2012

Multi-Robot Auction Based Coordination

Eisa Hassan Osman

North Carolina Agricultural and Technical State University

MULTI-ROBOT AUCTION BASED COORDINATION

by

Eisa Hassan Mohamed Osman

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Department: Electrical & Computer Engineering
Major: Electrical Engineering
Major Professor: Dr. Abdollah Homaifar
Co-advisor: Dr. Albert Esterline

North Carolina A&T State University
Greensboro, North Carolina
2012

ABSTRACT

Osman, Eisa Hassan Mohamed. MULTI-ROBOT AUCTION BASED COORDINATION. (**Major Advisor: Dr. Abdollah Homaifar**) North Carolina Agricultural and Technical State University. (**Co-advisor: Dr. Albert Esterline**) North Carolina Agricultural and Technical State University.

This dissertation studied the coordination problem for a Task Initiator (TI) with multiple ground stations (GSs). Each GS has a team of unmanned aerial vehicles (UAVs) that frequently collected data from a set of unattended ground sensors (UGSs) and delivered it to the source ground station (GS). The GSs made the information available to the TI. This problem formulated into a continuous, time-constrained version of the multi-travelling salesmen problem. A market-based coordination mechanism is presented that uses the concepts of price, revenue, cost, and a sequence of first-price, one-round auctions between the TI and GSs from one side, and double auction between GSs and UAVs from another side to distribute data collection tasks efficiently among team members. In a dynamic environment, this approach promises robustness, adaptation, and graceful degradation. Tasks from GS-to-UAV are double first-price sealed-bid sequential procurement auctions possibly with (additional) subcontracting (negotiation) and TI as a market matcher. To the author's knowledge, this is the first occurrence of using double auction as a coordination method in robot industries.

School of Graduate Studies
North Carolina Agricultural and Technical State University

This is to certify that the Doctoral Dissertation of

Eisa Hassan Mohamed Osman

has met the dissertation requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2012

Approved by:

Dr. Adollah Homaifar Major Advisor	Dr. Marwan Bikdash Committee Member
Dr. Albert Esterline Co-Advisor	Dr. Numan Dogan Committee Member
Dr. Christopher Doss Committee Member	Dr. John Kelly Department Chairperson

Dr. Sanjiv Sarin
Associate Vice Chancellor for Research and Graduate Dean

DEDICATION

My dissertation dedication and my most wholehearted thanks go to my grand family in Sudan. To my soul Awab, Rawiah and Ali you are the light of my eyes, source of the hope who lived in me and the base of my happiness, which accompanied me in my life. Without their unconditional love, patience, support and guidance, I would have never come this far. I cannot thank them enough for all I have been through. Finally, my most special thanks go to those who supported me with their prayers during my absence, for those who see the light after darkness and their hope hanging on the doors of skies and those who treat obstacles as a temporary period for taking their breath before they continue their long journey. May ALLAH (SWT) accept us and Reward us Paradise in the Hereafter.

BIOGRAPHICAL SKETCH

Eisa Hassan Mohamed Osman was born January 1, 1968, in Sudan. He received the Bachelor of Science degree in Public Administration from the National School of Public Administration, Rabat, Morocco in 1990, a graduate diploma in Technical Science from the University of Manchester Institute of Science and Technology, Manchester, U.K. in 1993, and a Master of Science degree in Computer Science from North Carolina Agricultural and Technical State University, Greensboro, NC, USA. He is a candidate for the Ph.D. degree in Electrical Engineering. While at North Carolina Agricultural and Technical State University, Mr. Osman served as a teaching assistant for undergraduate students. He also worked with a new group of engineering graduate students to revitalize the Graduate Engineering Student Alliance. Mr. Osman was also active in the NC A&T Chapter of the National Society of Black Engineers (NSBE). Additionally, he served as an active member of the Institute of Electrical and Electronics Engineers (IEEE). Mr. Osman's research includes project collaboration with the Wright Patterson Air Force Base, Dayton, Ohio. Currently, Mr. Osman works as a research scientist at the Detonation Science Branch, Naval Air Systems Command (NAVAIR), Naval Air Warfare Center Weapons Division (NAWCWD), in China Lake, California.

