i) - start_j)} \quad (3.8)$$

where $0 \leq \lambda \leq 1$ is the time discounted reward and $\tau_i^j(p_i) - start_j$ calculates the time difference between agent i 's arrival at task j given path p_i and the start time $start_j$ of the task. Thus as an agent receives the maximum reward by arriving on time, arriving later provides a reduced reward. However, the distance an agent travels to a task will provide an increasing cost such as the fuel requirement to travel or the additional risk encounter over long journeys. The distance discounted reward is calculated as

$$cost_j = d_i^j(p_i) \gamma \quad (3.9)$$

where $d_i^j(p_i)$ is the non-negative estimated distance agent i will travel to task j on the current path p_i and γ is the cost associated with traveling. This provides a further discounted reward for completing a specific task.

$$S_i^{p_i} = \sum (reward_j - cost_j) \quad (3.10)$$

The overall score for agent i completing an assignment is calculated in (3.10), with the use of both time and distance discounted rewards makes the scoring function satisfy DMG because of triangular inequality. The time and distance discounted reward can model the degradation of an expected reward for completing a task that is further away or for arriving at the task late.

As with the CBBA when a task j is added to the bundle b_i the marginal score improvement c_{ij} is calculated as

$$c_{ij}(b_i) = \begin{cases} 0, & \text{if } j \in b_i \\ \max_{n \leq |p_i|} S_i^{p_i \theta_n \{j\}} - S_i^{p_i}, & \text{otherwise} \end{cases} \quad (3.11)$$

where θ_n is the operation that slots the second list into position n of the first list and therefore $S_i^{p_i \theta_n \{j\}}$ is the score for slotting task j into position n in the path p_i . The position of task j in path p_i that provides the highest score improvement is used as an agent's bid for the task j . Agents are only able to bid on valid tasks such that if an agent or task does not meet the task requirements in (3.5) and (3.6) then their bid for that task will not be considered.

Before a bid selection the list of score improvements c_{ij} must be compared to the current highest placed bids to create the valid bid list h_{ij} . In the CBBA this process simply compared an agent's bid to the current winning bid found in y_{ij} , the valid bid list was generated using

$$h_{ij} = \text{II}(c_{ij} > y_{ij}), \forall j \in J \quad (3.12)$$

where $\text{II}(\cdot)$ is one if the argument is true otherwise it is equal to zero. However, the multi-agent requirement on tasks will require a change in this process. With single-agent tasks there was only one bid for comparison and an agent either provided a high enough bid or did not and thus replaced the previous bid. Multi-agent tasks allow multiple assignments so long as the number of assignments does not exceed the number of agents required for the task L_j . In addition multi-agent tasks do not require that an agent beat all the assigned bids, by beating and replacing the smallest bid the overall reward from a task will increase.

When $L_j = 1$ a task is considered a single-agent task and as such bid comparison will not change. If $L_j > 1$ then the task is a multi-agent task with two situations, either the number of assigned agents is less than the required number or the task is full in which case the agent must out bid another agent for the task. The

number of agents that agent i thinks are assigned to a task j can be calculated as the summation of all assigned agents in an agent's winning bid matrix, where a task is considered full when the number of assigned agents is equal to L_j as shown in (3.13).

$$\sum_{k=1}^{N_n} x_{ikj} = L_j \quad (3.13)$$

If (3.13) is satisfied then a task is considered full and therefore an agent must provide a bid higher than some other agent assigned to the task. Finding and replacing the minimum bid will gradually provide a higher scoring assignment. An agent can assign a bid to a task when (3.14) is true and in the case of tied scores agent ID is used to determine the winner.

$$c_{ij} \geq \min(y_{ikj}), \forall k \in I \quad (3.14)$$

When either (3.13) or (3.14) are true then an agent has a valid bid for the multi-agent task j and the valid bid list is updated with $h_{ij} = 1$. Once an agent has generated the marginal score improvement for each valid task it must then select the best task to be added to the bundle. The highest bid in c_{ij} that complies with the valid bid list h_{ij} is placed in agent i 's bundle and added to location n in the path p_i . The agent updates its own bid lists y_{ij} and x_{ij} before re-calculating all score improvements and adding another task to the bundle. This process is repeated until an agent can no longer add any more tasks to its bundle because either the bundle is full or there are no more valid tasks to bid on. Figure 3.1 shows a summary of the bundle construction phase for an agent i .

Algorithm 5: CBGA Bundle Construction for agent i at iteration t :

```
1: procedure BUILD BUNDLE( $x_i(t-1), y_i(t-1), b_i(t-1), p_i(t-1)$ )
2:    $y_i = y_i(t-1)$ 
3:    $x_i = x_i(t-1)$ 
4:    $b_i = b_i(t-1)$ 
5:    $p_i = p(t-1)$ 
6:   while  $|b_i| < M_m$  do
7:      $c_{ij} = \max_{n \leq |p_i|} S_i^{p_i \theta_n \{j\}} - S_i^{p_i}, \forall j \in J$ 
8:     for  $\forall j \in J$ 
9:       if  $L_j > 1$  then
10:        if  $L_j > (\sum_{k=1}^{N_n} x_{ikj} > 0)$ 
11:           $h_{ij} = 1$ 
12:        else if  $c_{ij} > \min(y_{ikj}), \forall k \in I$  then
13:           $h_{ij} = 1$ 
14:        else
15:           $h_{ij} = 0$ 
16:        end
17:       else
18:          $h_{ij} = \Pi(c_{ij} > \sum_{k=1}^{N_n} y_{ikj})$ 
19:       end
20:     end
21:      $J_i = \operatorname{argmax}_j c_{ij} \cdot h_{ij}$ 
22:      $n_{i,J_i} = \operatorname{argmax}_n S_i^{p_i \theta_n \{J_i\}}$ 
23:      $b_i = b_i \theta_{\text{end}} \{J_i\}$ 
24:      $p_i = p_i \theta_{n_{i,J_i}} \{J_i\}$ 
25:      $y_{ii_{J_i}}(t) = c_{i,J_i}$ 
26:      $x_{ii_{J_i}}(t) = 1$ 
27:   end while
28: end procedure
```

Figure 3.1: Bundle phase for CBGA.

3.2.3 Phase 2: Consensus

Phase 2 of the algorithm takes communications received from neighbouring agents and analyses their knowledge of assignments to come to a consensus. In the CBBA each agent would send their winning agent and winning bid list, with the CBGA because of the existence of multi-agent tasks agents are now sending a matrix of assignments instead of a single list. Each agent communicates their winning agent matrix x_i , winning bid matrix y_i and the time stamp s_i displaying the last information

update from each neighbouring agent. The data sent and received under a limited communication network can be seen in Figure 3.2.

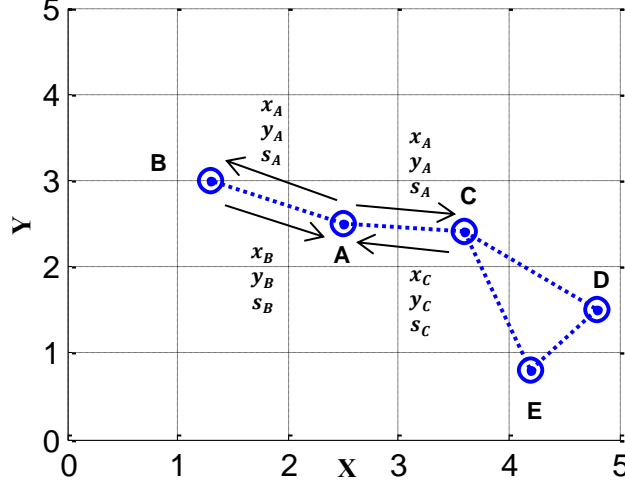


Figure 3.2: Data sent and received between agent A and its neighbours, a link between two agents represents that those agents are within communication distance and thus can communicate data between each other.

As agents receive assignment data from neighbours they will build up and store assignment matrixes for each neighbour where $x_{ikj} = 1$ shows that agent i thinks that agent k is assigned to task j with a corresponding bid y_{ikj} from the winning bid matrix. The consensus algorithm can be seen in Figure 3.3 and is split into two sections, the first section (line 4-6, Figure 3.3) deals with tasks that require a single agent, using L_j to determine the number of agents required for task j . Tasks requiring a single agent will require the same consensus algorithm as found in the CBBA (Choi et al., 2009) including the same decision table found in Table 2.1. The consensus algorithm assumes only valid bids are made during the bundle construction algorithm, thus no changes are required for single-agent tasks. The second section (line 7-25, Figure 3.3) contains the multi-agent consensus part of the CBGA where $L_j > 0$, which is split into two phases; the first correlates the receiver's current information with that of the sender. Secondly the receiver takes new information from

the sender and merges it with its own data to produce a consistent set of agreed information. The CBBA used a table for determining whether to update, leave or reset information; with the extended problem this becomes problematic. When another agent has differing assignments it does not necessarily require leaving or updating the information as done in the CBBA, the information could merge causing both agents to be correct. Further complications come when equipment requirements are taken into account. The algorithm is split into two phases to best handle the incoming information, by correcting each agent's information the agent can merge incoming data better by not having to account for mistakes in its own data.

Algorithm 6: Conflict Resolution for Agent i

```

1: send  $y_i, x_i$  and  $s_i$  to agent  $k$  with  $g_{ik}(\tau) = 1$ 
2: receive  $y_k, x_k$  and  $s_k$  from agent  $k$  with  $g_{ki}(\tau) = 1$ 
3: for  $\forall j \in J$ 
4:   if  $L_j = 1$  then
5:     Consensus from CBBA
6:   else
7:     for  $\forall m \in I$  where  $x_{imj} = 1$ 
8:       if  $m = k$  or  $s_{km} > s_{im}$  then
9:          $x_{imj} = x_{kmj}$ 
10:         $y_{imj} = y_{kmj}$ 
11:       end
12:     end
13:     for  $\forall m \in I$  where  $x_{kmj} = 1$ 
14:       if  $m \neq i$  and  $x_{imj} = 0$  and  $s_{km} > s_{im}$  then
15:         if  $(\sum_m x_{imj} > 0) < L_j$  then
16:            $x_{imj} = x_{kmj}$ 
17:            $y_{imj} = y_{kmj}$ 
18:         else if  $\min(y_{inj} \forall n) < y_{kmj}$  then
19:            $x_{inj} = 0, x_{imj} = x_{kmj}$ 
20:            $y_{inj} = 0, y_{imj} = y_{kmj}$ 
21:         end
22:       end
23:     end
24:   end  $s_{ik} = \tau$ 
25: end

```

Figure 3.3: Conflict resolution for the CBGA for multi-agent task.

The first phase (7-11, Figure 3.3) takes all the winning assignments the receiver i has for task j by checking for the existence of $x_{imj} = 1, \forall m \in I$ and compares how correct that information is with the sender. If the agent assigned is the sender or the sender has received a more recent update from the assigned agent using $s_{km} > s_{im}$, then the sender's information is more up to date thus their data will be more accurate. This could be either that a better bid was placed or that the agent is no longer assigned to the task. Either way the receiver will replace its bid data for that assigned agent m with the senders bid data. After comparing all the assignments receiver i has for task j , the receiver can be sure its assignments for that task are accurate, however, the sender may have new assignments that are better than the current assignments dealt with in the next phase.

During the second phase (12-23, Figure 3.3) the receiver's information is updated with new assignments from the sender. From the first phase an agent knows that all of its assignments, according to the sender, are currently up to date. Following this the receiving agent can proceed through each new assignment and evaluate whether the new bid is better. When the sender k has an agent m assigned to task j that is not the receiver nor is it assigned by i because phase 1 has dealt with this situation already and the sender has a better update time the new assignment can be validated. Two situations can occur, either the multi-agent task still has space, in which case the assignment will be added, or the task is full and the bid must be compared. The receiver checks all bids assigned to find the minimum bid, which is then compared to the new assignment keeping the highest bid of the two competing assignments.

When the algorithm replaces an agent in a current group it must replace an agent that is carrying at least one piece of identical equipment, if the task requires specific equipment, as the bidding process will not let a group form without meeting the required equipment list. There is still space in an assignment if $\sum_m(x_{imj} = 1) <$

L_j , but the new assignment must contribute to the correct equipment list. The current equipment list is calculated as $\{ae_i | x_{ij} = 0\}$ which provides the set of equipment of all assigned agents. Thus agent m can be assigned if $ae_m \in te_j \setminus \{ae_i | x_{ij} = 0\}$ that is true if there is still a requirement for an agent m with specific equipment ae_m without the need to replace an assigned agent. If there are not available spaces in the group then another agent must be replaced using

$$(\min y_{inj} \forall n) < y_{kmj} \text{ and } ae_n = ae_m \quad (3.15)$$

to find, if feasible, an agent with the same equipment as agent m but with a lower contribution score thus replacing that agent would provide an overall higher score. If these conditions are met then the algorithm can replace the lower scored agent.

If two agents both place the same bid for a task there is the chance of the agents creating a deadlock. This can be created by both agents refusing to leave the task with the assumption that the other agent has lost and will leave, alternatively they could both assume they have lost, leaving the task and re-assigning later. To avoid any chance of deadlocking there must be a system in place to prevent such situations. The agent with the highest ID is given priority that provides a systematic way for all agents to agree on the winner in the case of tied bids.

The consensus phase continues checking the assignment data received for each bid until finally the agent updates its time stamp s_{ik} with the current simulation time. The CBGA iterates between the bundle phase and the consensus phase until all agents involved coverage to the same conflict free solution. A brief assignment involving the CBGA and multi-agent tasks can be seen in Appendix A.2.

3.3 PERFORMANCE

3.3.1 Methodology

The CBGA is extended from a Matlab implementation of the CBBA; the algorithms are tested and compared using simulated experiments. The simulation parameters are kept as close as possible to the setup used in previous research (Choi et al., 2009) (Choi et al., 2010). The three main variables examined are the number of agents, number of tasks and number of agents required for each task. Changing these variables explores a large breadth of possible assignment situations, for example, testing consensus on tasks that require a large number of agents resulting in an assignment that requires significant cooperation. Covering a range of possible simulation parameters helps defend the algorithms performance and use at theoretically any number of agents, tasks and agent requirements of those tasks. With the increased complexity of the requirements of tasks it is important to consider the effect this has on the communication between agents. In the potential practical application of this algorithm communication bandwidth is an important aspect to test. The tasks have increased complexity in their requirements; requiring a longer decision making process with more information to communicate to achieve consensus and thus an overall increase is expected in the communication requirements, however, this should only be a linear increase on a per assignment basis.

Comparisons with existing methods are difficult as previous algorithms solve a specific aspect of the multi-agent task assignment problem. The extension of the CBBA for multi-agent tasks that solves ‘duo’ tasks performs a comparison of the percentage score improvement over the single agent solution provided by the CBBA. A similar comparison is made between the CBBA and the CBGA, although the experiments are setup to be as similar as possible there are still some differences between the two setups. However this comparison still has some value in showing the

possible improvement and strengths of the new algorithm and at the very least can indicate a similar quality of assignments produced by the CBGA.




3.3.2 Test Scenario

Specific scenarios were randomly generated to test different aspects of the CBGA in dealing with different requirements. Additionally comparisons were drawn with the original CBBA and the CBBA solution to duo tasks. Comparing the algorithms each test contains 20 tasks with a varying number of agents where $N_n \in \{1, \dots, 10\}$. The simulation environment is a 3-D space of size $W \times W \times H$ ($W = 5km$, $H = 2km$) with agents randomly placed on the floor and tasks randomly placed anywhere in the environment. Tasks are given a random start time window that lasts 5m within which agents must begin the task, additionally each task takes 5m to complete. The time-distance discounted reward in (3.10) is used to define the scoring function with a fixed reward $\bar{c}_j = 100$, time-discount $\lambda = 0.8$ and distance discount $\gamma = 0.8$. Each agent moves at a constant speed of 20 m/s.

The objective of each experiment is to maximise the total assignment score where a higher score is indicative of a better assignment from a shorter travel distance, timely task completion and greater number of valid assignments. The overall score of each experiment is the sum of the scoring function for each agent's path. Multi-agent tasks defined as requiring more than one agent will reward a fixed score to each agent involved signifying the difficulty and importance of such tasks. Both the amount of agents required and the specific equipment needed is modified to test the quality of assignments under certain circumstances. Observations will be made on the overall impact on the score and the number of communications agents require to come to a consensus. Communication between agent i and agent k where assignment data is sent is counted as a single communication step. Each specific experiment setup is run 100 times as is consistent with experimental data for the CBBA and the CBBA with

duo tasks (Choi et al., 2009) (Choi et al., 2010). For each experiment the average score and communication data is plotted to show trends, agent movement during a typical experiment is also shown to display the algorithm assigning agents correctly to the available tasks and to draw discussion on assignment quality.

Table 3.4: Icon key for agent path graphs.

	Agent start locations, different colours represent different type i.e an agent with a different piece or set of equipment.
	An agent's path through a simulation, agents of the same type are coloured the same.
	Task location, the crosses represent the start and end time window of when agents can attempt to start the task. Tasks with differing requirements are coloured and labelled differently.

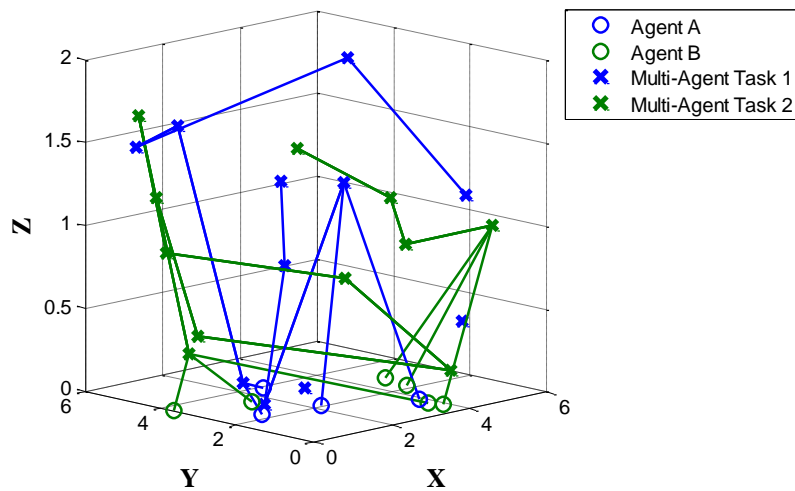


Figure 3.4: Plot of agent paths and their assignments in X Y and Z.

Agents complete tasks in a three dimensional environment to effectively simulate UAVs, all agents begin a simulation at a random location on the ground. Figure 3.4 shows the paths of ten agents completing twenty multi-agent tasks where task 1 requires two agents of type A and task 2 requires three agents of type B. Displaying the agent paths in this way does not provide enough useful information

and it is problematic to visualise the order in which agents are completing tasks or which agents are doing a task.

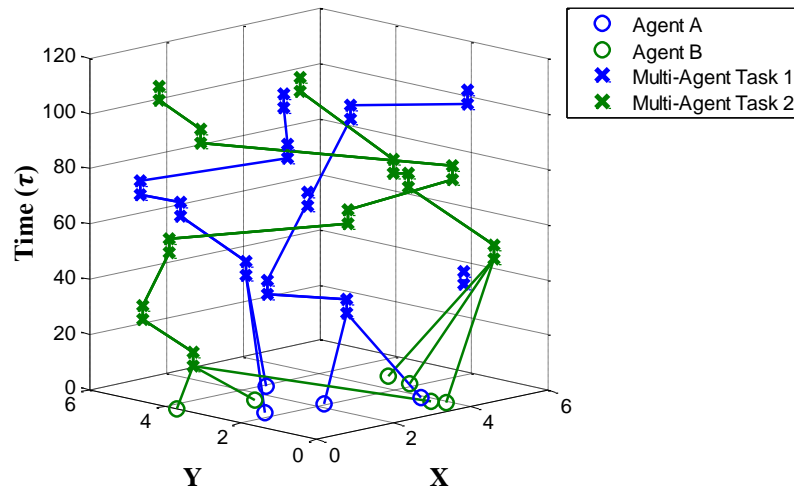


Figure 3.5: Plot of agent paths and their assignments through time in X and Y.

Figure 3.5 improves the display of agent's paths by removing one dimension and introducing time; this alleviates some of the problems and helps produce a better image of agent movement through the simulation. All agents begin the simulation on the ground plane, as time progresses agents move upwards through the graph, although this is not necessarily upwards in the 3D environment. Whilst this method of presenting the data is better than focusing on the pure coordinates of agent actions, it is still partially difficult to see what is happening although it provides a clearer picture. The most important information required to visualise the simulation is a metric of movement over time, reducing the number of dimensions displayed down to one is sufficient enough to convey movement through the simulation whilst still differentiating between each agent (excluding when agents move together between tasks). Figure 3.6 displays only the x coordinate of each agent as it changes through time displayed on the y axis. With each task having a specific time window in which it can be started this method provides a better way to visualise the activity in the simulation. An important property to note is that two tasks could be displayed in a

similar area on the graph but in practise are potentially very far from each other, generally this is not a problem and overall this method works well for displaying movement and activity. Whilst the simulation displayed in Figure 3.5 is quite complicated Figure 3.6 cleans up most of the difficulties allowing an easier visualisation of the movement and assignments of agents through the simulation, this method will be used to display agent assignments throughout this thesis.

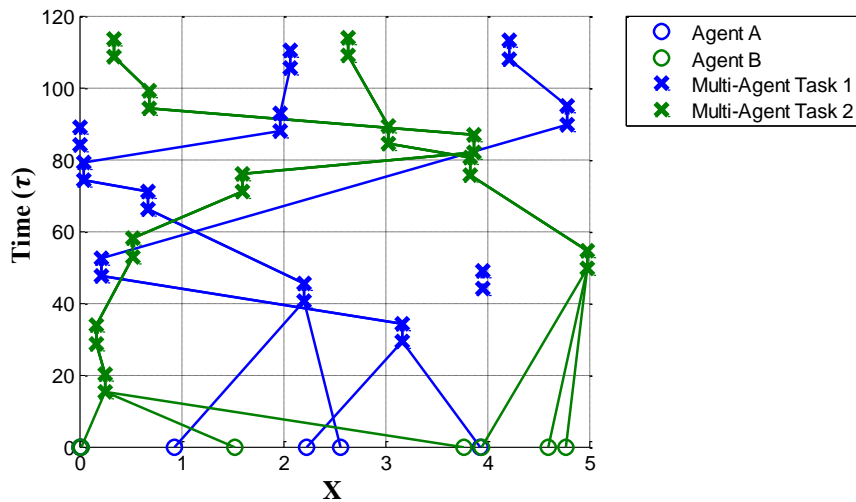


Figure 3.6: Plot of agent paths and their assignments through time and position X.

3.3.3 Multi-Agent Task Allocation

The CBGA algorithm is first tested to determine if it can come to a conflict free assignment for a number of situations that test the single and multi-agent task assignments. A conflict free solution is defined as an assignment in which all agents agree with the agent to task assignments. A conflict is said to have occurred if, for example, two agents are assigned to a task that only requires one agent. Although the algorithm has been created for multi-agent tasks the algorithm must still function for single-agent tasks, Figure 3.7 shows the algorithm successfully assigning agents for single-agent tasks and providing a conflict free solution. It can be seen that each task

requiring only one agent has exactly one agent assigned to it over the duration of the experiment.