ACKNOWLEDGEMENTS

The work in this dissertation was possible because of the support, guidance, and inspiration of so many individuals: family, friends, and colleagues. Their help and good advice constituted the basis of this research. Thanks to all of them. Particularly, I would like to thank my advisor, Professor A. Homaifar, for giving me the opportunity to work on this topic. I learned a great deal from his motivation and clarity of purpose. Of course, I am truly grateful to my co-advisor Dr. A. Esterline for his guidance, support, hard work, and patience. This dissertation would not have come to the form and shape it is today without him. I also acknowledge the rest of my doctoral committee, Dr. Marwan Bikdash, Dr. Numan Dogan, and Dr. Christopher Doss for all their knowledge and information they gave me over the duration of my doctoral degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In today's dynamic environment, the need of mapping a wider operational area gains significant attention due to the sensitivity of the information needed at real-time to the control unit such as in rescue, reconnaissance, and real-time surveillance operations. This issue is particularly relevant in military applications. In situations where unattended aerial vehicles (UAVs) need to communicate with a control unit, many problems arise such as sensor ranges and bandwidth. Distributing multiple ground stations (GSs) geographically will reduce the communication bottleneck, and simplify computation of deployed information. Interactions between many ground stations, a task initiator (TI), and UAVs become paramount, and require a special design to complete the mission effectively. Based on knowledge of the economic area, incorporating different auction approaches in engineering fields has helped resolve such issues. In this problem, single first-price sealed-bid sequential procurement auctions used between the TI and ground stations and double first-price sealed-bid sequential procurement auctions possibly with (additional) subcontracting (negotiation) from ground stations to UAVs.

This chapter provides a general overview of the research work, and an overview of each chapter in this dissertation.

1.1 Multi-UAVs and Data Collection

The rapid growth of sensor technologies in recent years has enabled scientists to solve complicated or difficult problems in many applications, such as the battlefield. Various mission tasks, such as target detection, reconnaissance and surveillance, and situation awareness include the major area of applications of unattended ground sensors (UGS) technologies.

In the robotic field, scientists follow different approaches when dealing with task allocation. There are three principal approaches to deal with task allocation, namely, centralized, distributed, and market-base. In a centralized approach, the robotic team is treated as a single system with many degrees of freedom. The leader or manager has the ability to plan for the entire team that requires the follower to inform the leader with their information to enable a manager to carry out actions. Since the leader has all the knowledge about the environment, the leader can perfectly allocate tasks based on this knowledge. The centralized approach produces optimal or near optimal results at the expense of high computational overhead and is prone to malfunctioning.

Overcoming the shortcoming of the centralized approach has encouraged scientists to come up with the idea of a distributed system. In this approach, distributing the responsibility of planning to the whole team reduces the communication bottleneck, and response to dynamic conditions is faster.

In general, this system is robust, and no single point of failure can occur. Conversely, basing the decision only on local information the results in general are highly sub-optimal. Preserving the advantages of centralized and distributed approaches, scientists migrates ideas from economic areas to overcome the disadvantages of these approaches, which led to the birth of market-based coordination. Market-based coordination is based on the free market economic model. Essentially, the competition between members is to maximize their profit. Members under this system usually compete to achieve their goals and sometimes negotiation with other team members to reach their goals.

A market-based approach accommodates multiple auctioneers who distribute an incoming information load among themselves. There are three types of auctions for acquiring a commodity, namely, English auction, Dutch auction, first-price sealed-bid auction, and second-price sealed-bid auction. In an English auction, an “ascending” movement of a potential buyer bids occur until the bidding stops. The winning bidder receives the item at the highest price, which could be less than its maximum valuation; however, is not always the case because bidders may tend to overbid, which causes the item to exceed its true valuation (Fasli, 2007).

The Dutch auction is an open, “descending,” bid auction designed to handle multiple identical items (usually in a lot). In this auction, the seller sets an opening price. If no bids are made, the price is lowered until a bid is received. The first bidder wins the first option of buying all or part of the lot. The bidder may lose the item if he/she waits too long to enter their bid.

In a first-price sealed-bid auction (FPSB), each bidder submits a sealed bid without knowing other bidders' valuation of the item, which reflects a private valuation for the auctioned item. In this auction, the highest bidder is the winner and pays the amount of his/her bid (Fasli, 2007).

In a second-price sealed-bid auction, each participant submits a sealed bid. The highest bidder wins the auction, and only pays the price of the second highest bid. Therefore, it is the bidder’s advantage to bid his/her true valuation of the item.

A double auction is an environment where multiple buyers and sellers participate to trade a commodity. In this environment, each buyer and seller submits a bid representing his/her offer to sell or buy the auctioned commodity. Then, submitted bids are matched, and afterwards the auction is cleared. Double auction, in this context, is a two-sided auction; one side represents a centralized approach while the other side represents a distributed approach, allowing the market to compute the information in an efficient manner while providing quick responses.