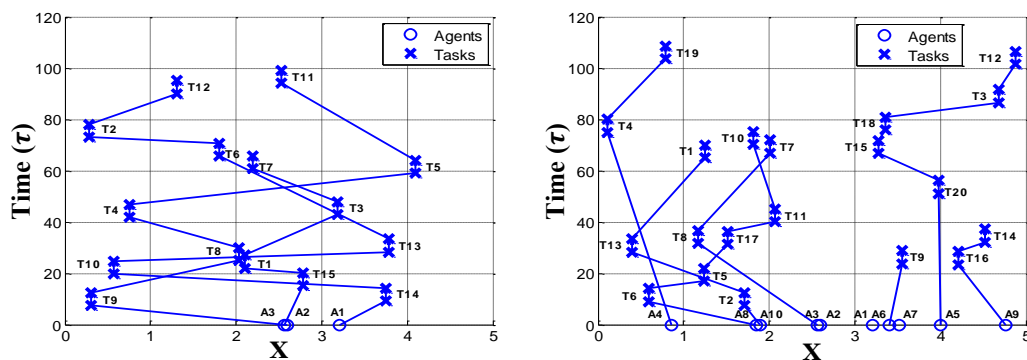


Figure 3.7: Agent paths with position (X) over time (τ). Both experiments show conflict free solutions for single-agent tasks. First experiment (left) has three agents completing fifteen tasks. Second experiment (right) has ten agents completing twenty tasks.

Multi-agent tasks are tested to show that agents are able to correctly assign multiple agents to a task. In Figure 3.8 it is shown that the CBGA can successfully converge on a solution for tasks requiring two agents. Upon completing the first assignment agents often move as a group to complete further tasks for the most efficient assignments. The second experiment in Figure 3.8 shows another multi-agent assignment, however, this time the number of agents $N_n = 10$ does not split evenly with the multi-agent requirement $L_j = 3, \forall j \in J$. Interestingly rather than leaving an agent on the side line the CBGA uses the spare agent to marginally improve the score provided by some of the tasks. The second experiment in Figure 3.8 provides an assignment score of 2777, removing the spare agent ‘A8’ the total assignment score is reduced to 2653.

In some cases, as seen in the second experiments, agents can switch between multi-agent tasks where the optimal choice would be to stay with the same agents and continue to complete multi-agent tasks together. This is caused by the way agents individually build up their own bundle of tasks to complete. An agent might add a

task close to its starting location as it would receive a high reward for completing that task, upon adding other tasks to the bundle that are earlier on the agents path can produce situations where agents split up. This situation could be avoided by focusing assignment of tasks to tasks that can be completed first but this would provide its own set of problems where agents would travel unnecessary distances simply because the task scheduled earlier. Both situations have their advantages and disadvantages primarily caused by the method of bundle creation.

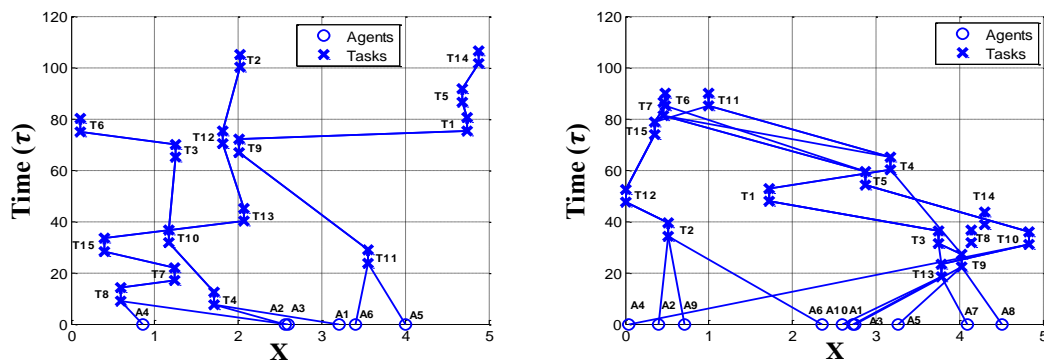


Figure 3.8: Agent paths with position (X) over time (τ). Both experiments show conflict free solutions for multi-agent tasks. First experiment (left) has four agents completing fifteen multi-agent tasks that require two agents each. The second experiment (right) has ten agents completing fifteen tasks that require three agents each.

It is also necessary to show that both systems can function together, that the algorithm can handle both single-agent tasks and multi-agent tasks in the same simulation. Figure 3.9 confirms that agents can assign the correct number to each task, specifically looking at tasks 3, 4 and 5 it shows how two agents group together to complete task 5 that requires two agents, then instead of both going to task 4, one of the agents completes task 3 en route then reunites with the previous agent to complete task 4. This experiment also shows that the algorithm does not necessarily need to use every available agent; here agent 2 is not required at all because a better score is produced by not using this agent.

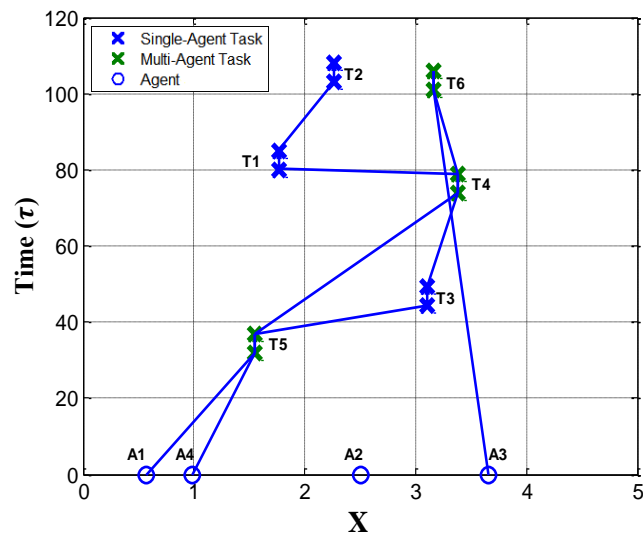


Figure 3.9: Agent paths with position (X) over time (τ). Four agents complete three single-agent and three multi-agent tasks that require two agents each.

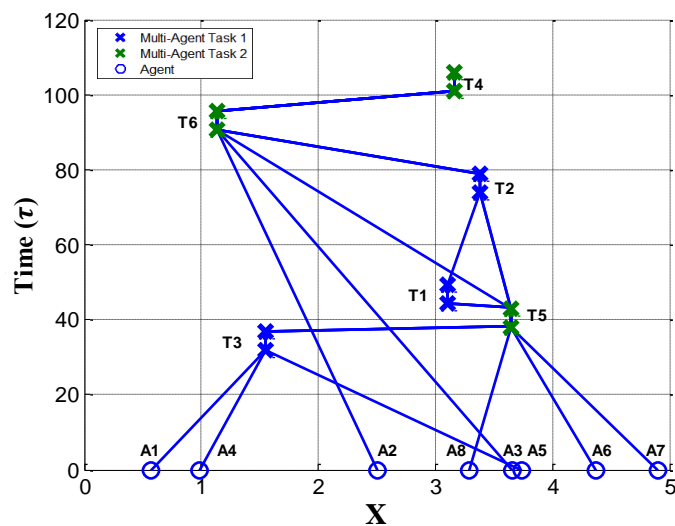


Figure 3.10: Agent paths with position (X) over time (τ). Eight agents complete two different types of multi-agent tasks. Task 1 requires three agents and task 2 requires six agents.

Finally Figure 3.10 illustrates that the algorithm can handle larger group requirements. This experiment contains two different tasks, one requiring three agents and another requiring six agents. Theoretically the algorithm can function with tasks requiring any number of agents although, as the requirements increase, so too can the computation time as shown later. Although as agents travel together it is not possible

to see how many agents are going to each task, but using Table 3.5 it shows the final conflict free assignment that the agents converged on. In this particular experiment agent 1 and 5 both finish after they have completed task 1.

Table 3.5: Winning assignment bids for the experiment displayed in Figure 3.10 where each task provides a fixed reward of $\bar{c}_j = 100$ modified by the cost of travel calculated from the marginal score improvement (3.11).

		Agents							
		A1	A2	A3	A4	A5	A6	A7	A8
Tasks	T1	99.6			99.6	99.5			
	T2				95.9		95.9	95.9	
	T3	95.7			95.7	94			
	T4		94.6	94.8	97.1		97.1	97	94.7
	T5	94.3			94.2	94.2	93.1	93.9	95.3
	T6		95.4	93.7	92.9		92.9	92.9	90

Investigating the effects of assignment and consensus of multi-agent tasks comparisons are made between the previous algorithm the CBBA and the extended algorithm the CBGA. The CBBA is used to solve single-agent tasks and the CBGA is used to solve multi-agent tasks, in addition comparisons are drawn to a setup involving both multi-agent and single agent tasks.

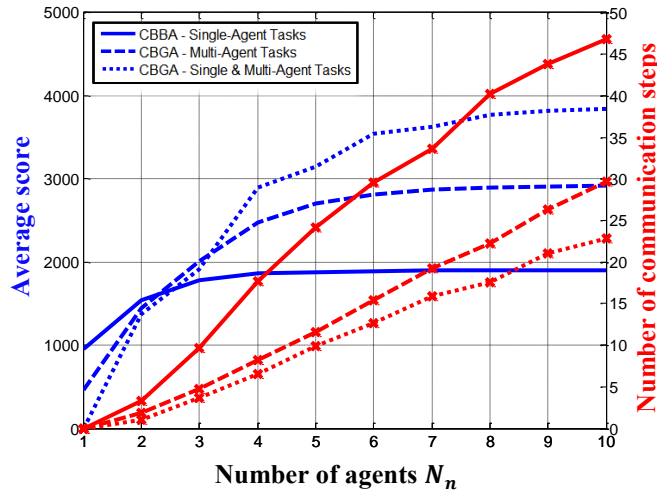


Figure 3.11: Comparison of average score and number of communication steps for consensus between the CBBA, CBGA and using both. Experiments contain twenty randomly generated tasks and multi-agent tasks require two agents.

Figure 3.11 has three experiments plotted that tested both algorithms, single-agent tasks use the CBBA and multi-agent tasks use the CBGA. The first experiment used just the CBBA where each task required a single agent to complete it. The second experiment tested the CBGA by requiring two agents to complete each task the assignments are seen in Figure 3.12. The final experiment used both types of tasks making the agents reach a consensus on assignments for ten multi-agent tasks and ten single-agent tasks. In the experiment the multi-agent tasks initially provide lower scores than the single-agent tasks but as the number of agents increases a greater increase in score is observed. Interestingly the total number of communication steps actually decreases with the introduction of multi-agent tasks, this is significant when noted that the tasks in the second experiment double the total number of assignments required for consensus from twenty assignments to forty because each multi-agent task requires two agents instead of one agent thus two assignments.

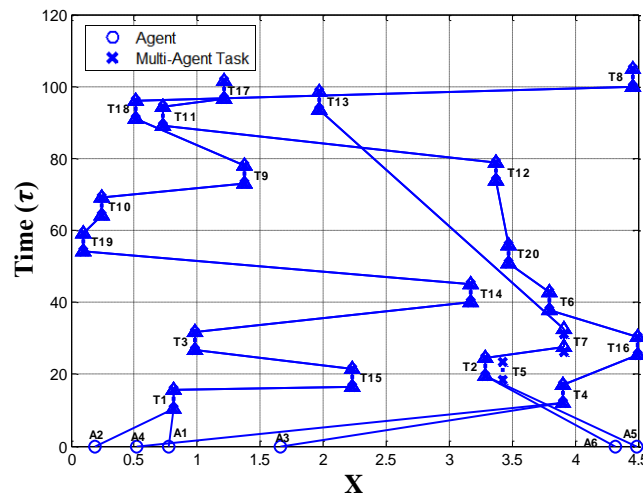


Figure 3.12: Cross section of agents movement through time (τ) and the X axis to complete twenty multi-agent tasks. Each task requires two agents for completion.

Looking at the movement of agents in Figure 3.12 presents reasons why communication drops are observed for multi-agent tasks with the CBGA. Between $\tau =$

0 and 20 each agent assigns and completes its initial task along with another agent. After completing the first task agents commonly stay together for succeeding tasks, with the closest task yielding the highest reward neither agent needs to dispute the best choice. Occasionally two groups may attempt the same task, which will then require consensus but overall each self-made group continues through the simulation effectively as one entity. With 5 agents mean communication steps decreased significantly (decrease of 14 steps) between the CBBA (24 ± 8 steps) single agent tasks and the CBGA (10 ± 2 steps) multi-agent tasks as shown in Figure 3.11. At 10 agents mean communication steps for consensus decreased further (decrease of 24 steps) showing a significant communication drop for consensus from single agent tasks (47 ± 14 steps) to multi-agent tasks (23 ± 4 steps). Improvements are significant to $p < 0.01$ for statistical significance at 1%.

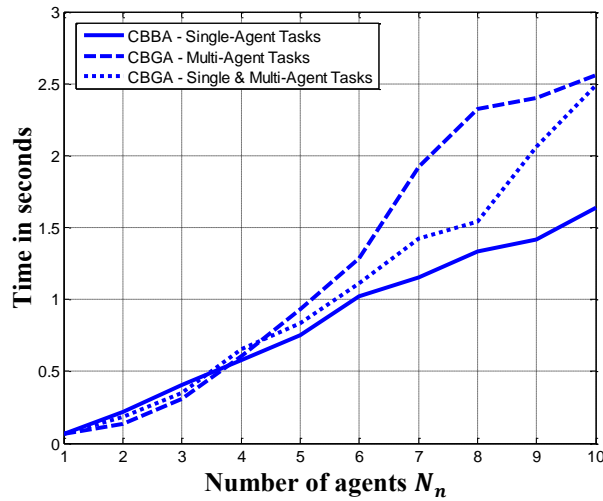


Figure 3.13: Computational times for running each experiment in Figure 3.11 CBBA assigns only single-agent tasks, the CBGA assigns multi-agent tasks and mix requires both the CBGA and CBBA.

As expected the computational times in Figure 3.13 show the CBGA takes longer to come to a consensus, this was expected due to the increased complexity of assignments. The CBBA in Figure 3.11 had to solve 20 assignments, 1 per task,

alternatively the CBGA had to solve 40 assignments because each task required 2 agents. Comparing computational time and the number of communication steps; the CBGA takes a longer time to compute the consensus when receiving new assignments, but requires less overall communication between agents to achieve the final consensus. This increase in computational run time is still acceptable as the base CBBA converges very quickly (Choi et al., 2010).

The preferred duo cooperation algorithm (Choi et al., 2010) was developed to address a limited version of the multi-agent task assignment problem. A test scenario was setup where tasks were classed as “preferred duo” which meant they required one agent and would reward a score of 100, however if a second agent assisted with the task they would receive an additional reward of 50 for a total score of 150. This was compared to the CBBA assigning agents to single-agent tasks rewarding a score of 100. Naturally this was always going to provide an improvement over the CBBA because the preferred duo tasks provide a higher potential score per task. Figure 3.14 shows the resulting improvement of the preferred duo algorithm over the CBBA. Using a similar setup the CBGA was tested where tasks required two agents and gave a reward of 150, comparing the resulting improvement or reduction in score over the CBBA and its single-agent tasks rewarding a score of 100.

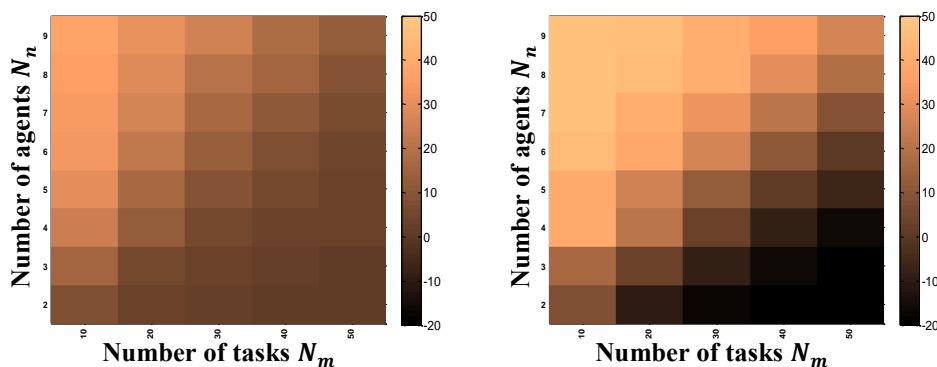


Figure 3.14: Percentage score improvement by using the preferred duo cooperation (left) (Choi et al., 2010) and the CBGA (right) over the original CBBA where darker squares show lower or no improvement and lighter squares show a higher improvement.

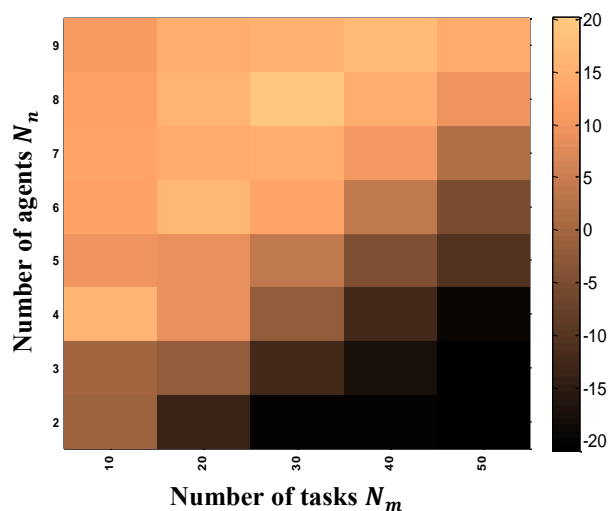


Figure 3.15: Difference in percent score improvement by using the CBGA over the CBBA preferred duo cooperation where darker squares show a decline in score and lighter squares show an improvement in score.

Whereas the preferred duo algorithm provides an improvement across the board the comparison between CBBA and CBGA shows poorer percentage score improvement with smaller agent numbers. This loss is caused by the inability for the CBGA to let a specific task be completed by either 1 or 2 agents as found with the preferred duo tasks. With small agent numbers compared to the total number of tasks completing two single-agent tasks in the CBBA provides a score of 200; however the CBGA cannot do this as agents are forced to complete the task in teams of two only providing a score of 150. Once the ratio of agents to tasks is greater the cooperation of agents becomes more rewarding with less unassigned tasks. The preferred duo scoring gives an advantage over the CBGA scoring better at high task to agent ratios. This issue is still prevalent at increased agent numbers but is offset by the assignment quality of the CBGA, showing percentage increases of up to 20% over the preferred duo cooperation as can be seen in Figure 3.15. These results show a greater performance improvement at higher levels of cooperativeness, where the best scores

are achieved by cooperatively completing as many tasks as possible because there is a shortage of tasks.

Algorithm 7: Sequential Greedy Algorithm for Multi-Agent Assignments

```

1: while  $y \neq 0$ 
2:    $y = 0, i = 0, j = 0$ 
3:   for  $\tilde{i} = 1$  to  $N_n$ 
4:     for  $\tilde{j} = 1$  to  $N_m$ 
5:       if  $\tilde{j} \in b_{\tilde{i}}$ 
6:          $c_{\tilde{i}\tilde{j}} = 0$ 
7:       else
8:          $c_{\tilde{i}\tilde{j}} = \max_{n \leq |p_{\tilde{i}}|} S_{\tilde{i}}^{p_{\tilde{i}}\theta_n\{\tilde{j}\}} - S_{\tilde{i}}^{p_{\tilde{i}}}, \forall \tilde{j} \in J$ 
9:       end if
10:      if  $(c_{\tilde{i}\tilde{j}} + \sum_{\tilde{n}=1}^{N_n} x_{\tilde{n}\tilde{j}}) > y$  and  $L_j > (\sum_{k=1}^{N_n} x_{ikj} > 0)$ 
11:         $y = (c_{\tilde{i}\tilde{j}} + \sum_{\tilde{n}=1}^{N_n} x_{\tilde{n}\tilde{j}})$ 
12:         $i = \tilde{i}, j = \tilde{j}$ 
13:      end if
14:    end for
15:  end for
16:   $n_{i,j} = \operatorname{argmax}_n S_i^{p_i\theta_n\{j\}}$ 
17:   $b_i = b_i\theta_{\text{end}}\{j\}$ 
18:   $p_i = p_i\theta_{n,i,j}\{j\}$ 
19: end while

```

Figure 3.16: Implementation of a sequential greedy algorithm to solve the multi-agent task assignment problem in a centralised system.

In Figure 3.16 a centralised solution to the multi-agent task assignment problem is presented that sequentially adds the greediest assignment at each iteration. The sequential greedy algorithm calculates the best marginal score improvement for every combination of task \tilde{j} to agent \tilde{i} given the agents current path $p_{\tilde{i}}$. If the assignment is valid such that a multi-agent task is not full and the marginal score improvement is better than any other score then the assignment is added. The algorithm repeats this process of calculating the marginal scores and adding the best until eventually there are no more valid score improvements and the algorithm ends.

This provides a greedy centralised solution to the multi-agent task assignment problem.

The CBBA was shown to provide the same solution as a centralised greedy algorithm; similarly the CBGA will be shown, with experimental results, to provide a similar result to a centralised solution. Consider the sequential greedy algorithm for multi-agent assignments (SGAMA) in Figure 3.16 that sequentially adds the best score improving task agent pair at that point in time. Continually updating the marginal score improvements of each agent the best agent task pair is added until no further improvement for the solution is found. This algorithm is a centralised solution where a central hub is given access to every agent’s scoring scheme and is able to add tasks to any agent’s bundle and path.

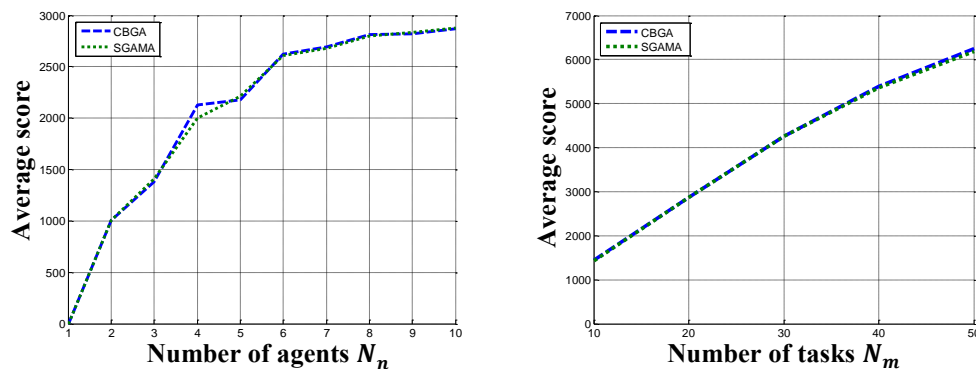


Figure 3.17: Average assignment score achieved by the CBGA and the SGAMA completing 20 tasks with N_n agents (left) and 20 agents completing N_m tasks.

Figure 3.17 shows how the CBGA provides very similar results to the SGAMA. Although the resultant assignments are similar the results are not exactly the same with small variation in the averages. Nevertheless neither produces a statistically different result. However, comparisons with the SGAMA reveal a flaw with the CBGA as shown in Figure 3.18. As the number of agents required per task L_j increases the CBGA scores similarly to the SGAMA until the ratio of agent

requirements L_j to the number of agents N_n is such that $\frac{N_n}{L_j} \leq 2$. At this point the CBGA provides a significantly worse score than the centralised solution. The assignment graph shows that agents are producing a number of invalid, although still conflict free, assignments that produce no score. This problem is caused by agents maximising individual score over the teams focus on completing multi-agent tasks. Chapter 5 will examine this situation and develop a solution.

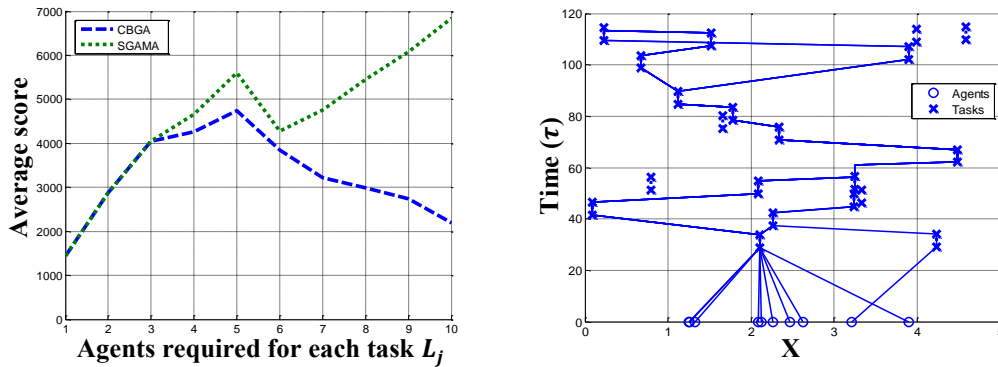


Figure 3.18: Average assignment score achieved by the CBGA and the SGAMA when agent requirements L_j are increased with 10 agents and 20 tasks (left). Assignment of 10 agents to 20 tasks with each task requiring 10 agents (right).

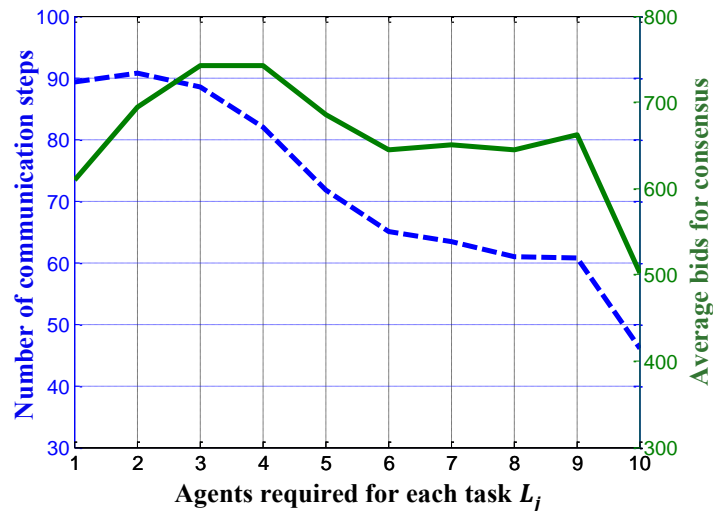


Figure 3.19: Comparison of the average number of communication steps and number of bids required to come to a consensus as the number of agents required per task increases. Experiments had 10 agents completing 20 tasks with increasing agent numbers required on tasks.

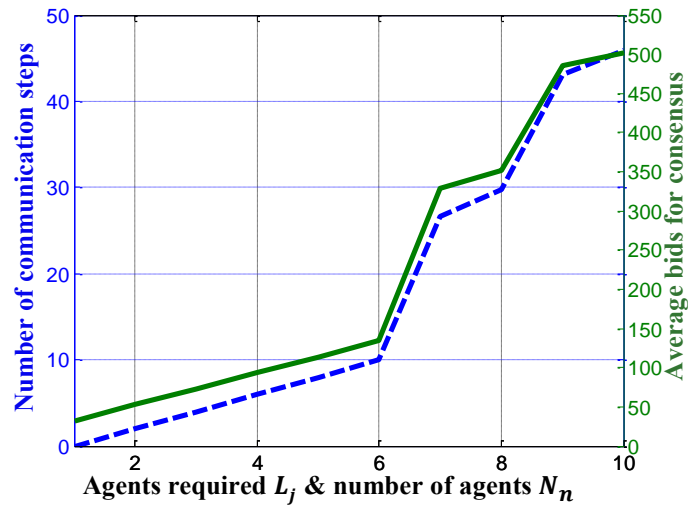


Figure 3.20: Effect on average communication steps and bid data as both the number of agents and required agents for multi-agent tasks increases.

Testing the effects of agent requirements on tasks Figure 3.19 shows the average number of bids required to come to a consensus on assignments as well as showing the number of communication steps it took. The number of agents and tasks are fixed at 10 and 20 respectively and with each set of experiments the number of agents needed for each task was increased. It can be seen that as the agent requirement increases the average communication steps decreases, a communication step is defined as the number of instances of receiving a communication message from another agent. The average number of bids has less correlation with the agent requirement, introducing some multi-agent tasks increases the number of bids for consensus but as the requirement increases closer to the maximum number of agents in the simulation the bids required drops. It seems likely that the overall decrease in bids and communications made is a result of the poor performance of the CBGA due to the low ratio of agent requirements to the number of agents. The decline observed in Figure 3.19 is consistent with the drop in performance from the centralised algorithm in Figure 3.18. Changing the number of agents in each set of experiments to match the required agents for tasks, a direct correlation is observed in Figure 3.20

between the communication steps and number of bids for consensus. This suggests that by increasing the number of agents each time to match the required agents per task both sets of data increase with a strong correlation to each other.

3.3.4 Restricted Task Assignments

Introducing heterogenous agents and further requirements on tasks increases the complexity of tasks. Initially tasks are limited to one type of agent and agents have one piece of equipment that defines their type $|ae_i| = 1$. Figure 3.21 presents two simulations that were run where the system was populated with two different tasks each requiring a different type of agent, here the blue agents complete the blue tasks and the green agents complete the green tasks. This functionality continues to work with the inclusion of multi-agent requirements as can be seen in the second experiment where each task requires two of the specified agents.

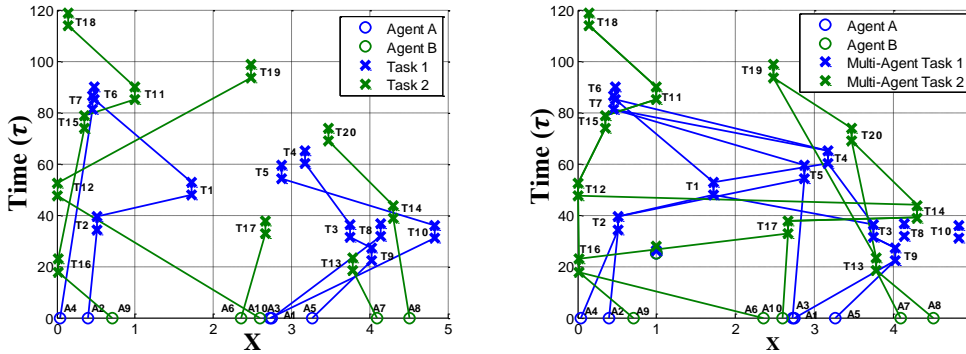


Figure 3.21: Agent paths with position (X) over time (τ). First experiment (left) shows two different types of agents completing two different types of single-agent tasks. The second experiment (right) shows the same group of agents and tasks except the tasks are now multi-agent tasks requiring two agents each.

Extending this experiment tasks can specify how many agents of a given type it requires, the two experiments in Figure 3.22 display tasks that specifically require one of each agent type such that $te_j = \{ae_i, ae_k\}$ where $ae_i \neq ae_k$ and $|ae_i| = |ae_k| = 1$.

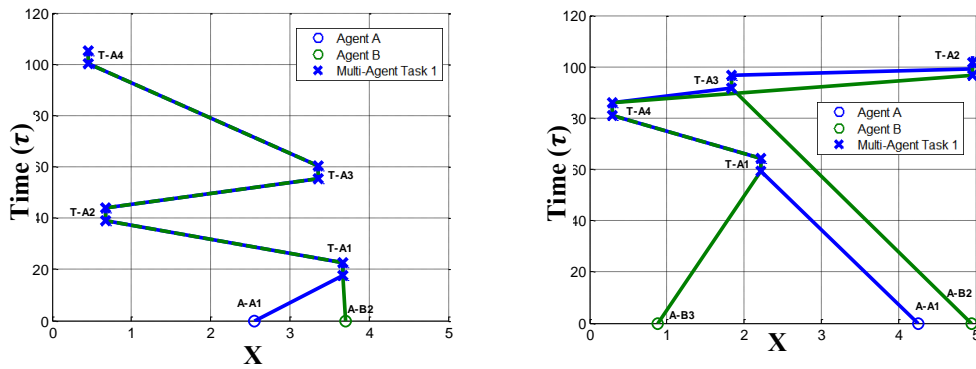


Figure 3.22: Two experiments both with multi-agent tasks that require two agents one of type A and one of type B.

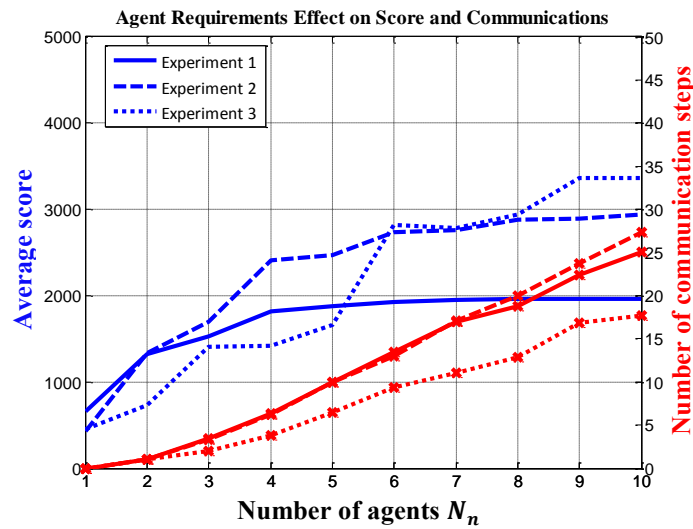


Figure 3.23: Comparison of average score and number of communication steps required to reach a consensus for tasks that require multiple different heterogeneous agents.

Figure 3.23 shows three experiments involving a varying number of heterogeneous agents and twenty tasks of differing types. Experiment 1 contains an even split of two agent types A and B completing single-agent tasks of which half require agent A and the other half require agent B. In experiment 2 ten single-agent tasks are split evenly between agents A and B, in addition the final ten tasks are multi-agent tasks that require both agents A and B. Finally experiment 3 contains three types of agents A, B and C along with three different tasks. The first task is a single-agent task that requires agent A, the second task is a multi-agent task that requires

agent A and B, finally the third task is another multi-agent task that requires all three agents. Agents are split evenly between the three types with extra agents created firstly as type A then type B. A split of 8-6-6 is created between the three tasks A, AB and ABC. The results in Figure 3.23 show that when the restrictions on tasks are high the number of communications required for consensus is lower as seen from experiment 3. With multi-agent tasks the assignment score does not always improve linearly in relation to the number of agents in the simulation, in the case of experiment 3 the significant increases in score are observed by examining the introduction of an additional agent of type C. This addition allows more of the multi-agent tasks to be completed that provide a better score for the team thus providing a more significant increase in score. Figure 3.24 shows a typical assignment of experiment 3 with reduced tasks for visual clarity. A reduction in the communication required to meet a consensus is observed from Figure 3.23 for experiment 2 3 where a task requires all three equipped agent types. These results might be a consequence of the time constraints on the tasks which will limit the available options from the maximum 20 tasks down to a much easier to manage set of the earliest obtainable.

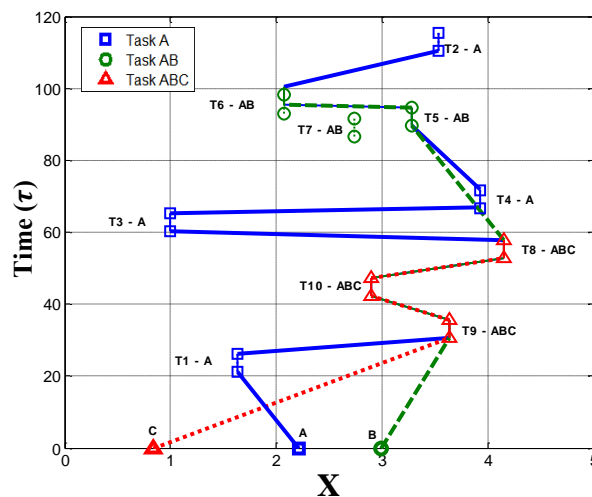


Figure 3.24: Agent's paths through time for 3 agent types (A,B,C) completing 3 different types of tasks (TA, TB, TC).

In Figure 3.24, for equipment dependant multi-agent tasks, it is seen how agent C has very little choice in its assignments and causes no conflict with other agents because it must depend on its teammates to arrive and aid in its tasks. Agent A freely moves between its tasks and, when required, aids its teammates. The reduced options for each agent greatly reduces the length of communication time required between team mates. More importantly the reduced amount of conflicts caused helps agents come to a quick consensus with smaller communication exchanges.

Figure 3.25 and Figure 3.26 show the introduction of tasks that have prerequisites where a specific task must be scheduled for completion before the task with the prerequisite requirement is assigned. Experiment 1 contains no tasks with prerequisite requirements such that $P_j = \emptyset, \forall j \in J$. Experiment 2 and 3 both have task dependancies on half the tasks where for 10 tasks $P_a = \emptyset$, 5 tasks require a specific task from the first group completed $P_b = \{a\}$ and the final 5 tasks require the completion of a specific task from the second group for example $P_c = \{b\}$. It shows that compared to CBGA tasks found in experiment #1, the number of communication messages sent to reach a consensus is usually lower with the additional restrictions. By putting these prerequisite requirements on half the tasks in the simulation the number of tasks that agents find conflict over is reduced, with the follow up tasks having fewer conflicts. Experiment #2 contained a problem where the task time window for completion was randomly generated. In some cases this meant a task with requirements was set before that of the requirement. This error created a number of tasks that could never be achieved and therefore limited the overall score obtainable. Interestingly when these time limits were removed in experiment three, the average score decreased even though more tasks had become available. Perhaps due in part to the fact that only agents who were involved in the prerequisite would attempt the follow up task, but often they would be busy completing other tasks. Although in contrast after opening up accessibility on these tasks the overall communication levels

increased. When tasks are made accessible to everyone points of conflict are amplified and therefore the number of communication steps required to come to a consensus is also increased. By limiting tasks to a small subset of agents the overall requirements on communication for consensus decrease.

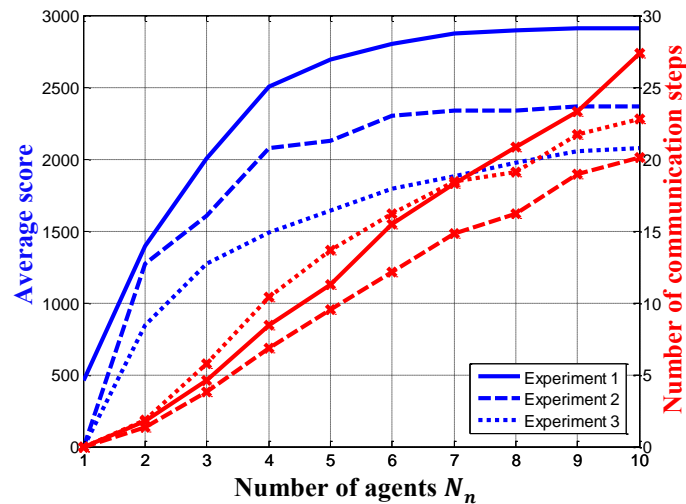


Figure 3.25: Comparison of total score and number of communication steps for tasks that require previous tasks completed. Experiment 1 used multi-agent tasks with no task dependency. Experiment 2 and 3 both added task dependencies but Experiment 3 removed the time requirement on follow up tasks.

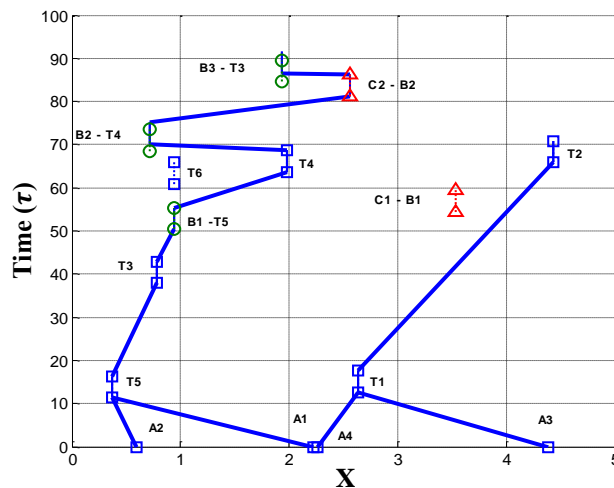


Figure 3.26: Agents 'A' movement through time and the X axis. Tasks marked 'B' require the corresponding task 'T' to be completed first, similarly tasks marked 'C' require a corresponding task 'B' completed.

Figure 3.26 shows the successful assignment of tasks requiring previous tasks to be completed as well as each task requiring two agents. The CBGA can successfully handle multi-agent tasks as well as deal with other restricted assignments on tasks including equipment limits and task requirements.

3.4 CONCLUSIONS

This chapter presented an extension of the CBBA that solves the multi-agent task assignment problem with group, equipment and task based dependencies. Communication increases were expected with the introduction of multi-agent tasks, which require an increase in information to reach consensus. However, multi-agent tasks were in fact shown to require less communication messages in certain situations to come to a consensus. These results might be derived from the time constraints on tasks, which limit the available options from the maximum 20 tasks down to a more manageable set of the earliest obtainable tasks. Added to this with constraints on which agents can perform each task further reduces the set of achievable tasks for any one agent. The necessity to bid over tasks and form consensus gradually disappears as we tighten the restrictions on each task. In some cases agents do nothing as they are not required, although it is far better for the group as a whole if they do not move, therefore causing no cost on travel or an increase in communications. This situation potentially brings the overall average communication down, where the agents working towards tasks are communicating more than the averages would suggest.

As the number of agents in the simulation increases the overall score reaches its full potential as agents complete all the available tasks. Eventually the only increase in score is caused by a greater chance of an agent starting near a task than is the case with fewer agents. For multi-agent problems agents group up and stick together to complete tasks, in some cases for the entire simulation as seen in Figure 3.8. With increasingly complicated group and equipment requirements it is found that

groups continue to work together where possible, but often an agent will leave to complete another task and merge back again to work on a later task. In some respects it can be said that when cooperation is a requirement it in fact simplifies the problem rather than complicates it.

Compared to a centralised solution in the SGAMA the CBGA performs almost identically in most situations, although this also highlighted a problem with the CBGA for a low ratio of agents required L_j to the number of agents N_n in the system. Agents are failing to agree on a single path when the number of agents and the required agents on tasks creates a single scoring path through the simulation.

Figure 3.19 and Figure 3.20 show some correlation between the number of agents and the agent requirements on tasks for the number of bids taken to come to a consensus. When the number of agents equals the agent requirement $N_n = L_j$ a steady increase in the number of bids required is observed, this is caused primarily from the increase in agents, for every extra agent in the simulation another set of bids needs to be communicated for consensus.

The initial introduction of multi-agent tasks increases the communication and bids required for consensus but once the amount of agents each task is requesting is similar in size to the number of agents in the system both communication and bids reduce. When assigning 10 agents to complete tasks that require 10 agents each the only complication is in deciding which task to complete first, after this point all the agents can agree fairly straightforwardly which task is the next best. Whilst multi-agent tasks reduce the overall communication steps required compared to single-agent tasks they increase the bid data needed, as the complexity of the multi-agent tasks increases though both communication steps and bid data decreases. Another issue with the solution to multi-agent tasks is the arrival times of each agent. Currently the time window for a task to start is small enough so that even if agents do not arrive at the same time the resulting delay is minimal and as agents can only assign themselves

to tasks they can start in time, the task is unaffected. However, if the start time for tasks was made greater or the start time was completely removed agents would then need to agree upon when they arrive at a task as well as who will be assigned to it. This issue adds a huge amount of complexity to the situation where agents must also communicate their earliest arrival time and must continually re-plan their assignment when other agents want to assign themselves to the task but with a later arrival time.

Comparisons were made with an alternative method of solving multi-agent assignments, although the duo cooperation extension of the CBBA (Choi et al., 2010) is limited to solving tasks with duo agent requirements. The CBGA when similarly limited to duo tasks can score comparatively to the duo extension of the CBBA. Additionally the CBGA performs better when there are a larger number of agents in the simulation, such that there are enough agents in the system to complete all the tasks. In this case the preferred duo tasks are almost identical to the multi-agent tasks in the CBGA where the best scores are achieved in high quality assignments with two agents assigned to each task. Furthermore the CBGA also removes the limitation of up to two agents per task that the duo cooperation algorithm is limited to providing robust and conflict free assignments to complex cooperative requirements involving any number of agents.

Chapter 4: *Dynamic Multi-Agent Consensus Storage*

Chapter 3 extended the consensus based bundle algorithm to deal specifically with multi-agent tasks, when the CBBA used two vectors to store an agent's winning bid list and winning agent list (Choi et al., 2009), by extending the algorithm to solve multi-agent tasks the winning agent lists became inefficient and unnecessary. This chapter examines combining the bids list into a concise and dynamic storage for handling multi-agent tasks, additionally further changes allow the algorithm to handle a dynamic environment with knowledge sharing. This method provides improved communication and the flexibility of the algorithm to handle a dynamic environment with discoverable knowledge of agents and tasks. The CBBA requires all agents to function and store data in the exact same way including agreeing on task and agent indexes in the winning bid lists. When considering a dynamic environment where each agent can discover new agents and tasks independently it cannot be assumed task and agent data will cross-over directly. Furthermore in practice different UAVs will have software specific to their hardware (Pastor et al., 2007) (Valenti et al., 2007) (Tisdale et al., 2006) and may store their data in differing ways. The CBBA requires homogenous agents that store assignments in a specific format, any agent which is not conforming to the correct standard is unable to communicate their assignments to the other agents; in a practical application these requirements could be unfeasible. This chapter provides a solution for consenting data between different UAVs providing they can understand any communication messages about bidding data.

4.1 PROBLEM

Multi-agent tasks require multiple agents to complete them, when agents communicate between each other they need to store every agent assigned to a task and their current bid. When agreeing on the data they receive they need to know whether a

task is at its maximum capacity i.e. whether a multi-agent task that requires 4 agents already has 4 agents assigned, and if it is, then the only agent it should replace is the agent with the smallest bid. Storing knowledge about tasks that require multiple agents with a single vector is not sufficient for all the data. To create a consensus agents require the knowledge of all agents assigned to a task and their respective bids.

With the CBBA task and agent information are stored locally at the beginning of the calculation and each agent uses two vectors that are stored and communicated between each agent, the winning bid list y_i and the winning agent list z_i . Data is linked between the two vectors by using a task ID, this is an increasing value that directly indexes the vectors $J \equiv \{1, \dots, N_m\}$, similarly the agent IDs used also represent the agent index in a vector $I \equiv \{1, \dots, N_n\}$, and this works in a simulation where all agents store data in exactly the same way, but in practice this should not be assumed.

The extension to the CBBA for dealing with duo cooperative requirements continues to use vectors in the storage of bid data. Tasks that require two agents are split into two separate subtasks j_1 and j_2 with associated scores. However, it is possible for an agent to assign itself to both parts of the task and so restrictions must be placed such that (Choi et al., 2010)

$$\begin{aligned} c'_{ij_1} &= c_{ij_1} \times \Pi(J_2 \in b_i) = c_{ij_1} \times (1 - \Pi(z_{ij_2} = i)) \\ c'_{ij_2} &= c_{ij_2} \times \Pi(J_1 \in b_i) = c_{ij_2} \times (1 - \Pi(z_{ij_1} = i)) \end{aligned} \quad (4.1)$$

where c_{ij_1} is the marginal score improvement an agent i can achieve for task j which is modified to c'_{ij_1} that is equal to c_{ij_1} if the second part of the task is not assigned by i e.g. $z_{ij_2} \neq i$; otherwise the score improvement is set to 0 because the agent is participating in the other part of the task. This method works for duo tasks but becomes redundant if the system is scaled up to increasing number of agents becoming needlessly complicated.

The system proposed in chapter 3 converts the winning agent list x_{ij} from the CBAA and the winning bid list y_{ij} from the CBBA into matrices, where agent i stores $y_{ikj} = 20$ when agent k has a winning bid of 20 for task j . When these assignments are communicated another agent refers to $x_{ikj} = 1$ as agent i believes agent k is assigned to task j . This system allowed tasks to require any number of agents but the system contained some redundant data, the winning agent matrix x_{ikj} is unnecessary as each agent knows that another agent is assigned to a task by the existence of a winning bid. Additionally because each agent's bid data is stored in a separate column and each agent also has access to the specific agent that made that bid. Finally the system continued to assume that agent and task IDs would be simple numerical scaling values such that agent $i = 1$ bid data would always be stored in y_{1j} for every single agent. In a real life system each agent can be assumed to have a unique identification number but those numbers would not necessarily be in any order which is easy to store.