As in battlefields, providing quick responses is a key factor when transmitting data back and forward to the TI. A TI could also be a control room or an agent with the ability to initiate tasks. Transmitting data are also a challenge when connectivity is an issue or when there are limited communication ranges such as unmanned ground sensors (UGSs) and large distances between UGSs and the TI. However, task allocation through multiple ground stations (GSs) is more cost effective than single GSs for mapping wider areas and acquiring better robustness.

To improve reliability, performance, and the cost of task allocation, logically one should consider a market-based coordination mechanism such as a free market economic model, which provides better results for optimizing distance data delivery from multiple UGSs.

Multiple GSs, in this context, will provide an excellent opportunity to overcome communication problems or sensor range limitations by forming a chain of communication to receive information from a further distance, reduce communication bottleneck, provide relief or reduce computation complexity for the control center (Moore et al., 2005).

When using double auction, the TI, GSs, and UAVs formulate layers of communications that provide quick responses in a robust environment to benefit from the market-based mechanism.

1.2 Dissertation Scope

The focus of this research is two-fold:

1. The coordination problem of task initiator (TI), ground stations (GSs) and a team of UAVs that is employed to frequently visit a set of remote UGSs, collect data from them, and return to the ground station to deliver the collected data. As mentioned earlier, the importance of time as a factor can be realized when GSs receive this data. A time constraint on data delivery is one of the concerns of this dissertation. The deadline limit on data delivery time should not exceed a specific time in order to validate the accuracy of the data. In particular, to achieve any task, the time between two successive tasks should not exceed a certain deadline time. This constraint is imposed by the nature of the UGS applications (as in

target detection and situation awareness), in which late-delivered data will lose its sensitive value and may not be useful anymore and may result in hazardous situations, and

2. The problem of assigning tasks when the environment has multiple GSs and multiple UAVs with full degree of freedom.

UAVs are responsible for accomplishing tasks such as tracking enemy targets in battlefields or gathering information from UGS. In these scenarios, the UAVs must coordinate their actions with GSs, usually through communication, in order to achieve their goals. The UAVs must make independent decisions based on their perception of the environment, and act in a manner that optimize the global utility.

Resources (utilities) or energy consumption is a constraint that the coordination system should satisfy. Optimizing the average distance traveled of performing any task should be part of the coordination methods. Under these conditions, this problem formulated as a continuous and time constrained version of the multi-traveling salesmen problem (MTSP).

Recent studies showed that multi-agent systems operating in dynamic environments such as surveillance, reconnaissance, and battlefields are highly prone to failures of many kinds, and it is crucial that the coordination method that deals with such kind of environment be robust to these failures (Ajorlou et al., 2007 and Dias et al., 2004).

Introducing the distributed coordination system helps to understand operating in this type an environment. Market based coordination is one mechanism that have an effective usage in an environment in which frequent auctioning, time limited contracts, and time-dependent prices ensure robustness in the face of loss of team members and failures of individuals (Dias et al., 2004).

Stentz and Dias (2003) in TraderBots market-based coordination approach cited the work of (Smith, 1980) in Contract Net Protocol, the implementation of contract net protocol by (Sandholm, 1993), and an extension of it by (Sandholm and Lesser, 1995). These concepts were used to control different dynamic environment systems. Stentz and Dias (1999) proposed a market-based approach for multi-robot coordination, which aims to exploit the desirable properties of both distributed and centralized approaches. In order to take advantage of such approaches, (Dias et al., 2004) proposed a distributed task allocation protocol that uses the concepts of cost, revenue, and profit that efficiently distribute available tasks among team members through a sequence of multiple different auctions. In this environment, each agent is self-interested in maximizing their personal profit, which can lead to a near global optimal plan for the entire team provided the costs and price functions are well defined. Generally, in this kind of task allocation, the cost of the task will determine its priority among other tasks. Adding a new task will be constrained to the due time of other tasks on the agent's current plan. So the lower the cost, the more demand needed to perform it. It is clear that the task's cost is not always the main factor. In some situations, a task will be given higher priority even though it has a high execution cost compared to other tasks due to the sensitivity of task's information at the time.

Negotiation between UAVs after clearing the auction will improve system efficiency by reducing the cost to participating agents. Because of the agents' interest, they will try to maximize their profit and reduce their cost. By design, the auction can accommodate a situation where an agent auctioned a task earlier even when the agent was not bidding during the auction time because this agent is deemed fittest to perform the task. In a system-optimized model, different negotiations produce the same result. Frequent auctioning will accommodate the recovery of task(s) timed out due to agent failure or death; reallocating these tasks to new agents will cause the system to be robust and guarantee the delivery of all auctioned tasks.