When transferring this data storage system over to a multi-agent task assignment problem, difficulties are caused with consistency between agents and tasks. Different tasks can have varying number of agents assigned to them, for tasks that require multiple assignments a vector cannot store data about each assignment. Therefore, with the CBGA a new system of data storage must be developed that allows access to which agents are assigned to a task (as far as that agent is concerned) and what the current bids are with the agent they were made by. Furthermore the system needs to be dynamic allowing new tasks and agents to be added during the simulation, without conflict arising from the order data is added.

4.2 ALGORITHM

4.2.1 Local Data

Originally the winning bids list y_i and the winning agent's list z_i are two vectors of length N_m where N_m is the number of tasks in the simulation as seen in Table 4.1. With these two vectors each agent can keep track of the highest bid for each task and which agent made that bid and therefore is assigned to the task.

Table 4.1: Winning bid list and winning agent list used in the CBBA.

		Winning Bid List (y_i)			
		Tasks			
		j	$j + 1$	$j + 2$	$j + 3$
Agent	i	y_{ij}	$y_{i,j+1}$	$y_{i,j+2}$	$y_{i,j+3}$

		Winning Agent List (z_i)			
		Tasks			
		j	$j + 1$	$j + 2$	$j + 3$
Agent	i	z_i	$z_{i,j+1}$	$z_{i,j+2}$	$z_{i,j+3}$

Once tasks require multiple agents both vectors must be converted into matrices to keep track of multiple contribution bids and multiple winners. This gives two matrices of size $N_m * N_n$ where N_n is the maximum number of agents in the system. However, this means agents must now communicate two matrices rather than two vectors as shown in Table 4.2.

Table 4.2: Winning bid matrix and winning agent matrix used for multi-agent assignments in CBGA. The winning agent matrix contains a binary value where $x_{ij} = 1$ signifies agent i is assigned to task j with the winning bid of y_{ij} .

		Winning Agent Matrix (x)				
		Tasks				
		j	$j + 1$	$j + 2$...	$j + m$
Agents	i	x_{ij}	$x_{i,j+1}$	$x_{i,j+2}$...	$x_{i,j+m}$
	$i + 1$	$x_{i+1,j}$	$x_{i+1,j+1}$	$x_{i+1,j+2}$...	$x_{i+1,j+m}$

	$i + n$	$x_{i+n,j}$	$x_{i+n,j+1}$	$x_{i+n,j+2}$...	$x_{i+n,j+m}$

		Winning Bid Matrix (y)				
		Tasks				
		j	$j + 1$	$j + 2$...	$j + m$
Agents	i	y_{ij}	$y_{i,j+1}$	$y_{i,j+2}$...	$y_{i,j+m}$
	$i + 1$	$y_{i+1,j}$	$y_{i+1,j+1}$	$y_{i+1,j+2}$...	$y_{i+1,j+m}$

	$i + n$	$y_{i+n,j}$	$y_{i+n,j+1}$	$y_{i+n,j+2}$...	$y_{i+n,j+m}$

With the increased data required for analysing multi-agent tasks and the reduced number of total communications needed to achieve consensus, unnecessary data can be removed by merging the two matrices into a single matrix X . This matrix contains all the winning bids and is of size $N_m \times N_n$ where N_n is the number of agents in the simulation and N_m the number of tasks. What this allows agents to do is use the row to display tasks and the column to display agents and assume that the existence of a winning bid is an agent's assignment to the task. Thus X_{ij} corresponds to the bid agent i has made for task j otherwise 0 if the agent has not made a bid. As with the CBGA using the values in each column the current total score rewarded for a task can be determined as

$$c_j = \sum X_{ij}, \forall i \in I \quad (4.2)$$

where c_j is the score for completing task j . This score is only the current reward for the task based on the assignments, the task could ultimately score nothing if the correct requirements are not met or more agents could assign themselves to the task increasing the reward. An agent can determine the number of assignments on a current task by counting the number of valid bids in the winning matrix column where $X_{ij} > 0$.

$$\sum_{i=1}^{N_n} \begin{cases} 1, & X_{ij} > 0 \\ 0, & otherwise \end{cases} \quad (4.3)$$

Using (4.3) to determine the number of assignments on a task removes the need for the original winning agent matrix x_i and if (4.3) is less than the required number of agents L_j then an agent can assign itself to the task assuming it meets any other requirements such as the correct equipment.

Additionally in a dynamic system we cannot assume each agent will store data in the same order, in a dynamic environment where agents look after their own data new tasks or agents can be stored in their memory in different orders to that of a neighbour. Therefore we cannot use the matrix index of X as a reliable identifier for an agent or task. Agents therefore need to store a separate agent vector A_i that contains all known agent IDs and a task vector T_i containing all known task IDs by agent i . These two vectors are used as lookups to the assignment matrix X . With this new matrix agents can store data dynamically and build up a list of agent to task assignments as they discover new agents or tasks in the system. When a new agent or task is discovered an agent can create a new matrix column or row respectively and the ID is added to the appropriate task or agent vector. Agents can individually build up their assignment matrix in different orders but still store and exchange the data

reliably. Update times from agents can continue to be stored in a vector s_i and are identified using the agent vector A_i .

Table 4.3: Dynamic variable storage on agents, where X_{ikj} references the winning bid agent i believes agent k has made for task j . The connection is made using the offset of the agent and task in the associated lists.

		Task List T_i				
		1	2	3	...	j
		Assignment Matrix X_i				
Agent List A_i	1	X_{11}	X_{12}	X_{13}	...	X_{1j}
	2	X_{21}	X_{22}	X_{23}	...	X_{2j}
	3	X_{31}	X_{32}	X_{33}	...	X_{3j}

	k	X_{k1}	X_{k2}	X_{k3}	...	X_{kj}

Using Table 4.3 an agent can access the bid agent 3 has made on task 2 by using $X_{A_{i3}, T_{i2}}$ which is equivalent to $X_{3,2}$ but in the former case the agent and task id can equal anything and the winning bid can still be located, assuming agent and task IDs are unique. The difference between this method and the previous method is that the existence of a bid such that $X_{kj} > 0$ can be assumed that agent k is assigned to task j otherwise there would be no bid.

4.2.2 Communication

The CBBA communicated three sets of data to nearby agents, the winning bids list y_i , the winning agent list z_i and the time stamp s_i . The data sent was not indexed and it was assumed that all agents would have the same setup i.e. that agent one was in the first row. In a dynamic environment it can be assumed agents may order their assignment matrix differently therefore the matrix cannot be directly communicated; using the index of a matrix for a task or agent may not equal the task or agent in the same index on another agent. Instead agents will transform the

assignment matrix into a $3 \times u$ matrix where u is the number of assignments in X_i sent to each agent. This matrix takes each individual assignment (those with a value greater than 0) and reforms them as $[j k X_{ikj}]$ where i is the sender who thinks that agent k is assigned to task j with a bid of X_{ikj} . This message is used by the receiver to come to a consensus on the received winning assignments. If the receiving agent m decides the bid sent from the sender agent i is valid it can update the assignment matrix with $X_{mkj} = X_{ikj}$.

Table 4.4: Agent A1's winning bid matrix and the related communication message. Highlighted cells show where the information is coming from for a specific assignment. Although task and agent IDs are in numerical order this is irrelevant for the working system.

Tasks List T_1 for agent A1								
	j	$j+1$	$j+2$	$j+3$	$j+4$	$j+5$	$j+6$	$j+7$
ID	T1	T2	T3	T4	T5	T6	T7	T8

Agent list A_1 for agent A1						
	i	$i+1$	$i+2$	$i+3$	$i+4$	$i+5$
ID	A1	A2	A3	A4	A5	A6

Assignment Matrix X_1 for agent A1									
		Tasks							
		j	$j+1$	$j+2$	$j+3$	$j+4$	$j+5$	$j+6$	$j+7$
Agents	i	99.6			99.6	99.5			
	$i+1$				95.9		95.9	95.9	
	$i+2$	95.7			95.7	94			
	$i+3$		94.6	94.8	97.1		97.1	97	94.7
	$i+4$	94.3			94.2	94.2	93.1	93.9	95.3
	$i+5$		95.4	93.7	92.9		92.9	92.9	90

Assignments for A1		
Task	Agent	Bid
T1	A1	99.6
T1	A3	95.7
T1	A5	94.3
T2	A4	94.6
T2	A6	95.4
T3	A4	94.8
...
T8	A6	90

Table 4.4 shows the process an agent A1 will go through when constructing their communication message for assignments. Each assignment found in the winning assignment matrix is added into the communication message for A1, using the agent list and task list the relevant IDs can be found for the winning assignment $X_{1,i+3,j+1}$ where the agent equal to $A_{1,i+3}$ is assigned to task $T_{1,j+1}$. The complete assignment message is $[T_{1,j+1} A_{1,i+3} X_{1,i+3,j+1}]$.

On initial communication with an agent their ID i and equipment list ae_i is sent to enable agents to calculate when equipment requirements are met for a task or what is still needed. As in the CBBA the time stamp vector s_i is sent once per communication as well as all the assignments in an agents winning assignment list $X_{ikj} \forall k, \forall j$. Using this method the amount of data sent per agent when dealing with multi-agent tasks should reduce, however it is expected to increase when dealing with single-agent tasks.

Table 4.5: Agent A1 able to add newly discovered agents to its knowledge base; a similar method can be used to add newly discovered tasks.

Agent List for A1								
	i	$i+1$	$i+2$	$i+3$	$i+4$	$i+5$...	$i+n$
ID	A1	A2	A3	A4	A5	A6	...	A_n

										Tasks								
										j	$j+1$	$j+2$	$j+3$	$j+4$	$j+5$	$j+6$	$j+7$	
Agents	i	99.6			99.6	99.5												
	$i+1$				95.9		95.9	95.9										
	$i+2$	95.7			95.7	94												
	$i+3$		94.6	94.8	97.1		97.1	97	94.7									
	$i+4$	94.3			94.2	94.2	93.1	93.9	95.3									
	$i+5$		95.4	93.7	92.9		92.9	92.9	90									
	...																	
$i+n$	0	0	0	0	0	0	0	0										

An agent will always store its own assignments in the first row of the assignment matrix and the first column of its agent list. Additional agents it communicates with will be added to both the assignment matrix and the agent list when a first communication message is received as seen in Table 4.5. It is assumed that on first contact agents would exchange relevant information such as their ID and equipment list. This allows agents to build up their own agent list independently of any other agents but in a way that allows them to continue to communicate. When a

winning assignment is received from another agent the winning agent's ID is found in the receiver's agent list and the index can be used to update the agent's bid in the assignment matrix. In the case that the correct agent ID is not found a new agent is added to the agent list and assignment matrix. Therefore agents can continue to consensus on assignments even when they start with no knowledge of an agent and only receive assignments propagated from another agent.

4.3 RESULTS

An increase in the volume of data sent when considering single agent tasks is expected because the data required for a single bid is now larger, this increase will be equal to an extra vector in the worst case scenario. In the worst case scenario every task will have an assignment, the CBBA sends the winning agent list z_i , the winning bid list y_i and the update times s_i . Therefore the total data sent for the final round of bidding will equal to $2N_m + N_n$. Compared to the CBGA which will now send three pieces of data for each assignment $[i j x_{ij}]$ and the update times s_i totalling $3N_m + N_n$.

However when looking at multi-agent tasks there will be a significant reduction in the data transferred and at a worst case scenario send a similar amount of data. With changes made to the data structures on each agent, comparisons can be made between the two different data storage methods. Adapting the original method to multi-agent tasks uses multiple matrices to store assignment data with each entire matrix being communicated. The new method uses the dynamic matrix for each agent with individually sent assignments in the form $[i j x_{ij}]$.

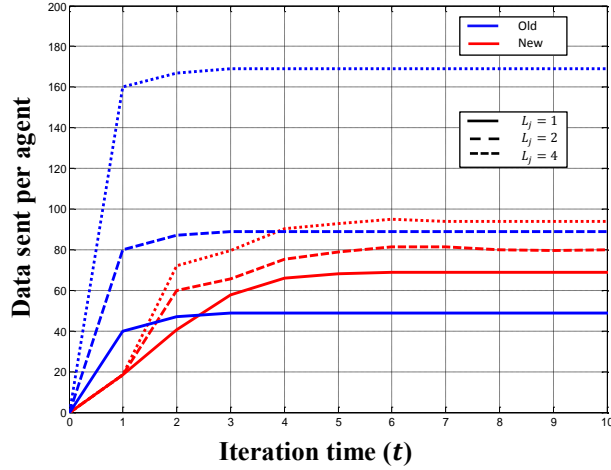


Figure 4.1: The average amount of data sent in progressive steps through each iteration of the CBGA. Simulated experiments contain 10 agents completing 20 tasks requiring L agents per task. Data is calculated as an individual piece of information sent from one agent such as a single winning bid.

Figure 4.1 shows that the new system reduces the amount of data sent for multi-agent tasks. Data sent with the new system gradually increases over the duration of the simulation. The old method involved sending the entire assignment data regardless of whether bids had been made. With the new system redundant data is removed allowing agents to send only the required information. As consensus is achieved with the new system the amount of data sent becomes fixed where data sent becomes determined by the number of assignments required on each task equal to $3(\sum_{j=1}^{N_m} L_j) + N_n$. Similarly in the the worst case scenario where every agent is required for every task the old CBGA required $2N_m N_n + N_n$ where as the new method would require more data at $3N_m N_n + N_n$.

A set of experiments was set up to look at the effects of limited communication and agent knowledge discovery. The distance over which agents could communicate was increased in steps of 0.1 with 100 simulations run for each set and the average data between these simulations recorded. Figure 4.2 - 4.6 shows assignments where communication is limited, additionally each agent starts the

simulation with no knowledge about the other agents, knowledge is added through direct communication between agents and indirect communication where an agent shares its knowledge of other agents and assignments.

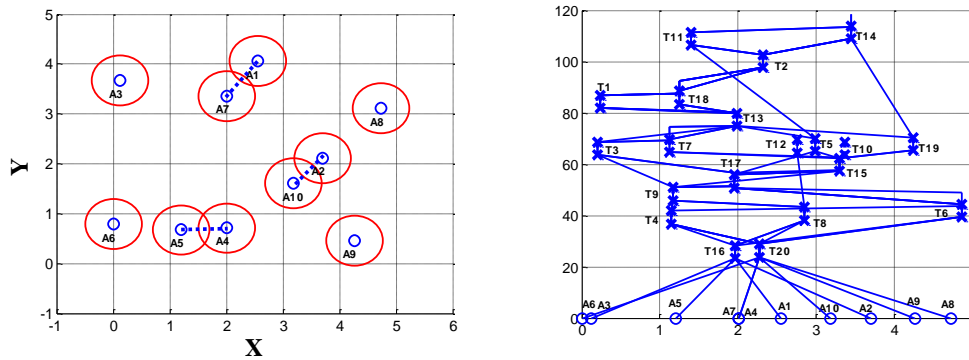


Figure 4.2: Communication range limited to a distance of 1, overlapping circles and connected lines show connecting communication networks. Associated assignments for 10 agents completing 20 single-agent tasks (right).

The resulting assignments from limiting communication to a distance of 1 are shown in Figure 4.2, the communication distances are shown and any overlapping circles show the available communication channels. Without being able to communicate to the entire group many agents assign themselves to the same tasks resulting in conflicting assignments and wasted effort.

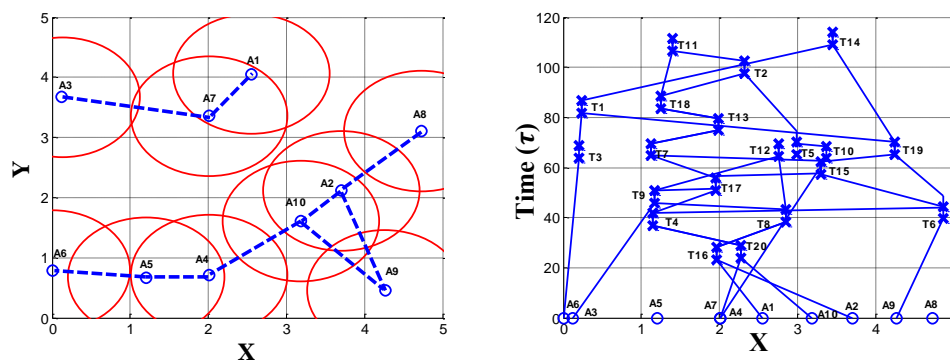


Figure 4.3: Communication range limited to a distance of 2, connected communications shown by connected lines. Associated assignments for 10 agents completing 20 single-agent tasks (right).

By increasing the communication distance to the point where most agents are connected as seen in Figure 4.3, assignments become clearer and conflict is less common with a number of agents not needing to do any assignments. However there still is not a complete connected communication channel where an agent has indirect communication with every other agent. In this case two groups of agents have communication and have solved the assignment as two separate groups.

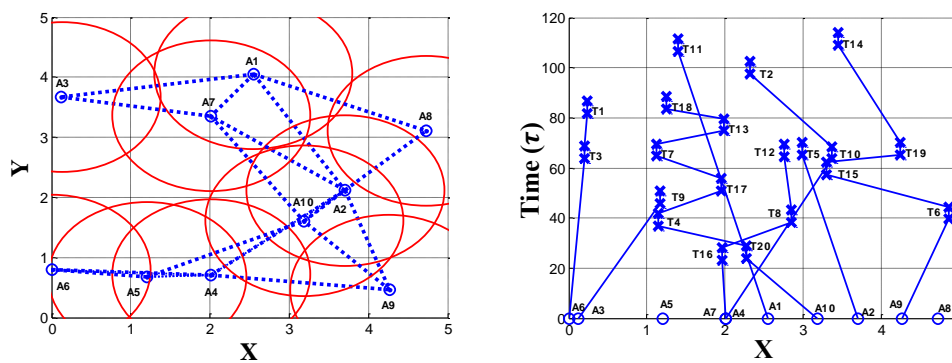


Figure 4.4: Communication range limited to a distance of 2.5, overlapping circles show connecting communication networks, blue link shows the networked agents. Associated assignments for 10 agents completing 20 single-agent tasks (right).

Finally increasing communication such that every agent is connected in the network graph as shown in Figure 4.4 a complete conflict free assignment is now available. Multi-agent tasks function slightly differently; when agents have no communication with other agents they have no way to make any valid assignments and thus will not assign themselves to any tasks. However, for the few groups of agents in reachable communication distance with another will assign themselves as normal with its neighbours. As seen in Figure 4.5 three groups of two agents exist and perform assignment and consensus with each other, although because these sub-groups have no contact with each other they will produce conflicting assignments with the other groups as is seen for individual agents in Figure 4.2.

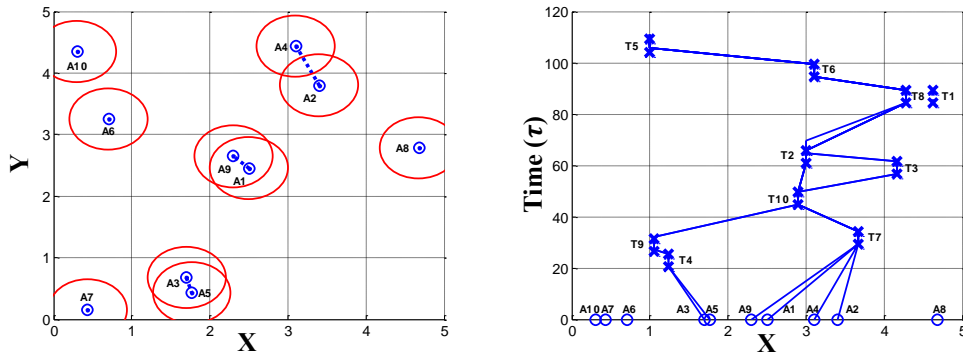


Figure 4.5: Communication range limited to a distance of 1, overlapping circles show connecting communication networks, blue link shows the networked agents (left). Associated assignments for 10 agents completing 10 multi-agent tasks where $L_j = 2$ (right).

Figure 4.6 identifies that initially when a connected communication has not been achieved convergence time is faster than with a full connection but that the score achieved is diminished. This effect is seen because each connected group is smaller than the maximum number of agents in the simulation, they are able to come to a consensus on their knowledge at a faster rate but due to conflicting assignments that they are unaware of the overall score is reduced. As the simulations achieve a complete connected network the score continues to rise but the convergence time is greater than that of global communication. Namely this is caused by the travel distance of data where in a globally connected assignment all agents will bid for a task and often the best bid will succeed and not be replaced, whereas when the communication is only on a connected network the best bid will take longer than one round to propagate to all other agents. Such that the convergence time of a single bid will equal U where U is the maximum number of links between any two agents in a group such that $U \in \{1, \dots, n - 1\}$ for a n agent assignment and when $U = 9$ and $n = 10$ would imply all 10 agents in a row only able to communicate with at most two neighbours and such that the two agents at the end of the links can only communicate with one other agent. In a case where one end agent provides the

highest bid for a task it would take 9 bidding rounds for this bid data to move through the network to the agent at the opposite end.

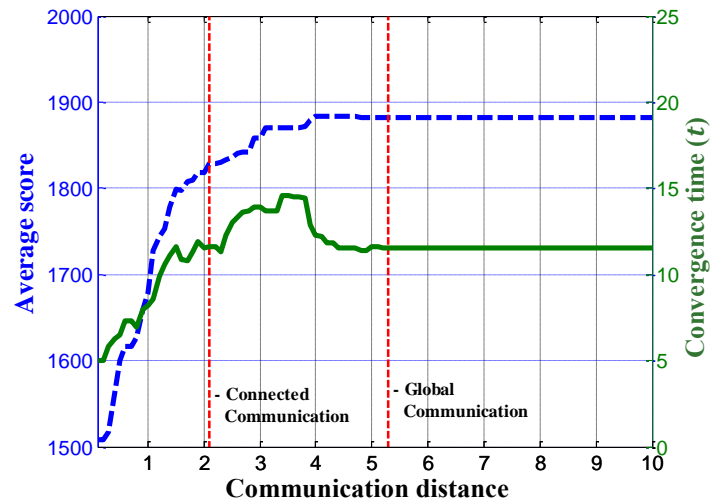


Figure 4.6: Total Score and convergence time for consensus using the CBGA measured in the number of bid rounds required as the distance of communication increases. Markings for the approximate point where all agents are networked and when all agents can globally communicate with any other agent. Experiments use 10 agents completing 20 multi-agent tasks where $L_j = 2$.

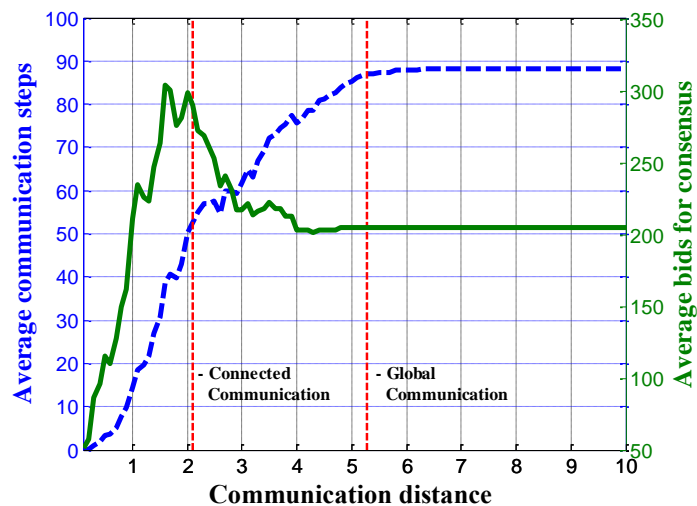


Figure 4.7: The effect on the number of communication steps and bids required in the CBGA for consensus as the distance of communication is increased. Markings for the approximate point where all agents are networked and when all agents can globally communicate with any other agent. Experiments use 10 agents completing 20 multi-agent tasks where $L_j = 2$.

This reasoning is supported in Figure 4.7 where the total required number of bids for consensus is at its highest just before a complete connected network is achieved and gradually drops until a global connection is reached. The number of communication steps continues to rise steadily where communications continue to happen even when no new information is presented. Once global communication is achieved the increase in communication distance has no effect thus all recorded data for each set of experiments is the same.

Using the CBBA as a baseline Figure 4.8 shows the bandwidth requirement for three sets of experiments where it can be seen how the amount of data sent and received increases as the number of agents required L_j is increased. The CBBA (blue line) was run with an increasing number of agents to complete 20 single-agent tasks. Comparisons are made between the bandwidth requirements of multi-agent tasks and that of single-agent tasks. Experiments were run using the CBGA for multi-agent tasks, one set required two agents per task (green line) and another required three agents per task (red line).

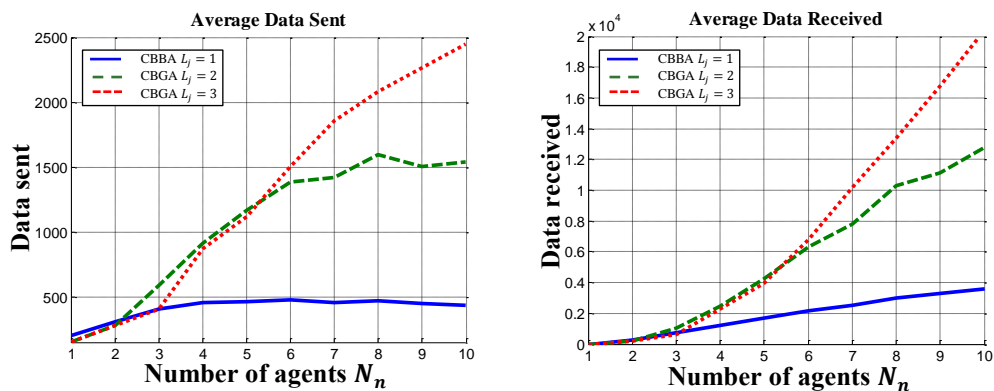


Figure 4.8: Average data sent and received per agent to achieve a consensus for twenty tasks using the CBBA for single-agent tasks and the CBGA for two types of multi-agent tasks requiring two and three agents.

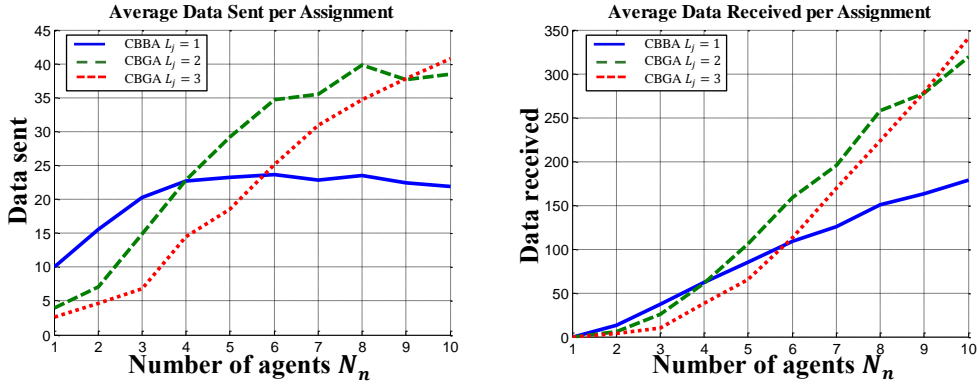


Figure 4.9: Average data sent and received per agent for each assignment where a multi-agent task requiring three agents will require three assignments. Consensus required on twenty tasks with the CBBA for single-agent tasks and the CBGA for two types of multi-agent tasks requiring two and three agents.

Figure 4.8 shows multi-agent tasks using the new communication method requires more bandwidth when compared to single-agent tasks. These increases were expected as each multi-agent task requires more assignments than a single-agent task. With multi-agent tasks more information is needed to reach a consensus over single-agent tasks, thus the amount of data sent and received increases. When allocating agents for twenty multi-agent tasks where $L_j = 2, \forall j \in J$, the number of correct assignments required is double that of single agent tasks. In Figure 4.9 the data is adjusted to show the data sent or received per assignment, where the number of assignments to reach a consensus is equal to $\sum_{j=1}^{N_m} L_j$ for N_m tasks. An increase in bandwidth can still be observed but the difference has been reduced by looking at bandwidth per assignment and in smaller assignments the CBBA has a larger bandwidth requirement per assignment. This difference can be attributed to the number of conflicting bids found during an assignment as seen in Figure 4.10. When the agent requirement on multi-agent tasks is similar to the number of agents in the system fewer conflicting bids are seen. With fewer conflicts assignments are quicker and thus the amount of overall data sent per assignment is reduced. As the number of agents increases relative to the agent requirement an increasing number of conflicts

arise. Overall the CBGA performs reasonably in terms of communication requirements in solving multi-agent tasks given the increased complexity of those tasks over single-agent tasks.

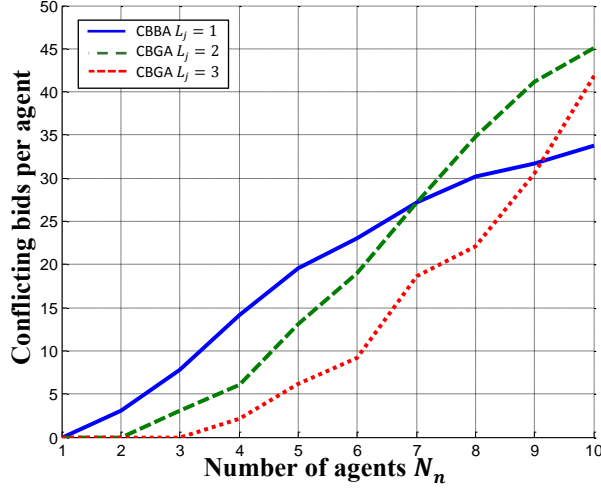


Figure 4.10: The average number of conflicting bids per agent where a conflicting bid is defined as a bid for a task that is replaced by a better bid. Consensus required on twenty tasks with the CBBA for single-agent tasks and the CBGA for two types of multi-agent tasks requiring two and three agents.

The duo cooperative extension of the CBBA splits each task into two separate sub tasks j_1 and j_2 . Both these subtasks must be recorded in the winning agent list z_i and the winning bid list y_i as well as sending the agent update list s_i . These changes introduced by the duo cooperative algorithm almost doubles the amount of data each agent has to send. In the worst case scenario the amount of data sent will be equal to $4N_m + N_n$. However, the CBGAs worst case scenario requires $3(\sum_{j=1}^{N_m} L_j) + N_n$, which simplified is $6N_m + N_n$ when $L_j = 2$. This means in the worst case scenario the CBGA requires more data to be sent to communicate an assignment, but the CBBAs average case is very similar to its worst case where the entire assignment is sent at each iteration of the algorithm. Figure 4.11 shows that the CBGA provides a much lower average case that gradually increases as agents converge on a solution. The

primary cause of this is because not every task is assigned due to agent and time constraints.

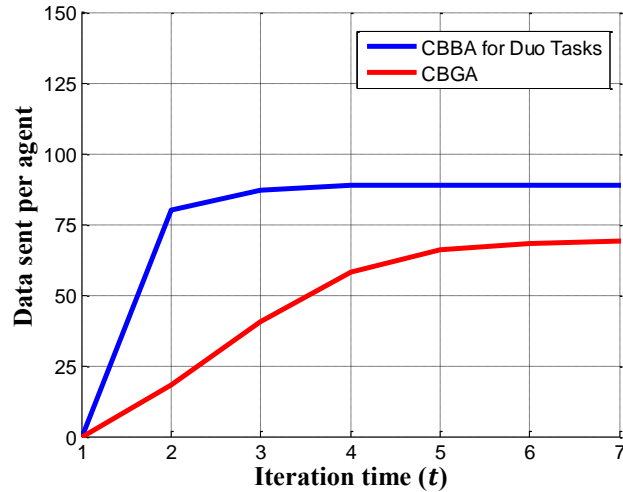


Figure 4.11: Comparison between the CBGA and the CBBA for duo tasks on the amount of data sent per agent to communicate an assignment at progressive iterations of each algorithm. Experiments contain 10 agents and 20 multi-agent tasks requiring 2 agents each.

4.4 CONCLUSIONS

This Chapter provides an efficient method of storing data for the CBGA in order to reduce data when assigning multi-agent tasks in the multi-agent task assignment problem. Experimental data showed that the agents were able to acquire and share knowledge of other agents in the system to the point where, once all agents were connected, convergence to a solution was performable in a similar time to global broadcast and converged on the same solution. Additionally it is shown that the bandwidth increases from multi-agent tasks were as expected compared to single-agent tasks but with the changes made to data storage and the method of communication reduced the amount of communicated data for the CBGA.

Communication increases were expected with an increase in the complexity of tasks that require more information to reach consensus. However, an overall decrease

in communications required for consensus are seen when moving from single-agent to multi-agent tasks; though the actual size of data required for consensus has increased. Further it can be seen how the assignment data is stored more efficiently and for multi-agent tasks less data is sent than the original method in Chapter 4. Adding to the constraints on which agents can perform each task further reduces the set of achievable tasks for any one agent.

Chapter 5: *Biologically Inspired Improvements to the CBGA*

This chapter analyses research on biological cooperation to bring improvements to the multi-agent task assignment problem. The fields of robotics and artificial intelligence are moving increasingly towards biology to develop more sophisticated systems resulting in biologically inspired systems (Hinchey & Sterritt, 2007). Many aspects of biology are used to develop biologically inspired systems because animals have evolved over long periods of time to adapt and survive in their environment. The physiology of animal species can be used to develop new robots for specialised tasks (Bar-Cohen, 2006) such as looking at the movements of snakes to produce similar robots that can navigate challenging environments (Pettersen et al., 2013) or the creation of micro air vehicles based on flying insects for example bees (Wood, 2008). The mental processes of animal behaviours and their social systems can help improve or develop artificial intelligence (Anderson & Donath, 1990) using the flocking behaviour of populations of animals for instance birds and fishes to create complex formation control from simple behaviours (Antonelli et al., 2010). This chapter specifically looks at various animal behaviours in eusocial species that can help improve the cooperative behaviours of agents when assigning and coming to a consensus on multi-agent tasks. Eusociality is the highest level of animal social organisation where focus is on the survival of the group but not necessarily the individual, containing animal species such as ants, bees and termites. Other levels of animal sociality are less useful when applied to robotic and AI in the area of cooperation because whilst many animal species are social and cooperative their evolved behaviours to do this are mostly selfish with their primary goal survival of the individual (Dawkins, 2006). Eusocial species like ants are better suited for cooperative robotic systems because members of the group have the same goal, the survival of the group, which sometimes comes at the expense of the individual.

Similarly with a cooperative group of UAVs the overall goal is to maximise the results the team produces rather than maximising the contribution of an individual in the group. If an agent reduces its own result but in doing so increases the group's result then that is the action it should take.

Multi-agent tasks present a unique set of problems relating to team organisation and cooperation. The CBBA as the basis for this extension is focused entirely on the individual and improving its score which in turn improves the overall team score. In the case of multi-agent tasks a greater individual improvement is not necessarily the best improvement for the team where incomplete team assignments give no reward. Taking inspiration from collective animal behaviours of large groups like ants can be useful in developing algorithms for group decision making. With the inclusion of multi-agent tasks the developed algorithm has a greater focus on the agenda and score of other agents. Using the task quitting method from bee colonies (Johnson, 2009) and the team focused assignments of ant nests (Gordon, 1999) improvements in the cooperative assignment for multi-agent tasks can be compared. The reliance on decisions of other agents adds to the problem. To deal with this challenge, inspiration can be drawn from the cognitive behaviours of eusocial animals using their complex behaviours for group decision making (Plowes, 2010; al-Rifaie et al., 2012).

5.1 PROBLEM

The CBGA developed in chapter 3 provides a framework for the assignment and organisation of multi-agent tasks. However, at times the algorithm can provide unnecessary assignments or produce less than optimal assignments. Although producing the optimal result in a decentralised system is difficult and unfeasible changes can potentially be made to improve the assignments. This chapter will

implement two concepts derived from eusocial animal groups and test for improvements in assignments by implementing the following:

- Team improving assignments
- A method of task quitting

With the implementation of these features comparisons to the CBGA without such features can be made. Any improvements will increase the effectiveness of the CBGA and will prove the usefulness of these adaptations found in eusocial species.

5.1.1 Team Improving Assignments

One of the unique features of a eusocial species is the self-sacrificial nature of the workers in the colony. In an ant colony each worker provides no contribution to the physical reproduction of the nest (Heinze, 1998), whilst worker ants provide function to the colony by completing various tasks the life of a single ant is inevitably unimportant. This leads to the self-sacrificial behaviour by ants when the colony is in danger (Tofilski et al., 2008), the individual value of an ant is not as important as the colony as a whole.

Comparatively in the multi-agent task assignment problem the individual score of a single agent is not as important as the overall score of the team. In the CBBA agents would assign themselves to tasks based on which task would provide the greatest increase in score at a specific point in the agent's path. Similarly when the CBBA was extended into the CBGA agents would continue to choose assignments based on the individual increase in score per task.

$$\begin{aligned}
 c_{ij} &= \max_{n \leq |p_i|} S_i^{p_i \setminus \{n\}} - S_i^{p_i}, \forall j \in J \quad b_i \\
 h_{ij} &= \Pi(c_{ij} > y_{ij}), \forall j \in J \\
 J_i &= \operatorname{argmax}_j c_{ij} \cdot h_{ij}
 \end{aligned} \tag{5.1}$$

Shown in (5.1) is a section of the assignment algorithm from the CBBA that chooses which task should be added to the current bundle. As seen here the primary

decision for which task should be added is the calculation of J_i , which provides the task with the highest individual increase that beats any current bid. Whilst this will provide conflict free assignments it will not always provide the best assignments. For example Figure 5.1 shows a simple situation where individual focused assignments will provide a lower score than team based assignments. In this situation task 2 requires two agents for completion but the base reward is twice that of the single-agent task 1, and with still more when costs are considered. Agent 1 works out its contribution to task 2 as lower than completing task 1 due to the travel distance, but had the calculation assumed the task would be completed then the team would produce a higher overall score at the expense of agent 1's individual contribution to the team score. Additionally the current assignment leaves agent 2 travelling to a task that cannot be completed reducing the team score slightly from travel cost with no completion reward.

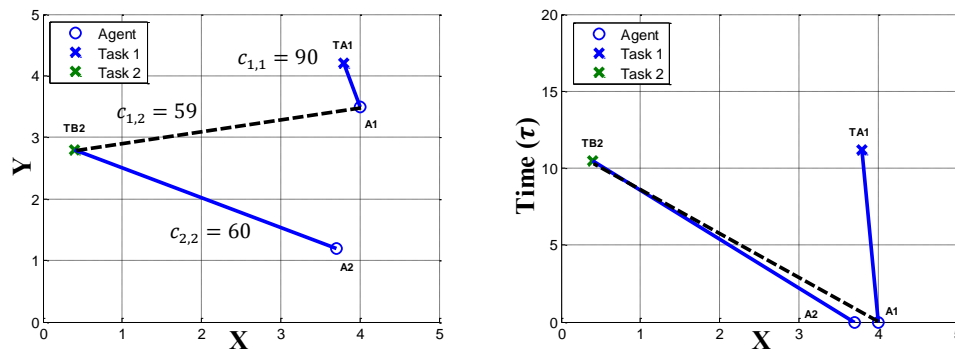


Figure 5.1: An example situation where individual assignment priority will result in a lower score than a team focused assignment. Task 1 is a single-agent task and task 2 is a multi-agent task that requires two agents.

Modification to the assignment algorithm must be made to change the agent focus from maximising individual scores to taking into account the effect of the group score.

5.1.2 Task Quitting

Another eusocial animal behaviour observed in bee hives is that of frequent task quitting helping to distribute assignments to high demand areas. Observation of bee colonies and experimental results showed that by implementing a system where agents quit their current task, agents were able to easily adapt to a changing environment (Johnson, 2009). Whilst the benefits observed are associated with adaption in a dynamic environment there is potential for these benefits to also address the changing demand on multi-agent tasks. Initially in the CBGA all tasks are unassigned and therefore the demand for assistance on each task is equal. After the first round of bids the demand for each task changes, the multi-agent tasks that have assignments but are yet to meet the correct assignments would be considered as in more demand than other tasks. The closer a multi-agent task is to being fully assigned the greater the demand is to complete this task over a single-agent task as the payoff for completing the assignment will be greater. For efficient assignments agents must fulfil the requirements of as many tasks as possible, every agent assigned to a task that is yet to meet its requirements is a wasted resource for the team. In combination with individual focused assignments agents can often assign themselves to a task that is rewarding for the individual but that no other agents can find time or justification to assign themselves to the task, in these situations the agent will be stuck on a task that ultimately will provide no score. The use of task quitting will allow agents to unassign themselves and re-evaluate their choices shifting assignments into higher demand tasks such as those which have more assignments and thus are contributing a greater reward.

5.2 ALGORITHM

The Consensus Based Grouping Algorithm developed in Chapter 3 is an extension of the consensus based bundle algorithm. The algorithm allows agents to

come to a consensus on multi-agent tasks, this section explains the modifications to the algorithm to improve the quality of assignments using the two concepts of task quitting and team based rewards from eusocial species.

5.2.1 Team Improving Assignments

In the CBBA agents assigned themselves to a task j that provides the greatest increase in score at a specific point in the agent's path p_i . Using $S_i^{p_i \emptyset_n \{j\}} - S_i^{p_i}$ an agent could determine its improved contribution to the total score by completing task j in position n of its path p_i . Calculating this improvement for every task would determine the best task to add to the agents bundle b_i as shown in (5.2).

$$c_{ij}(b_i) = \max_{n \leq |p_i|} S_i^{p_i \emptyset_n \{j\}} - S_i^{p_i}, \forall j \in J \quad b_i \quad (5.2)$$

Similarly the CBGA calculates the score improvement using the same function but with additional requirements on calculating viable tasks. Differences between the two algorithms occur when calculating the conflicting winning bids. The CBBA containing only single-agent tasks can only ever have one agent assigned to each task, thus using (2.1) an agent must beat the current highest bid and if the agent is able to do so can potentially assign themselves to that task depending on if that bid is the overall highest out of agent i 's possible task choices. An agent calculates h_{ij} containing a list of binary values for each task, where 1 is for those tasks with bids which are higher than the current bid, or 0 for failed bids.

$$h_{ij} = \Pi(c_{ij} > y_{ij}), \forall j \in J \quad (5.3)$$

The CBGA deviates from this because an agent does not necessarily have to have the highest bid to win the assignment. An agent bidding on a multi-agent task can win in two ways, either the task team is not yet full thus any bid is acceptable or the task team is full in which case the agent simply needs to beat the minimum bid for

the task rather than the highest bid. Replacing the minimum bid in a tasks team creates the most improved marginal score gain and therefore gradually creates the highest scoring team for that task as each lowest bid is replaced with better bids. Figure 5.2 shows the algorithm for determining successful bids in the CBGA.

Algorithm 8: Team focused bids for Agent i

```

1: for  $\forall j \in J$ 
2:   if  $L_j = 1$  then
3:      $h_{ij} = \Pi(c_{ij} > \sum_{k=1}^{N_m} X_{ikj})$ 
4:   else
5:     if  $(L_j > (\sum_{k=1}^{N_m} X_{ikj} > 0))$  then
6:        $h_{ij} = 1$ 
7:     else if  $c_{ij} > \min(X_{ikj}), \forall k \in I$  then
8:        $h_{ij} = 1$ 
9:     else
10:       $h_{ij} = 0$ 
11:    end
12:  end
13: end

```

Figure 5.2: Algorithm for determining valid task list h_{ij} in the CBGA.

Finally both the CBBA and the CBGA after filtering an individual's bids down to the highest winning bids, chose the task with the greatest improvement to add to its bundle using (5.4)

$$J_i = \operatorname{argmax}_j c_{ij} \cdot h_{ij} \quad (5.4)$$

The problem with the assignments in the CBGA is agents assume each task is of equal value, (5.3) must be modified to account for additional value in multi-agent tasks. Potential problems can occur if an agent consistently uses the team focused bidding throughout the assignment algorithm, such as consistently alternating between two assignments or multiple agents replacing each other in a task at alternating bid rounds. These cases of deadlocking can be avoided by only using the total score for

the team as the final deciding factor in which task an agent should choose. Modifying (5.4) where an agent makes its final choice on a bid based on the highest individual improvement into (5.5) where the bids are first added to the current team score for that task. Taking into account any replaced bids and adding the current agent's bid will produce the team score for that bid after this round of bidding, thus the agent can choose the task which provides the highest increase in score for the entire team rather than just the individual. Figure 5.3 shows the choice an agent makes between three tasks, with team focused bidding the agent should choose the team task that requires one more agent.

$$J_i = \operatorname{argmax}_j (\sum X_{ij} \forall i \in I + c_{ij}) \cdot h_{ij} \quad (5.5)$$

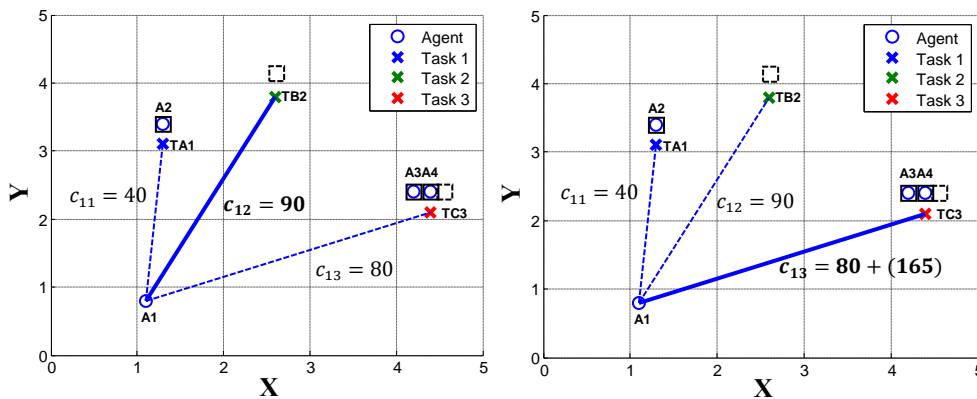


Figure 5.3: Comparison of the bidding decision between individual focused bidding (left) and team focused bidding (right). Agent ‘A1’ must decide between three different tasks, task one requires 1 agent and is already assigned, task 2 requires one agent and task 3 requires 3 agents with 2 assignments already.

This method also gives higher priority to larger tasks that are closer to completion, for example, a task that requires five agents but currently only has four is a more valuable task to finish than a two agent task with only one agent assigned.