1.3 Overview of Chapters

Chapter 2 gives a general overview of the basic concepts in market-based coordination including auction mechanisms, types, and approaches. It also provides a literature review. Chapter 3 focuses on the problem formulation and methodology. Particular emphasis is on different auction structures. Issues related to cost estimation and robustness are also discussed. Presented in Chapter 4 are the simulation results, and the performance of the double auction. The dissertation concludes with Chapter 5, which provides concluding comments and possible improvements.

CHAPTER 2

LITERATURE REVIEW

2.1 Relevant Work

Multi-robot coordination has received much attention in the last few decades. This is due to the demand for automation in application domains where multiple robots can accomplish the same tasks more efficiently than a single robot. With a team of coordinated robots, tasks achievement is faster, safer, better than a single robot, and can accomplish operations that a single robot cannot execute alone (Lemair et al., 2004). Accordingly, coordinating multiple robots to complete a task cooperatively is a difficult problem that has attracted much attention from the robotics research community. Based on the manner in which team members interact, multi-robot coordination mechanisms can be categorized into two groups: intentional swarm type cooperation. Deneubourg, et al., (1991) mentioned that in swarm-type robotic systems, numerous homogeneous autonomous robots interact directly by exchanging their information with one another or by acting on their environment; this collective activity may produce coordinated behavior. In contrast to this, there is an intentional coordination, in which agents negotiate explicitly and exchange task related information. The motivation behind this kind of coordination is to satisfy mutual interest.

Lemair et al., (2004) mentioned that the potential applications of this kind of coordination range from mapping missions of buildings or in a natural environment, rescue or intervention missions in hazardous areas to planetary exploration or deployment of equipment without human intervention. A system supported with several robots to perform a given mission should be flexible enough to allow robots to allocate tasks to each other and build their plans accordingly to complete the mission. They should also be able to modify the allocation dynamically and consequently to their plans to adapt to changes in their environment or to new requests issued by the operator. However, the system must also satisfy the limited constraints on energy resources and communication ranges.

In their TraderBots, (Dias et al., 2004) mentioned that multi-agent systems operating in dynamic environments such as battlefields must accommodate many kinds of failures, frequent dynamical changes, and uncertain or imperfect information. Therefore, it is crucial that any kind of coordination methods applied for multi-agent systems be able to function well under such conditions. Market based coordination was derived from a category of intentional coordination mechanisms, and is a promising method for handling these conditions. Frequent auctioning, time limited contracts, and time-dependent prices ensure robustness in the loss of team members and individual failures, which also enable the team to get by with uncertainty and online tasks introduced over time (Dias et al., 2004, and 2005). The distributed nature of market-based coordination enables the team to rely on local knowledge so they can respond quick and fast to dynamic changes within their environment without the need of a central planner. Since information is decomposed into bids, the market-based coordination systems can communicate efficiently and compute efficiently due to the parallelism.

During the last two decades, coordination mechanisms for multi-robot task allocation have been developed that are based on market-base coordination. The M+ (Botelho and Alami, 1999) architecture, based on a greedy algorithm, was the first market-based approach to multi-robot task allocation. The MURDOCH (Gerkey and Mataric, 2002), as a completely distributed system, offers a distributed approximation to a global optimum of resource usage, which is equivalent to an instantaneous greedy scheduler. An online task assignment algorithm also assigns a newly created task to the fittest available robot (Gerkey and Mataric, 2004). TraderBots models (Dias et al., 2004) represent a multi-robot team as an economy of self-interested agents that try to maximize their individual profits. In these models, reallocating tasks allow for solution improvements over initial assignments, and for adapting task assignments as new information is ascertained.

In this dynamic environment, agents who have the ability of planning for themselves and negotiating may do so by swapping some tasks (as self-interested agents). This redistribution of tasks and resources simultaneously at the end result in lower cost solutions, which imply some profit, and therefore will improve efficiency. Given appropriate costs and revenue functions, this method can lead to a near globally optimal allocation. Constrained tasks will not be dealt with in TraderBots where interrelated costs among the tasks are considered. Hoplites (Karla et al., 2005) seem to be the first market based approach to constrained task execution. In Hoplites, passive coordination produces locally developed solutions since agents frequently exchange information of their intended actions and locally select their actions. In a situation where there is a constraint violation, agents actively propose and bid on joint plans to resolve the constraint. The performance of Hoplites is validated in perimeter sweeping (Karla et al., 2005) and, more recently, in constrained exploration (Karla et al., 2006), during exploration of a hazardous area, robots are restricted to remain in communication with the base station directly or through a chain of teammates. Lemaire et al. (2004) put soft time constraints on subtasks of a complex task to synchronize subtask execution. To define the cost of a plan for tasks, needed are the sum of the distance cost of the plan and a cost term corresponding to the quality of the time-constraint satisfaction. Agents, therefore, will try to reduce the deviation from the expected execution time while trading tasks.