5.2.2 Task Quitting

The purpose of a task quitting system is to move resources to higher demand areas, in the case of multi-agent task assignment task quitting will help remove agents from tasks with their requirements not met to assist in tasks that are closer to meeting the requirements. Agents should quit an assigned task if the following criteria are met:

1. The task has unmet requirements
2. The task quitting threshold has been reached
3. The agent has not quit the task before

Each agent will record the current time T of their assignment to a task j in their assignment time vector H_{ij} , this allows agents to track the time passed since their assignment. The task quitting threshold δ is the required amount of time that has expired since an agent was assigned to the task. If the assignment time has passed such that $H_{ij} < T - \delta$ for task j , then agent i will un-assign itself from the task, subsequently it will also release any tasks that follow as is done for outbid tasks because later assignments are dependent on previous tasks in the agent's path. Additionally when new agents are assigned to a team task all the agents assigned should update their recorded assignment time to the current time, this prevents agents deadlocking where agents un-assign themselves at different times, never staying long enough for all the requirements to be met. Synchronising the assignment time of all agents assigned to the task allows the entire group to come to a consensus on when they should quit the task. Finally to preserve the convergence properties of the CBGA agents will only be allowed to quit a specific task once to re-evaluate its assignments. Without this limit situations can occur where there are too many agents for the multi-agent tasks, spare agents could continually assign, quit and re-assign themselves to other tasks that they cannot complete preventing convergence.

As each task is added to an agent's bundle using the bundle construction algorithm in Figure 3.1 an agent will record the time of the assignment using (2.1).

$$H_{ij} = T, j = J_i \quad (5.6)$$

The task quitting algorithm runs when agents clean up their assignments, as agents find out they have lost bids for tasks they will remove the lost task as well as any following tasks as those assignments are only valid for a specific path p_i . Similarly as an agent is removing outbid tasks it will check the assignment times of any unfilled multi-agent tasks using algorithm 3 displayed in Figure 5.4.

Algorithm 9: Task Quitting for Agent i

```

1: for  $\forall j \in J$ 
2:   if  $L_j > 1$  and  $X_{ij} > 0$  and  $Q_{ij} \neq 1$  then
3:     if  $(\sum_m X_{imj} > 0) < L_j$ 
4:       if ( $X_{ij}$  is updated) then
5:          $H_{ij} = T$ 
6:       else if  $(H_{ij} < T - \delta)$  then
7:          $X_{ij} = 0$ 
8:          $Q_{ij} = 1$ 
9:       end
10:    end
11:  end
12: end

```

Figure 5.4: Task quitting algorithm for the CBGA.

Here agent i checks each task which it has made a bid on and is classified as a multi-agent task by use of $L_j > 1$. In addition the agent checks that it has not already quit the specific task using Q_{ij} where Q_{ij} is a binary list of each task equal to 1 when an agent i has already quit task j . For tasks that have incomplete assignments such that $(\sum_m X_{ijm} > 0) < L_j$ make ideal candidates for task quitting. Firstly for potential tasks to quit the agent checks if the task has been updated in the last cycle, any updates will reset the assignment time as progress is being made on completing the

task. Secondly, if $H_{ij} < T - \delta$ is true then the elapsed time since agent i was assigned to task j , is greater than the threshold value of δ and so the agent will un-assign itself from the task. However the value for δ must be determined, a higher value allows more time for other agents to assign themselves to a task but increases the time to come to a consensus, lower values allow for a quicker assignment but could potentially provide little benefit if agents quit a task too early. Once an agent has decided to quit a task assignment it updates its task quitting list $Q_{ij} = 1$ allowing the agent to track which tasks it has already quit allowing the algorithm to continue to converge on a solution.

5.3 PERFORMANCE

5.3.1 Test Scenario

Each test contains a standard 20 tasks and 10 agents, unless otherwise specified, that have randomly generated characteristics such as location and task start times. The seed used to randomly generate each simulation is repeated for each specific set of experiments such that any environment set up for completion using one algorithm is also used when using other algorithms. The objective of each experiment is to maximise the total agent score. The overall score of each experiment is the sum of all rewards for completed tasks minus the cost of agents travelling to each task, where each task has a fixed score reward. Multi-agent tasks defined as requiring more than one agent will reward a score to each agent involved signifying the difficulty and importance of such tasks. Various experiments will be run so that comparisons can be made between the original CBGA and the developed algorithm with the biologically inspired functions (i.e. team improving and task quitting). Each specific experimental setup was run 100 times as in keeping with previous experiments.

5.3.2 Results

In order to analyse the effectiveness of the biologically inspired mechanisms, comparisons are made to the original CBGA as developed in Chapter 3. The resulting score for a simulation is an indication of the assignment quality where completing more assignments and reducing travel distance provides a greater score. Firstly the effect of introducing team focused bidding is tested by increasing the number of agents required for the tasks in each group of experiments. Tasks that require a greater number of agents provide increasingly complex assignments where more agents must come to a consensus on each individual task. As the number of agents required increases so too does the maximum potential score where a single-agent task scores at most 100 and a 4 agent task will score 400.

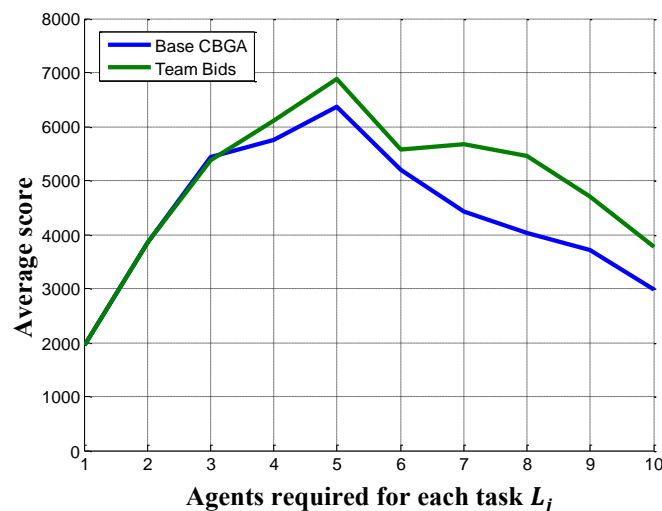


Figure 5.5: Performance of the CBGA algorithm compared to the CBGA using Team focused bidding as the agent requirements (L_j) on tasks increases.

Figure 5.5 shows the results of these simulations. Each point in this graph is the average result of 100 randomly generated simulations in which agent and task locations are randomly created as well as the time window in which each task must be performed. The team focused bidding shows an overall improvement over the base

CBGA once $L_j = 4, \forall j$, where the effect of team focused bidding is more noticeable due the increased cooperative requirements. The team bidding follows a similar pattern with the score dropping off after $L_j = 5, \forall j$. This is most likely due to the reduced number of feasible tasks, although there are 20 tasks there are now only enough agents for one task to be completed at any given time thus the majority of tasks will not be achievable and will produce a lower score despite each task being worth a greater reward. Looking at the spread of results in Figure 5.6 although the team focused bidding can produce a greater average score it is very inconsistent with the larger agent requirements, producing some very high scoring assignments but also producing some assignments with lower scores than the CBGA. This could be caused by agents failing to agree on which task they will assign themselves to, where only one task is completable at a time due to the ratio of agents (N_n) to agent requirements (L_j). As this ratio decreases the margin for error is reduced where there are less agents spare to complete team assignments, for example, when $L_j = 6$ there are 4 agents spare from the 10 to help if the other 6 agents do not agree on which task to complete. This leads to inconsistent assignments where agents refuse to leave their assignment to assist others, which could be solved by the use of task quitting later. As shown in Figure 5.7 agents plan differing tasks and refuse to leave their task to assist, this is caused by how agents build up their individual assignments based on distance to their start location but not necessarily by which task they will complete first. Although even with the most optimal assignment a number of the tasks would still be unfeasible due to the time constraints and travel distances.

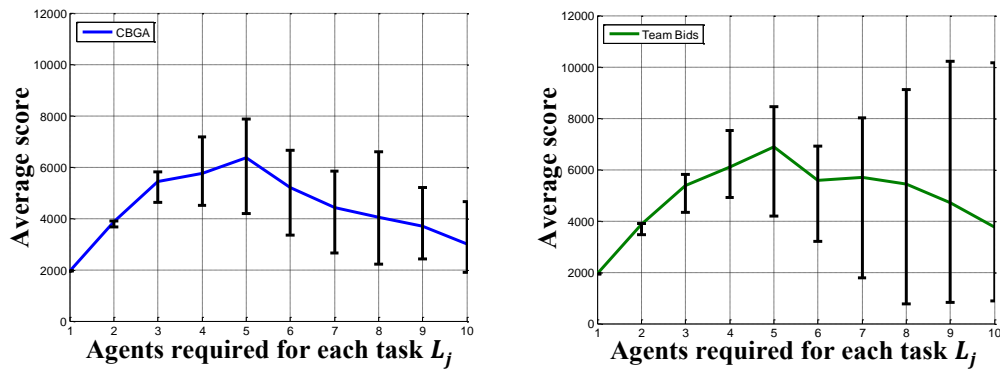


Figure 5.6: Performance with the result spread for the CBGA (left) and the CBGA with team focused bidding (right) with increasing agent requirement (L_j).

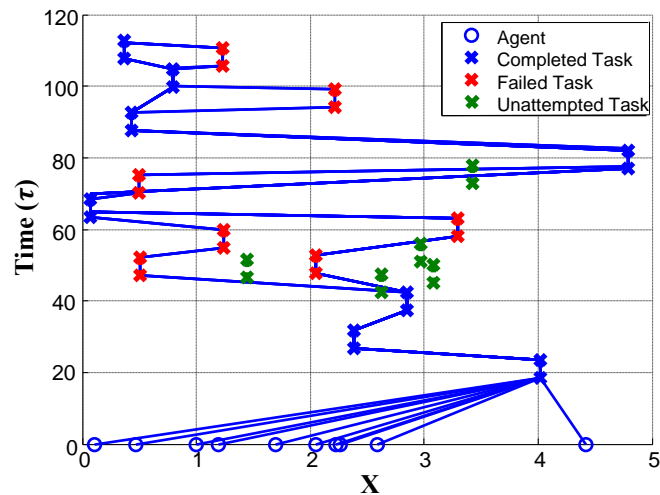


Figure 5.7: Assignment of 10 agents completing 20 tasks resulting in a score of 7295. Each task requires 10 agents to be completed; tasks that do not have all agents assigned are classed as failed tasks.

With the introduction of task quitting to the CBGA Figure 5.8 shows that again this new addition to the algorithm is providing some improvements and similarly to the team focused bidding the improvements are shown after $L_j = 3, \forall j$. This is primarily caused by the ratio of number of agents in the simulation to the required agents per task. When $L_j = 4$ the ratio of agents to the requirement is now below 3 making it harder and more important to assign completed groups, similarly

once the ratio is below 2 a drop in score is observed after $L_j = 5$. This reasoning is supported by the second graph in Figure 5.8, firstly, when the number of agents in the simulation is reduced to 8 the effect of task quitting is observed earlier with a very slight improvement at $L_j = 3$ where again the ratio of agents to required agents is below 3. Secondly, after $L_j > \frac{N_n}{2}$ a drop is observed in the score of both algorithms but the task quitting continues to provide marginally better scores.

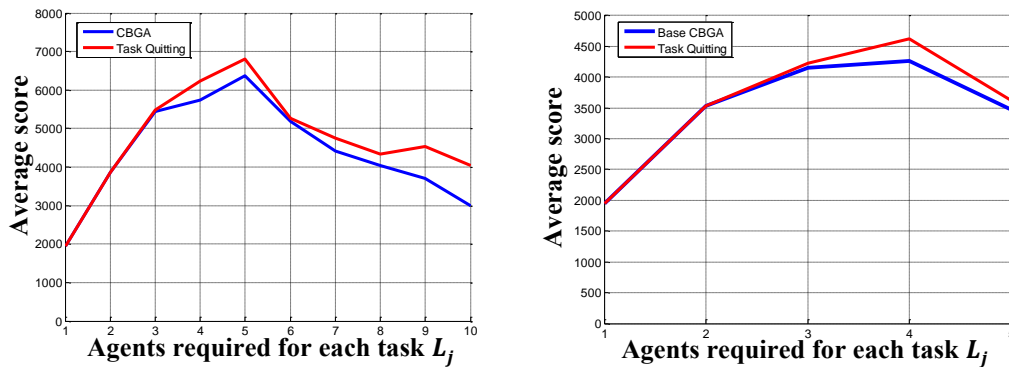


Figure 5.8: Performance of the CBGA algorithm compared to the CBGA using task quitting where the number of agents is set to 10 (left) and when the number of agents is set to 8 (right) completing 20 multi-agent tasks.

The distribution of results is more consistent with task quitting than with the team focused bidding but the spread of results are still greater than the original CBGA as seen in Figure 5.9. As with team focused bidding the algorithm provides the potential for high scoring assignments but in most cases produces scores worse than the original CBGA. Task quitting redistributes resources to higher demand areas, in this case the measurement for demand changes so that upon quitting a task the previous decision may no longer be valid. In comparison to CBGA the value of each task rarely changes resulting in agents reassigning to the same task. Task quitting works when an agent quits a task that was added to its bundle early on in the assignment, once it quits the task other better options might be available, but this situation does not happen often as can be seen by the small improvement generated by task quitting.

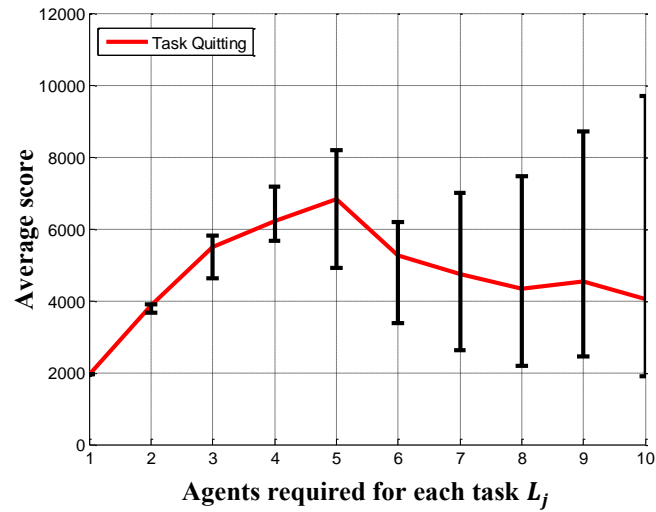


Figure 5.9: Performance of the CBGA using task quitting with the spread of results as the agent requirements (L_j) on tasks increases. Task quitting threshold δ set to 1.

The task quitting algorithm uses a threshold value δ to determine how long an agent should wait before un-assigning itself from a task. Figure 5.10 displays the effect on assignment score by modifying the threshold value for differing communication distances. By reducing the distance over which agents can communicate increases the time it takes for an assignment to propagate to every agent in the simulation. This delay could potentially improve the performance of agents waiting longer before removing incomplete assignments; however the results in Figure 5.10 show this not to be the case. Across the board the changes in score due to the threshold value is minimal with a value of $\delta = 1$ producing slightly higher assignments on average. Lower scores from communication limits are due to some agents becoming separated from the main network of agents and thus not being able to cooperate with the main group. The results here suggest that task quitting can provide some increases in the score of assignments and that it is not important when an agent quits an incomplete task just that they do it at some point to allow better assignments to high demand tasks.

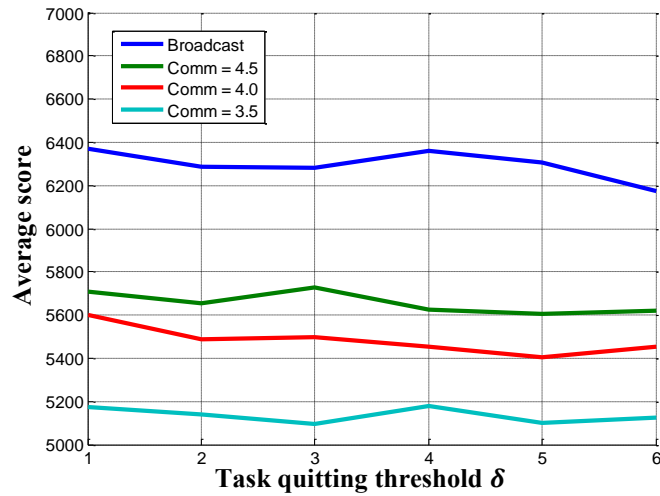


Figure 5.10: Performance of the task quitting algorithm as the task quitting threshold δ is modified. Variation in communication distance to show any changes when consensus takes longer due to network spread.

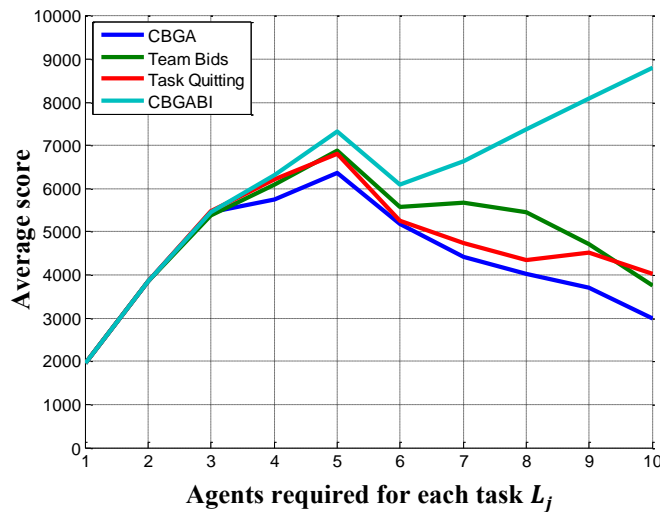


Figure 5.11: Performance of all variations of the CBGA with task quitting and team focused bidding, each simulation contains 10 agents and 20 tasks.

Both the task quitting and team focused bidding show improvements over the CBGA but both provide high distribution of scores and fall into similar problems as the CBGA when assigning and coming to a consensus on high agent requirements. Task quitting requires a variable measurement of demand and team focused bids requires a way to take advantage of the changing information as a simulation goes on.

Combining the biologically inspired improvements with the CBGA (CBGABI) covers the short falls of both the improvements, task quitting alone is not a great improvement if agents does not recognise the value of completing tasks with assignments already. Further recognising the benefit of completing tasks with current assignments is unnecessary if agents cannot leave their current assignments. Figure 5.11 displays the results of using both algorithms compared to the other three approaches, as before similar results are displayed with slight improvements as cooperative requirements are increased. Although a drop off in score is observed after $L_j = 5$ similar to the other variations this resulting drop is due to the number of agents and the requirement on the tasks. When $L_j = 5$ the group of ten agents can form into two teams of five completing more tasks overall than when $L_j = 6$. Although tasks that require more agents provide a greater reward significantly fewer tasks are completed and that does not make up for the loss in quantity of tasks. As well as the improvement in score the use of both algorithms provides less variance in results displayed in Figure 5.12, although still a greater variation than the base CBGA.

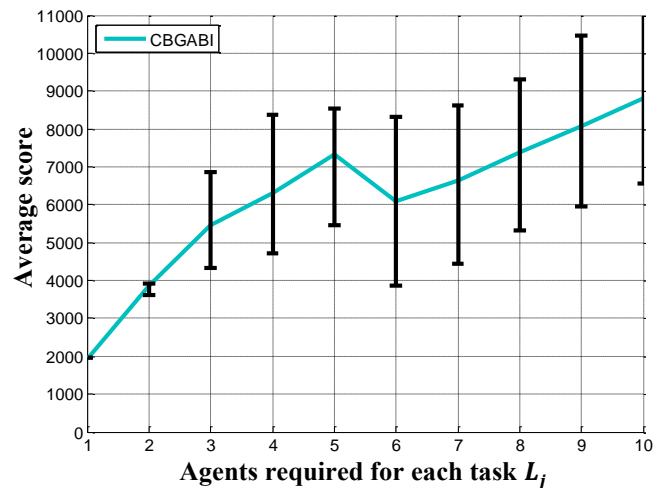


Figure 5.12: Variance of results from assignments using the CBGA with both task quitting and team focused bidding.

Running the exact simulation as displayed in Figure 5.7 but now using both task quitting and team focused bidding shows a near optimal assignment in Figure 5.13. Agents group up and stick together for the duration of the simulation as every agent is required to complete tasks. Previously agents would build up differing bundles based on their starting location and be unable to change their assignment. With the addition of task quitting agents can now build up an assignment as display in Figure 5.7 but quit the incomplete tasks and re-evaluate their task choice such that all the agents eventually decide upon an agreed conflict free solution.

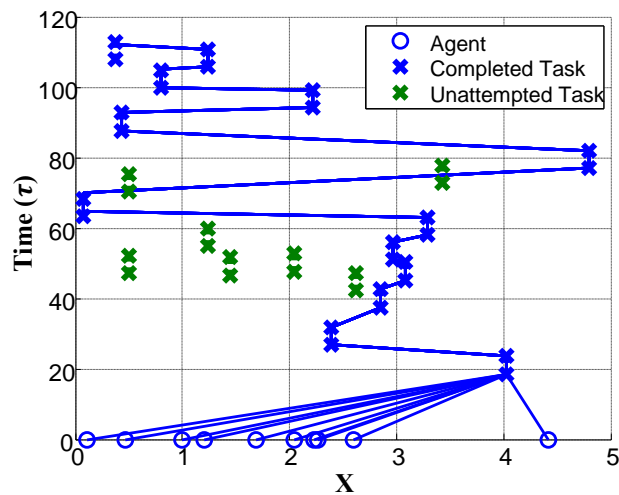


Figure 5.13: Assignment of 10 agents completing 20 tasks that require all 10 agents each the simulation setup is exactly the same as in Figure 5.7 but using task quitting to produce a score of 9391.

Considering again the sequential greedy algorithm presented in Figure 3.16 the CBGABI now scores slightly better than the SGAMA for a low number of agents but in general performs similarly. Figure 5.14 shows the average score for a group of agents completing a set of multi-agent tasks where $L_j = 2, \forall j \in J$ as the number of agents or tasks are changed.

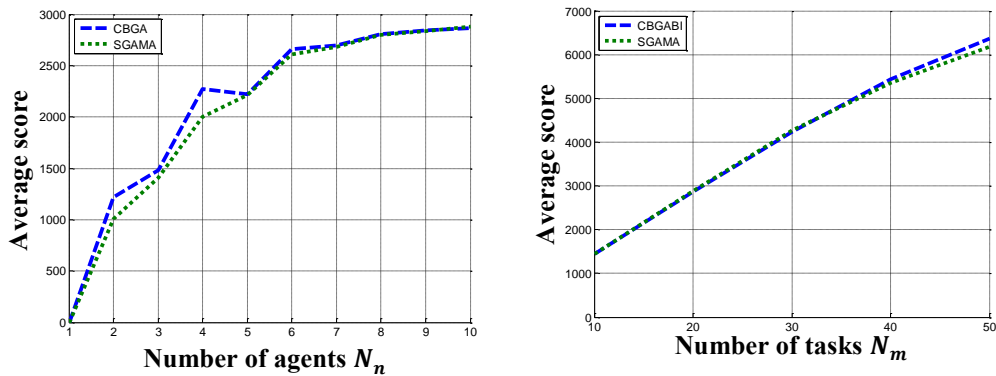


Figure 5.14: Average assignment score achieved by the CBGA with biologically inspired improvements (CBGABI) and the SGAMA completing 20 tasks with N_n agents (left) and 20 agents completing N_m tasks.

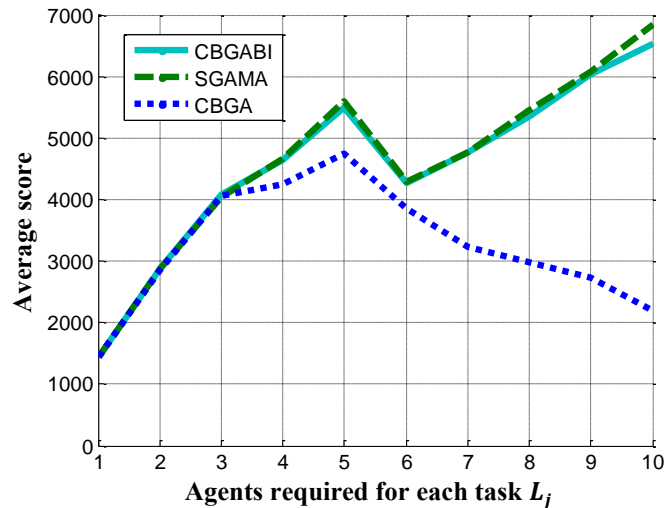


Figure 5.15: Average assignment score achieved by the CBGABI and the SGAMA when agent requirements are increased with 10 agents and 20 tasks.

With the biologically inspired improvements to the CBGA Figure 5.15 shows the previous problem observed in the CBGA assignment for a low ratio of agent requirement to agent numbers has been solved. Where previously the CBGA would assign agents to large multi-agent tasks but have no way to coordinate or redistribute all the agents to one path when the simulation would naturally limit the solution to one path due to $L_j > \frac{N_n}{2}$ meaning only one task can be completed at any given time.

The CBGABI now provides a similar score to the centralised solution SGAMA but in a decentralised system.

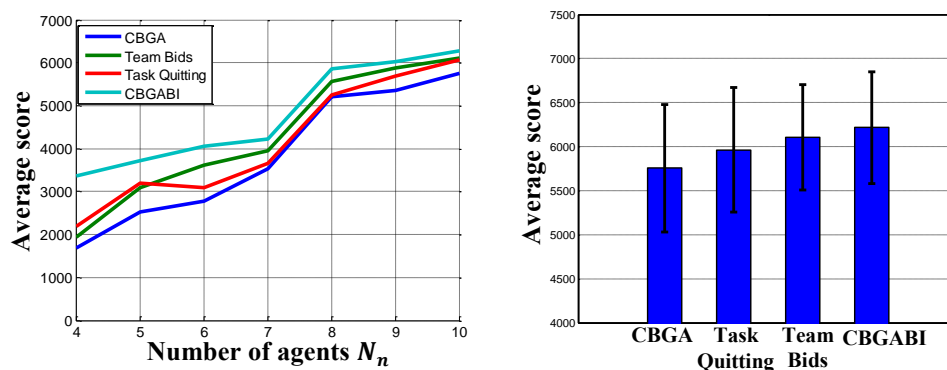


Figure 5.16: The effects of adding task quitting and team bidding to the CBGA for multi-agent tasks. Experiments used a varying number of agents completing 20 tasks where each task required 4 agents for successful completion. A bar plot displays the results and variation when $N_n = 10$ (right).

When addressing multi-agent tasks using an algorithm that focuses on individual improvement, additional agent incentive is required to increase the effectiveness of multi-agent assignments. The overall scores generated by the variations of the CBGA with biologically inspired improvements can be seen in Figure 5.16. Using either task quitting or team rewards produced more complete assignments which in turn provided a higher score in 78% and 79% of cases respectively over the CBGA. Implementing task quitting on its own provides an average increase of 206 over the CBGA with an average score of 5962 ± 708 . Another improvement of 144 can be achieved by assigning with respect to the team rewards over task quitting producing an average score of 6106 ± 601 but this improvement is only significant to $p < 0.15$. Further improvements are gained from using both functions where the CBGABI improved the score of the CBGA in 99% of cases seen in Figure 5.16. Providing an increase in the average score of the CBGA from 5756 ± 722 to a mean score of 6216 ± 634 with a statistical significance to $p <$

0.01. The biologically inspired improvements show a statistically significant improvement over the original CBGA developed in Chapter 3.

5.4 CONCLUSIONS

This chapter presented improvements to the CBGA based on biologically inspired animal behaviours. Aspects from the biological social structures in bees and ants were used to improve team focused consensus in multi-agent assignments. Using bee inspired task quitting (Johnson, 2009) agents can re-assign themselves to tasks which are more demanding by removing failed team assignments where requirements are not met. Results shown in Figure 5.8 that on average some small improvements are achieved using a method of task quitting but that on its own it creates additional problems and introduces a large variation in assignments. Using the method of task quitting in combination with team focused bidding as inspired by cooperative behaviours in ant colonies (Tofilski et al., 2008) (Anderson & Ratnieks, 1999) (Gordon, 1996) proves to provide consistent average score improvement over the CBGA. In particular it provides substantial improvements as seen in Figure 5.11 when there is a high cooperative demand over a majority of the agents in the simulation. These improvements solve a problem found with the original CBGA and bring the assignment quality up to a level equal to a centralised solution in the sequential greedy algorithm. Statistical results show that using these biologically inspired functions created statistically significant improvements to the multi-agent assignment problem. As well as improving the assignment quality of the CBGA these results also prove the usefulness of frequent task quitting in distributing agents to high demand areas in assignment problems.

Chapter 6: *Conclusions and Future Work*

6.1 CONCLUSIONS

The main purpose of this thesis has been to create a framework that provides a reasonable, robust and dynamic solution to the multi-agent task allocation problem. The CBGA was developed as an extension of the CBBA using a newly developed consensus algorithm for handling multi-agent tasks whilst still providing a conflict free solution. Further to this the research focused on creating a system that was relatable to real world problems such as the data structure of assignments between non-standard agents. This chapter outlines the contributions of this work as well as the limitations and problems with the current implementation. Finally the chapter ends with possible directions of further work to improve and extend the research.

6.1.1 Key Contributions

This thesis proposed the CBGA delivers a conflict free feasible solution to the multi-agent task assignment problem. Unlike previous implementations the CBGA is not limited to a specific number of agent assignments and provides a solution that can scale with the requirements. Results show that despite the increased complexity of the tasks a conflict free consensus can be reached within similar times as the single-agent task solutions. Furthermore the algorithm can compete with a centralised greedy solution.

Additionally this research has improved the data structure of the consensus based algorithms with a focus on the real world application. The CBGA has structured its data and assignment messages to reduce the cost on the communication channel. Furthermore each agent can store its knowledge of assignments in a different manner to its neighbours, but still allowing the group of agents to understand and come to a consensus on each other's information.

Taking biological inspiration from the task quitting behaviours found in bees this research integrated these mechanisms into the CBGA to improve the quality of assignments. Experiments demonstrated the usefulness of a quitting mechanism in the redistribution of agents to higher priority tasks. In addition this work also shows how the concept of task quitting can be used to shift the allocation of resources to high demand areas.

6.1.2 Limitations

As with most research the solutions provided in this thesis are not without some limitations and problems. The CBGA provides a conflict free solution to the multi-agent task assignment problem but it is not guaranteed to find an optimal solution. Although in a problem such as this optimality is not a necessity it does leave room for improvement resulting in better assignments. Additionally the equipment limits on tasks were fixed to a specific number of agents, in reality a task does not require a specific number of agents but rather a minimum number that have the required equipment to complete the task.

6.2 FUTURE WORK

The area of cooperative UAS is becoming progressively larger with huge applications for the use of UAVs in many fields. The current practical applications for UAVs are generally limited to single UAVs or small teams with limited cooperativeness between each other. This research has reached out further into the future to a time where multiple UAVs have a greater ability to cooperate together and created a foundation for the autonomous organisation of these UAVs. Although the practical application of this research is limited for the time being this is most certainly the direction in which UAS will head towards.

The CBGA provides a framework for multi-agent task allocation with which a conflict free solution can be produced; however, the algorithm has areas that can be further improved. Equipment dependencies were directly linked to the number of agent requirements such that the total amount of equipment needed for a task was the amount of agents required. An extension of this issue would be to remove the link between the number of agents and the required equipment. Such that the required number of agents varies and an agent can potentially bring multiple pieces of equipment to the task or for a particularly well equipped agent it could satisfy the task requirements on its own thus reducing the amount of resources allocated to the task and improving the overall score for the assignment. Additionally the algorithm used is a pre-planned assignment where assignments are decided on at the beginning of the simulation and conflicts dealt with then, whilst this will work in real time its ability to deal with dynamically changing assignments is restricted. If new tasks or agents are added part way through the assignment the only solution for re-addressing assignments is to run the assignment algorithm again with the new data, it seems reasonable that there should be a way to assimilate the new tasks and agents into a simulation without having to redo the entire assignment again. A system for simple task dependencies was added but in areas of AI planning sometimes a pre-condition that is satisfied can be removed before the follow up task is completed. In these situations on-the-fly adaptation would be required to reassess any pre-conditions and the validity of a current assignment. Finally one specific situation that can occur where the optimal solution is rarely chosen in its simplest form involves an agent that can accomplish two tasks. The first task provides a higher score but can be completed by other agents for a lower score; the second provides a lower score but cannot be reached by any other agent. In this situation both the CBBA and CBGA would fail to match the assignments optimally, agents would need to select their second best choice for the benefit of the team. This situation is easily solved in a centralised system but a

decentralised solution requires more information than is currently sent. A possible solution would involve agents communicating their potential bids for each task, in this way agents can determine which tasks no other agents can reach and thus determine if there is an improved score from assigning itself to that task.

The algorithms proposed in this thesis were tested in simulations and their performance confirmed by the numerous simulation experiments and results. As with all simulations assumptions are made and conditions are simplified to a reasonable extent, practical implementations are required on real hardware to test the validity of the algorithm in a real environment as with any complex problem.

As the field of UAS continues to grow and the practical applications for them extend, this research is necessary for the increasing cooperative behaviours required for robotics in real environments. UAVs are beginning to make their way into mainstream applications and their usage across the globe is increasing as restrictions lower and the problems surrounding their use are solved. The abilities and applications of individual UAVs are significant and constantly improving, but as with many problems cooperation can expand the possibilities even further. This research helps provide a framework for the future use of cooperative autonomous UAVs.

Publications

Hunt, S., Meng, Q., & Hinde, C. J. (2012, January). An extension of the consensus-based bundle algorithm for group dependant tasks with equipment dependencies. In *Neural Information Processing* (pp. 518-527). Springer Berlin Heidelberg.

Hunt, S., Meng, Q., & Hinde, C. J. (2012, December). An Extension of the Consensus-Based Bundle Algorithm for Multi-agent Tasks with Task Based Requirements. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on* (Vol. 2, pp. 451-456). IEEE.

Hunt, S., Meng, Q., Hinde, C., & Huang, T. (2014, April). A Consensus-Based Grouping Algorithm for Multi-agent Cooperative Task Allocation with Complex Requirements. *Cognitive Computation*, 1-13. Springer US.

References

- Alighanbari, M., 2004. *Task assignment algorithms for teams of UAVs in dynamic environments*. PhD Thesis. Massachusetts Institute of Technology.
- al-Rifaie, M.M., Bishop, J.M. & Caines, S., 2012. Creativity and autonomy in swarm intelligence systems. *Cognitive Computation*, 4(3), pp.320-31.
- Amgoud, L., 2005. Towards a formal model for task allocation via coalition formation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems.*, 2005.
- Anderson, T.L. & Donath, M., 1990. Animal behavior as a paradigm for developing robot autonomy. *Robotics and Autonomous Systems*, 6(1), pp.145-68.
- Anderson, C. & Ratnieks, F.L., 1999. Task partitioning in insect societies. I. Effect of colony size on queueing delay and colony ergonomic efficiency. *The American Naturalist*, 154(5), pp.521-35.
- Andersson, A., Tenhunen, M. & Ygge, F., 2000. Integer Programming for Combinatorial Auction Winner Determination. In *Fourth International Conference on MultiAgent Systems.*, 2000. IEEE.
- Antonelli, G., Arrichiello, F. & Chiaverini, S., 2010. Flocking for multi-robot systems via the Null-Space-based Behavioral control. *Swarm Intelligence*, 4(1), pp.37-56.
- Antsaklis, P.J., Passino, K.M. & Wang, S.J., 1989. Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues. *Journal of Intelligent and Robotic Systems*, 1(4), pp.315-42.
- Arathi, H.S. & Spivak, M., 2001. Influence of colony genotypic composition on the performance of hygienic behaviour in the honeybee, *Apis mellifera*. *Animal Behaviour*, 62(1), pp.57 – 66.
- Argyle, M., Casbeer, D.W. & Beard, R., 2011. A multi-team extension of the consensus-based bundle algorithm. In *American Control Conference (ACC), IEEE.*, 2011.

- Axelrod, R. & Hamilton, W.D., 1981. The evolution of cooperation. *Science*, 211(4489), pp.1390-96.
- Bar-Cohen, Y., 2006. Biomimetics—using nature to inspire human innovation. *Bioinspiration & Biomimetics*, 1(1), p.1.
- Beard, R. & Stepanyan, V., 2003. Information consensus in distributed multiple vehicle coordinated control. In *Proceedings. 42nd IEEE Conference on Decision and Control.*, 2003.
- Bellingham, J.S., Tillerson, M.J., Richards, A.G. & How, J.P., 2001. Multi-Task Assignment and Path Planning for Cooperating UAVs. In *Cooperative Control: Models, Applications and Algorithms*. Springer US. pp.23-41.
- Berhault, M. et al., 2003. Robot Exploration with Combinatorial Auctions. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, 2003.
- Bernard, M., Kondak, K., Maza, I. & Ollero, A., 2011. Autonomous transportation and deployment with aerial robots for search and rescue missions. *Journal of Field Robotics*, 28(6), pp. 914-931.
- Bertuccelli, L.F., Choi, H.L., Cho, P. & How, J.P., 2009. Real-time multi-UAV task assignment in dynamic and uncertain environments. In *AIAA Guidance, Navigation, and Control Conference.*, 2009.
- Biesmeijer, J.C. & de Vries, H., 2001. Exploration and exploitation of food sources by social insect colonies: a revision of the scout-recruit concept. *Behavioral Ecology and Sociobiology*, 49(2-3), pp.89-99.
- Burkard, R.E., Dell'Amico, M. & Martello, S., 2009. *Assignment problems*. Society for Industrial and Applied Mathematics.
- Campoy, P. et al., 2009. Computer vision onboard UAVs for civilian tasks. In *Unmanned Aircraft Systems.*, 2009. Springer Netherlands.
- Chao, H., Cao, Y. & Chen, Y., 2010. Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, 8(1), pp.36-44.

- Chen, Y.M. & Wu, W.Y., 2012. Cooperative Electronic Attack for Groups of Unmanned Air Vehicles based on Multi-Agent Simulation and Evaluation. *International Journal of Computer Science Issues*, 9(2), pp.1-6.
- Choi, H., Brunet, L. & How, J.P., 2009. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics*, 25(4), pp.912-26.
- Choi, H.-L., Whitten, A.K. & P, J., 2010. Decentralized task allocation for heterogeneous teams with cooperation constraints. In *American Control Conference (ACC)*., 2010.
- Claes, R. & Holvoet, T., 2011. Weighing Communication Overhead Against Travel Time Reduction in Advanced Traffic Information Systems. *Progress in Artificial Intelligence*, 1(2), pp.165-72.
- Connelly, J., Hong, W.S., Mahoney, R.B.J. & Sparrow, D.A., 2006. Current challenges in autonomous vehicle development. In Gerhart, G.R., Shoemaker, C.M. & Gage, D.W., eds. *Unmanned Systems Technology VIII*., May 2006. Proceedings of the SPIE.
- Conradt, L. & Roper, T.J., 2005. Consensus decision making in animals. *Trends in Ecology & Evolution*, 20(8), pp.449-56.
- Cox, M.D. & Myerscough, M.R., 2003. A flexible model of foraging by a honey bee colony: the effects of individual behaviour on foraging success. *Journal of theoretical biology*, 223(2), pp.179-97.
- Cruzen, C. & Thompson., J.T., 2013. Advancing Autonomous Operations Technologies for NASA Missions. In *IEEE Aerospace Conference*., 2013.
- Dalamagkidis, K., Valavanis, K.P. & Pieggl, L.A., 2012. *On integrating unmanned aircraft systems into the national airspace system: issues, challenges, operational restrictions, certification, and recommendations*. 2nd ed. Springer.
- Dawkins, R., 2006. *The selfish gene*. Oxford University Press.
- De Vries, S. & Vohra, R., 2003. Combinatorial Auctions: A survey. *INFORMS Journal on computing*, 15(3), pp.284-309.

- Deneubourg, J.L. et al., 1991. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats.*, 1991.
- Detrain, C. & Deneubourg, J.L., 2006. Self-organized structures in a superorganism: do ants “behave” like molecules? *Physics of Life Reviews*, 3(3), pp.162-87.
- Dewi, J., 2005. Power Line Inspection - a UAV concept. In *The IEE Forum on Autonomous Systems.*, 2005.
- Di Paola, D., Naso, D. & Turchiano, B., 2011. Consensus-based robust decentralized task assignment for heterogeneous robot networks. In *American Control Conference (ACC).*, 2011.
- Fahlstrom, P. & Gleason, T., 2012. *Introduction to UAV systems*. John Wiley & Sons.
- Farinelli, A., Iocchi, L. & Nardi, D., 2004. Multi-Robot Systems: A Classification Focused on Coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(5), pp.2015-28.
- Fierro, R., Song, P., Das, A. & Kumar, V., 2002. Cooperative control of Robot Formations. In *Cooperative Control and Optimization*. Springer US. pp.73-93.
- Garratt, M.A., Pota, H.R., Lambert, A. & Maslin, S.E., 2007. Systems for automated launch and recovery of an unmanned aerial vehicle from ships at sea. In *Proceedings of the 22nd International UAV Systems Conference.*, 2007.
- Gogarty, B. & Hagger., M., 2008. The Laws of Man over Vehicles Unmanned: The Legal Response to Robotic Revolution on Sea, Land and Air. *Journal of Law & Information Science*, 19, pp.73-145.
- Goldberg, D.E. & Holland, J.H., 1988. Genetic algorithms and machine learning. *Machine learning*, 3(2), pp.95-99.
- Gordon, D.M., 1996. The Organization of Work in Social Insect Colonies. *Nature*, 380(6570), pp.121-24.

- Gordon, D.M., 1999. Interaction patterns and task allocation in ant colonies. In *Information Processing in Social Insects*. Birkhäuser Basel. pp.51-67.
- Hatano, Y. & Mesbahi, M., 2004. Agreement over Random Networks. *IEEE Transactions on Automatic Control*, 50(11), pp.1867-72.
- Heinze, J., 1998. Intercastes, intermorphs, and ergatoid queens: who is who in ant reproduction? *Insectes Sociaux*, 45(2), pp.113-24.
- Hinchey, M. & Sterritt, R., 2007. 99%(Biological) Inspiration. In *Proceedings of the 4th IEEE International Workshop on Engineering of Autonomic and Autonomous Systems.*, 2007.
- Hirschfeld, R.A., Aghazadeh, F. & Chapleski., R.C., 1993. Survey of robot safety in industry. *International Journal of Human Factors in Manufacturing*, 3(4), pp.369-79.
- Hirsch, M.J., Ortiz-Peña, H.J. & Eck, C., 2012. Cooperative Tracking of Multiple Targets by a Team of Autonomous UAVs. *International Journal of Operations Research and Information Systems (IJORIS)*, 3(1), pp.53-73.
- Hirsch, M.J., Ortiz-Pena, H. & Sudit., M., 2011. Decentralized Cooperative Urban Tracking of Multiple Ground Targets by a team of Autonomous UAVs. In *Proceedings of the 14th International Conference on Information Fusion.*, 2011.
- Hoeing, M., Dasgupta, P., Petrov, P. & O'Hara, S., 2007. Auction-based multi-robot task allocation in comstar. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems.*, 2007.
- Huang, H.M., Pavek, K., Albus, J. & Messina, E., 2005. Autonomy Levels for Unmanned Systems Framework: an update. In *Proceedings of the 2005 SPIE Defense and Security Symposium.*, 2005.
- Huang, H.M. et al., 2005. A framework for autonomy levels for unmanned systems (ALFUS). In *Proceedings of the AUVSI's Unmanned Systems North America.*, 2005.
- Innocenti, M., Pollini, L. & Bracci, A., 2010. Cooperative Path planning and Task Assignment for Unmanned Air Vehicles. *Journal of Aerospace Engineering*, 224(2), pp.121-31.

- Jiang, W., Wenkai, F. & Qianru., L., 2013. An integrated measure and location method based on airborne 2D laser scanning sensor for UAV's power line inspection. In *Fifth International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*., 2013.
- Jin, Y., Minai, A.A. & Polycarpou, M.M., 2003. Cooperative real-time search and task allocation in UAV teams. In *Decision and Control*., 2003.
- Johnson, B.R., 2009. A Self-Organizing Model for Task Allocation via Frequent Task Quitting and Random Walks in the Honeybee. *The American Naturalist*, 174(4), pp.537-47.
- Kontogiannis, S.G. & Ekaterinaris, J.A., 2013. Design, performance evaluation and optimization of a UAV. *Aerospace Science and Technology*, 29(1), pp.339–50.
- Kreps, S. & Kaag, J., 2012. The Use of Unmanned Aerial Vehicles in Contemporary Conflict: A Legal and Ethical Analysis. *Polity*, 44(2), pp.260-85.
- Larrauri, J.I., Sorrosal, G. & Gonzalez., M., 2013. Automatic system for overhead power line inspection using an Unmanned Aerial Vehicle — RELIFO project. In *International Conference on Unmanned Aircraft Systems (ICUAS)*., 2013.
- Lau, H.C. & Zhang, L., 2003. Task allocation via multi-agent coalition formation: Taxonomy, algorithms and complexity. In *15th IEEE International Conference on Tools with Artificial Intelligence*., 2003.
- Lin, L. & Goodrich, M.A., 2009. UAV Intelligent Path Planning for Wilderness Search and Rescue. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*., 2009.
- Li, D., Sun, X. & Li, X., 2012. Multi-UAVs Cooperative Target Tracking Control Law Design Based on Computer Vision. *Journal of Systems Engineering and Electronics*, 34(2), pp.364-68.
- Liu, Y. & Sun, D., 2012. *Biologically inspired robotics*. CRC Press.
- Lo, V.M., 1998. Heuristic Algorithms for Task Assignment in Distributed Systems. In *IEEE Transactions on Computers*., 1998.
- Lum, C., 2009. *Coordinated Searching and Target Identification Using Teams of Autonomous Agents*. ProQuest.

- Lutz, R., 2011. Software Engineering for Space Exploration. *Computer*, 44(10), pp.41-46.
- Manisterski, E., David, E., Kraus, S. & Jennings, N.R., 2006. Forming efficient agent groups for completing complex tasks. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems.*, 2006.
- Martinez-Val, R. & Perez, E., 2009. Aeronautics and Astronautics: Recent Progress and Future Trends. *Journal of Mechanical Engineering Science*, 223(12), pp.2767-820.
- Maza, I. et al., 2011. Experimental Results in Multi-UAV coordination for Disaster Management and Civil Security Applications. *Journal of Intelligent and Robotic Systems*, 61(1-4), pp.563-85.
- McCarley, J.S. & Wickens, C.D., 2005. *Human factors implications of UAVs in the national airspace*. Urbana-Champaign: University of Illinois.
- Mercker, T., Casbeer, D.W., Millet, P.T. & Akella, M.R., 2010. An extension of consensus-based auction algorithms for decentralized, time-constrained task assignment. In *American Control Conference (ACC).*, 2010.
- Merino, L., Caballero, F., Martínez-de Dios, J.R. & Ollero, A., 2005. Cooperative fire detection using unmanned aerial vehicles. In *Proceedings of the International Conference on Robotics and Automation, IEEE.*, 2005.
- Müller, V.C., 2012. Autonomous cognitive systems in real-world environments: less control, more flexibility and better interaction. *Cognitive Computation*, 4(3), pp.212-15.
- Nakamura, S., Nakagawa, H., Tahara, Y. & Ohsuga, A., 2013. Towards solving an obstacle problem by the cooperation of UAVs and UGVs. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing.*, 2013.
- Nodine, M., Chandrasekara, D. & Unruh, A., 2001. Task Coordination Paradigms for Information Agents. In *Proceedings of the 7th International Workshop on Agent Theories, Architectures and Languages.*, 2001.
- Papadales, B. & Downing, M., 2005. UAV science missions: A business perspective. In *Infotech@Aerospace Conferences*. Arlington, 2005. AIAA.

- Pastor, E., Lopez, J. & Royo, P., 2007. UAV Payload and Mission Control Hardware/Software Architecture. *Aerospace and Electronic Systems Magazine, IEEE*, 22(6), pp.3-8.
- Pettersen, K.Y., Liljebäck, P., Stavadahl, Ø. & Gravdahl, J.T., 2013. Snake Robots From Biology to Nonlinear Control. *Nonlinear Control Systems*, 9(1), pp.110-15.
- Plowes, N., 2010. An Introduction to Eusociality. *Nature Education Knowledge*, 1(11), p.7.
- Ponda, S. et al., 2010. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC)*., 2010.
- Ren, W., Beard, R.W. & Atkins, E.M., 2007. Information Consensus in Multi-Vehicle Control. *Control Systems, IEEE*, 27(2), pp.71-82.
- Richards, A., Bellingham, J., Tillerson, M. & How, J., 2002. Coordination and control of multiple UAVs. In *AIAA guidance, navigation, and control conference*. Monterey, CA, 2002.
- Schneiderman, R., 2012. Unmanned Drones are Flying High in the Military/Aerospace Sector. *Signal Processing Magazine, IEEE*, 29(1), pp.8-11.
- Schweiger, D.M., Sandberg, W.R. & Ragan, J.W., 1986. Group approaches for improving strategic decision making: A comparative analysis of dialectical inquiry, devil's advocacy, and consensus. *Academy of management Journal*, 29(1), pp.51-71.
- Seeley, T.D., Camazine, S. & Sneyd, J., 1991. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28(4), pp.277-90.
- Shehory, O. & Kraus, S., 1998. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1), pp.165-200.
- Shehory, O.M., Sycara, K. & Jha, S., 1998. Multi-agent coordination through coalition formation. In *Intelligent Agents IV Agent Theories, Architectures, and Languages*. Springer Berlin Heidelberg. pp.143-54.
- Sujit, P.B. & Beard, R., 2007. Multiple MAV Task Allocation using Distributed Auctions. In *AIAA Guidance, Navigation and Control Conference and Exhibit*., 2007.

- Tahbaz-Salehi, A. & Jadbabaie, A., 2006. On Consensus Over Random Networks. In *44th Allerton Conference on Communication, Control, and Computing.*, 2006. 44th Annual Allerton Conference.
- Tisdale, J. et al., 2006. The software architecture of the Berkeley UAV platform. In *IEEE International Conference on Control Applications.*, 2006.
- Tofilski, A. et al., 2008. Preemptive Defensive Self-Sacrifice by Ant Workers.. *The American Naturalist*, 172(5), pp.239-43.
- Tomic, T. et al., 2012. Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue. *Robotics & Automation Magazine, IEEE*, 19(3), pp.46-56.
- Tuna, G. et al., 2012. Unmanned Aerial Vehicle-Aided Wireless Sensor Network Deployment System for Post-disaster Monitoring. In *Emerging Intelligent Computing Technology and Applications*. Springer Berlin Heidelberg. pp.298-305.
- Valenti, M. et al., 2007. Embedding health management into mission tasking for UAV teams. In *American Control Conference. IEEE.*, 2007. Proceedings of the 2007 American Control Conference.
- Whitten, A.K., Choi, H.L., Johnson, L.B. & How, J.P., 2011. Decentralized task allocation with coupled constraints in complex missions. In *American Control Conference (ACC).*, 2011.
- Willmann, J. et al., 2012. Aerial Robotic Construction Towards a New Field of Architectural Research. *International Journal of Architectural Computing*, 10(3), pp.439-60.
- Wood, R.J., 2008. The first takeoff of a biologically inspired at-scale robotic insect. *IEEE Transactions on Robotics*, 24(2), pp.341 - 347.
- Wooden, D. et al., 2010. Autonomous navigation for BigDog. In *IEEE International Conference on Robotics and Automation (ICRA).*, 2010.
- Wright, M.B., 1990. Speeding up the Hungarian algorithm. *Computers & Operations Research*, 17(1), pp.95-96.

Wu, C.W., 2006. Synchronization and Convergence of Linear Dynamics in Random Directed Networks. *IEEE Transactions on Automatic Control*, 51(7), pp.1207-10.

Yamaguchi, H., Arai, T. & Beni, G., 2001. A Distributed Control Scheme for Multiple Robotic Vehicles to Make Group Formations. *Robotics and Autonomous systems*, 36(4), pp.125-47.

Zhao, Z., Ding, Q., Wang, Z. & Chen, L., 2012. A Coupled Approach to Wilderness Search and Rescue Problem Based on Cross-Entropy. In *Proceedings of the 2012 International Conference on Electronics, Communications and Control.*, 2012.

Zhu, S., Wang, D. & Low, C.B., 2013. Cooperative Control of Multiple UAVs for Source Seeking. *Journal of Intelligent & Robotic Systems*, 70(1-4), pp.293-301.

Appendix A

A.1 TASK ASSIGNMENT AND CONSENSUS WITH THE CBBA

This section provides an insight into a typical assignment build up using the CBBA. Figure A.1 displays the initial layout of an example scenario involving 4 agents and 5 single-agent tasks.

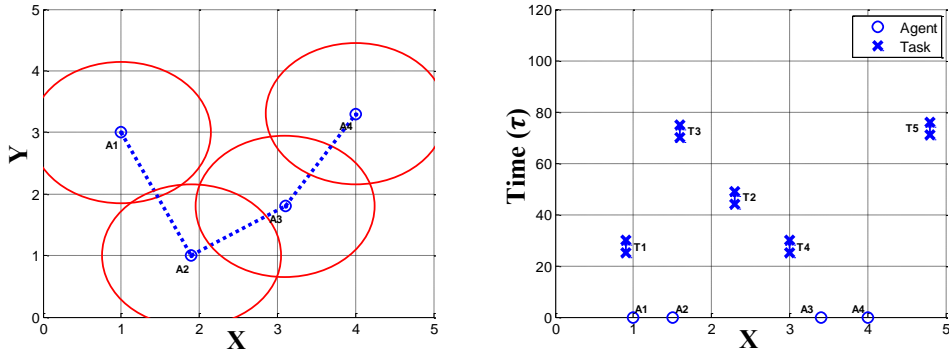


Figure A.1: Initial set up of the example simulation with a top down view of the agent starting positions and their communication network (left) and the view of task positions and their time windows (right).

Each agent individually constructs their own bundle of tasks they plan to complete. With no knowledge about any other agent's bids each agent attempts to complete as many tasks as it can to produce the highest score possible. Figure A.2 displays the initial path agents will take to complete their assignments, without any prior communication means every task produces a conflicting assignment. Table A.1 shows the state of agent A1's winning agent list z_i and winning bid list y_i containing the assignments it has initially decided to do. Table A.2 shows the data that A2 will send to both agents A1 and A3, for the initial round of consensus agent A1 will only have assignments from agent A2, assignments from other agents will need to propagate through the communication network where only valid winning bids will reach A1.

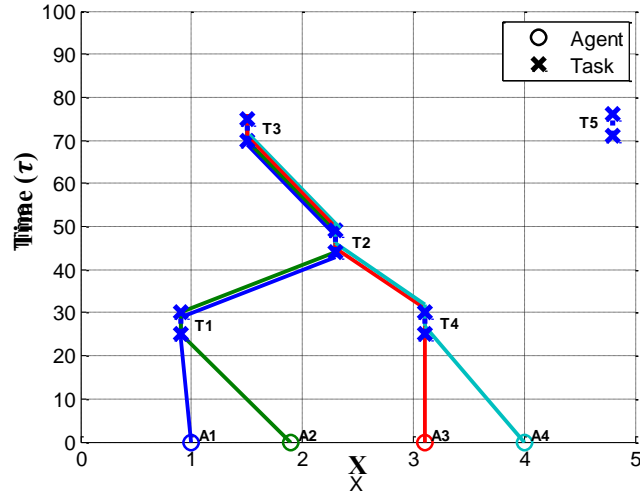


Figure A.2: Initial assignments before any communication and consensus has taken place. Agent paths offset to allow easier viewing of agents on the same path.

Table A.1: Agent A1's initial winning agent list z_i and winning bid list y_i after the bundle construction phase but before any consensus.

Winning bid list y_i for agent A1				
$T1$	$T2$	$T3$	$T4$	$T5$
73.98	71.56	72.44	0	0

Winning agent list z_i for agent A1				
$T1$	$T2$	$T3$	$T4$	$T5$
A1	A1	A1	0	0

Table A.2: Agent A2's winning agent list z_i and winning bid list y_i that is communicated to agent A1 and A3 after the bundle construction phase.

Winning bid list y_i for agent A2				
$T1$	$T2$	$T3$	$T4$	$T5$
71.98	71.56	72.44	0	0

Winning agent list z_i for agent A2				
$T1$	$T2$	$T3$	$T4$	$T5$
A2	A2	A2	0	0

After each agent has received bid information from their neighbours each agent consensus the data with its own keeping only the highest bids and removing any other bids. When agent A1 receives the assignment data from agent A2 as seen in Table A.2, it determines that it has the better bid for task T1, however, it loses the assignments for tasks T2 and T3 because both agents provide the same score and in a tie break situation a higher agent ID determines the winning in this case $A2 > A1$.

Once the winning data an agent has received has been merged with its own data through the consensus phase, agents return back to the bundle phase where any tasks they have lost are removed as well as any tasks that happen later in the agent's path. Again each agent adds tasks to its own bundle that give it the greatest improvement in score, however, this time agents can only add tasks that can beat previous winning bids if such a bid exists. Figure A.3 displays the assignments of agents and tasks after the second iteration of the bundle phase.

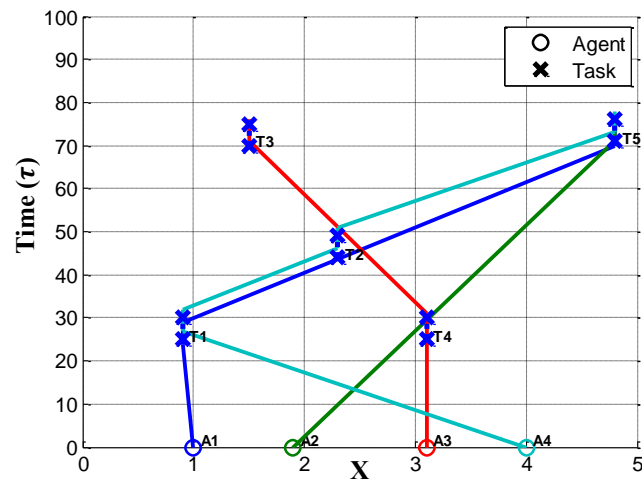


Figure A.3: Assignments after the second iteration of the bundle phase. Agent paths offset to allow easier viewing of agents on the same path.

Table A.3: Agent A1's winning agent list z_i and winning bid list y_i after the second iteration of the bundle construction phase.

Winning bid list y_i for agent A1				
$T1$	$T2$	$T3$	$T4$	$T5$
73.98	71.56	72.44	0	67

Winning agent list z_i for agent A1				
$T1$	$T2$	$T3$	$T4$	$T5$
A1	A2	A2	0	A1

Table A.4: Agent A2's winning agent list z_i and winning bid list y_i that is communicated to agent A1 and A3 after the second iteration of the bundle construction phase.

Winning bid list y_i for agent A2				
$T1$	$T2$	$T3$	$T4$	$T5$
73.98	72.44	72.44	73.6	68.47

Winning agent list z_i for agent A2				
$T1$	$T2$	$T3$	$T4$	$T5$
A1	A3	A3	A3	A2

The assignment data for agent A1 in Table A.3 shows that agent A1 still thinks agent A2 is assigned to task T2 and T3, this information will be corrected once it receives another communication message from A2. Table A.4 shows the new assignment data that A2 has, in this the second round of communication it can be seen that agent A1 will now be given some information about the assignments of A3. Although it can be seen that the data is only partially valid but with each round assignments change less and valid assignments will propagate through the communication network. Figure A.4 shows the assignments at each round until finally all the agents converge on a single conflict free solution.

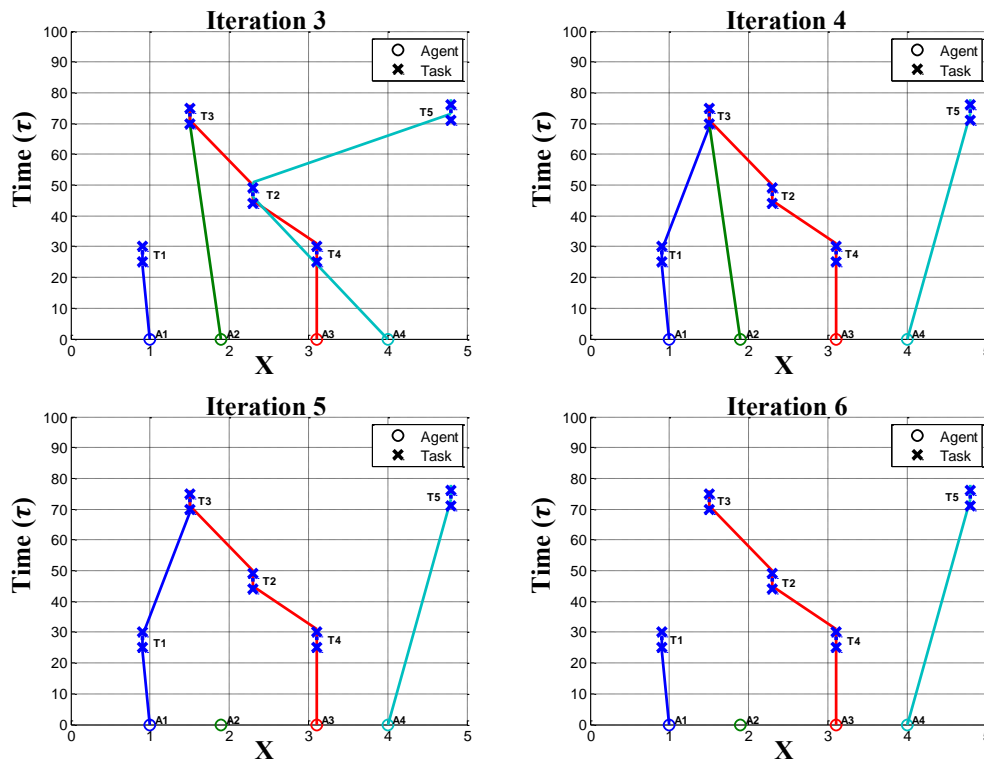


Figure A.4: Agent assignments at successive iterations of the CBBA. Agent paths offset to allow easier viewing of agents on the same path.

A.2 TASK ASSIGNMENT AND CONSENSUS WITH THE CBGA

This section provides the convergence of a multi-agent assignment created using the CBGA. Figure A.1 displays the initial layout of an example scenario involving 4 agents and 5 multi-agent tasks where $L_j = 2, \forall j$. This scenario is exactly the same as that found in Figure A.1 except this time each task is a multi-agent task requiring the CBGA to form assignments.

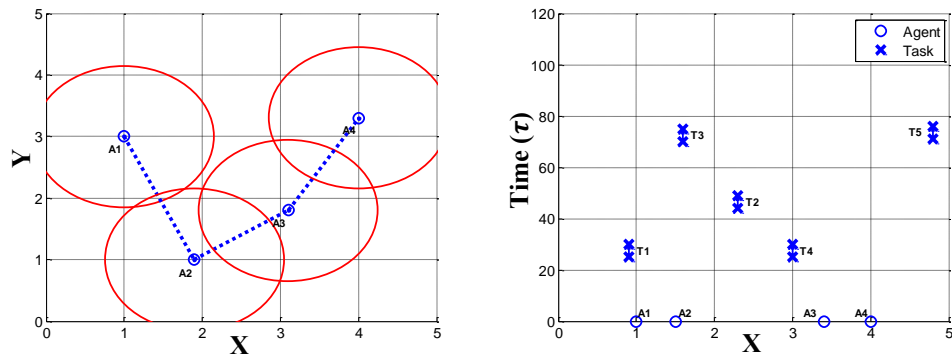


Figure A.5: Initial set up of the example simulation with a top down view of the agent starting positions and their communication network (left) and the view of task positions and their time windows (right).

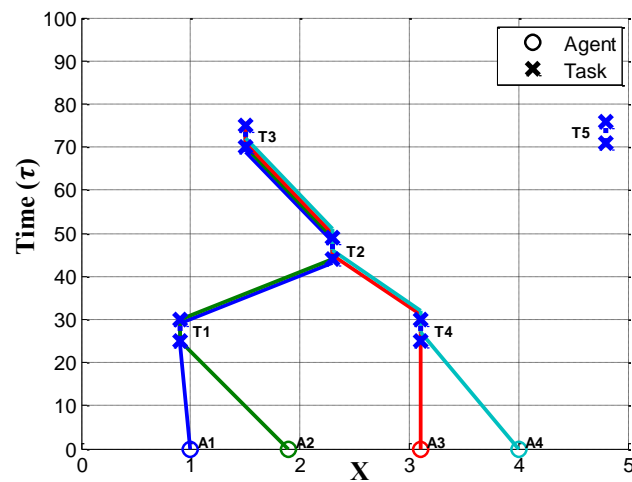


Figure A.6: Initial assignments before any communication and consensus has taken place. Agent paths offset to allow easier viewing of agents on the same path.

The initial assignment displayed in Figure A.6 is the same first assignment as found with the CBBA, replacing the single-agent tasks with multi-agent tasks does not have any effect on how agents initially build their assignments. Each agent finds the path that provides the greatest individual improvement. Similarly the assignment matrices in Table A.5 and Table A.6 contain the same bids as found in the CBBA.

Table A.5: Agent A1's initial winning agent matrix x_i and winning bid matrix y_i after the bundle construction phase but before any consensus.

Winning bid matrix y_i for agent A1					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	73.9	71.6	72.4	0	0
A2	0	0	0	0	0
A3	0	0	0	0	0
A4	0	0	0	0	0

Winning agent matrix x_i for agent A1					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	1	1	1	0	0
A2	0	0	0	0	0
A3	0	0	0	0	0
A4	0	0 </td <td>0</td> <td>0</td> <td>0</td>	0	0	0

Table A.6: Agent A2's winning agent matrix x_i and winning bid matrix y_i that is communicated to agent A1 and A3 after the bundle construction phase.

Winning bid matrix y_i for agent A2					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	0	0	0	0	0
A2	71.9	71.6	72.4	0	0
A3	0	0	0	0	0
A4	0	0	0	0	0

Winning agent matrix x_i for agent A2					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	0	0	0	0	0
A2	1	1	1	0	0
A3	0	0	0	0	0
A4	0	0	0	0	0

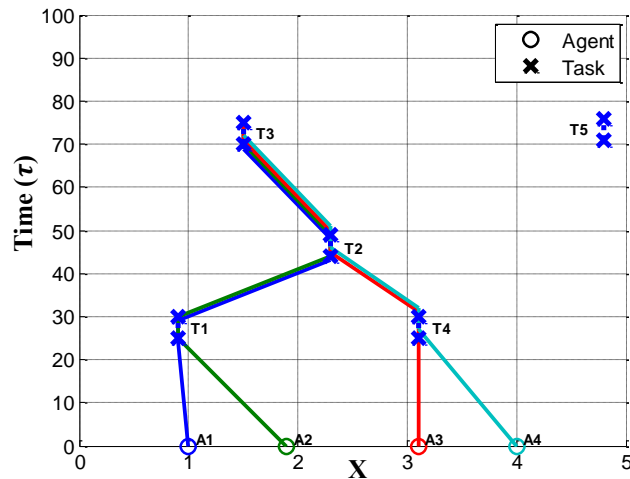


Figure A.7: Assignments after the second iteration of the bundle phase. Agent paths offset to allow easier viewing of agents on the same path.

Figure A.7 shows the assignments after the second bundle phase which is exactly the same as the previous assignments in Figure A.6. Because each multi-agent task allows two assignments, no agent has enough information to believe its assignments are not valid. Agent A1 believes only two agents are currently assigned to tasks T1, T2 and T3 as seen in Table A.7. Table A.8 shows that agent A2 knows that this is not true, however, because its ID is greater than A1 it has unassigned agent A1 and kept itself on all three tasks.

Table A.7: Agent A1's winning agent matrix x_i and winning bid matrix y_i after the second iteration of the bundle construction phase.

Winning bid matrix y_i for agent A1						Winning agent matrix x_i for agent A1					
	T1	T2	T3	T4	T5		T1	T2	T3	T4	T5
A1	73.9	71.6	72.4	0	0	A1	1	1	1	0	0
A2	71.9	71.6	72.4	0	0	A2	1	1	1	0	0
A3	0	0	0	0	0	A3	0	0	0	0	0
A4	0	0	0	0	0	A4	0	0	0	0	0

Table A.8: Agent A2's winning agent matrix x_i and winning bid matrix y_i that is communicated to agent A1 and A3 after the second iteration of the bundle construction phase.

Winning bid matrix y_i for agent A2						Winning agent matrix x_i for agent A2					
	T1	T2	T3	T4	T5		T1	T2	T3	T4	T5
A1	73.9	0	0	0	0	A1	1	0	0	0	0
A2	71.9	71.6	72.4	0	0	A2	1	1	1	0	0
A3	0	72.4	72.4	73.6	0	A3	0	1	1	1	0
A4	0	0	0	0	0	A4	0	0	0	0	0

At the third iteration of the CBGA the final solution has been reached as can be seen in Figure A.8 although insufficient time has elapsed such that agent A1 still has no information about agent A4's assignments as seen in Table A.9. This assignment will still take a few more iterations before every agent converges on the same solution. Table A.10 shows that agent A2 is very close to having the complete assignment but is still missing the final assignment from A1.

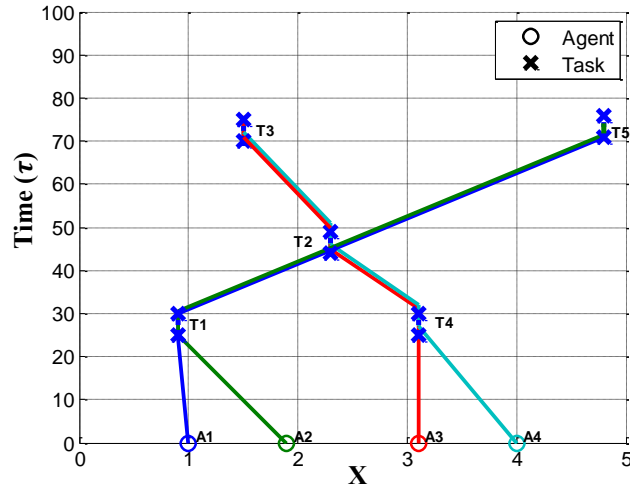


Figure A.8: Final assignment of agents, agent paths offset to allow easier viewing of agents on the same path.

Table A.9: Agent A1's winning agent matrix x_i and winning bid matrix y_i after the third iteration of the bundle construction phase.

Winning bid matrix y_i for agent A1					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	73.9	0	0	0	67
A2	71.9	71.6	72.4	0	0
A3	0	72.4	72.4	73.6	0
A4	0	0	0	0	0

Winning agent matrix x_i for agent A1					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	1	0	0	0	0
A2	1	1	1	0	0
A3	0	1	1	1	0
A4	0	0	0	0	0

Table A.10: Agent A2's winning agent matrix x_i and winning bid matrix y_i that is communicated to agent A1 and A3 after the third iteration of the bundle construction phase.

Winning bid matrix y_i for agent A2					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	71.9	0	0	0	0
A2	71.9	0	0	0	68.5
A3	0	72.4	72.4	73.6	0
A4	0	72.4	72.4	73.1	0

Winning agent matrix x_i for agent A2					
	$T1$	$T2$	$T3$	$T4$	$T5$
A1	1	0	0	0	0
A2	1	0	0	0	1
A3	0	1	1	1	0
A4	0	1	1	1	0