The price of a task determines the cost the auctioneer will pay an agent that accomplishes the task. Using time-varying prices, the auctioneer announces higher prices for tasks that have become more important. Therefore, bids reflect not only the agents' costs but also the importance of the tasks.

2.2 Market-based Coordination

This section briefly explains the basic concepts of market-based coordination mechanisms.

2.2.1 Overview

In market-based coordination methods, participants form an economy that allocates tasks to members through auctions. Normally, a user or team members that have task creation capability (Dias, et al., 2005) generate tasks. An auctioneer offers all its available tasks to other agents in its environment, collects their bids, evaluates the collected bids, and assigns some or all of its tasks to them. As discussed in the previous section, some market-based coordination systems allow reassignment of a task. This means that an agent in charge of performing a task have the ability to resell that task to another agent, e.g., TraderBots (Dias et al., 2004), M+ (Botelho and Alami,1999), the system presented in (Ajorlou et al., 2007), and Sandholm's implementation of the contract net protocol (Sandholm, 1993). In such systems, any team member can negotiate with teammates to improve their personal profit as a self-interested agent. An agent who offered a task may submit a bid on it. A submitted bid in this context represents the cost to the agent for performing the offered task. The global objective of the application and resource consumption are two main factors in bid valuation. By assigning the tasks to team members through a bidding process, the auctioneer tries to lower the overall team cost by allocating the tasks to team members with lower costs.

2.2.2 Instantaneous Assignment (IA) vs. Time-extended Assignment (TA)

Gerkey and Mataric (2002) categorized multi-robot task allocation mechanisms based on instantaneous assignment (IA) and time extended assignment (TA). In IA, robots do not have the ability to plan for their future activities, which mean the available information concerning the robots, the tasks, and the environment permits only an instantaneous allocation of tasks to robots, with no planning for future allocations. Therefore, the agents can only buy or sell one task at a time, which indicates that there is no room for parallelism. This type of allocation mechanism is useful for the applications in which tasks are introduced to the system online such as MURDOCH (Gerkey and

Mataric 2002), first-price auctions (Dias et al., 2003), and dynamic role assignment (Gerkey and Mataric, 2004).

In TA, agents have more information about the environment, such as the set of all tasks that needs an assignment, or a model of how tasks are expected to arrive over time. In this type of assignment, agents are allowed to make plans for the future by accepting more than one task at a time.

2.2.3 Auction Mechanisms

This section describes various types of auctions. In particular, how auctions differ and how auctioneers function within them.

2.2.3.1 Procurement Auction

A procurement auction, also called reverse auction, is a type of auction in which the role of the buyer and seller are reversed. The primary objective here is to drive purchase prices downward. In this kind of auction, sellers compete to obtain business. In a procurement auction, a buyer puts up a request to purchase a particular item. Multiple sellers bid to sell the requested item and the winner of the auction is the seller who offers the lowest price (www.wordiq.com/definition/Procurement_auction). In a procurement auction, the bidders seek a higher clearing price, and the auctioneer seeks a lower one.

2.2.3.2 Double Auction

Dynamic pricing mechanisms, and especially auctions with multiple buyers and sellers, are becoming popular in electronic commerce. “Double auction” refers to a market system where multiple buyers and sellers submit their bids for standardized units of well-defined items or securities by stating how much and at what price they will trade.

In double auction, each trader can express the subjective preference for the traded goods by using a utility function. Thus, properly defining the utility for representing each trader's preference is an important issue for research on double auction. Double auctions occur in an environment that has one commodity in the market with multiple buyers and sellers each submitting a single bid to buy or sell one unit of the commodity. According to (Fasli, 2007) the general process is as follows:

- Both buyer and sellers submit their bids.
- Bids rank from highest to lowest to generate demand and supply profiles.
- From the profiles, the maximum quantity exchanged can be determined by matching selling offers with demands bids.
- The transaction price is set and the market clears.

In this auction, each GS will sell only one task at a time, and any UAV will bid for only one task at a time. Each bidder has a private utility value for the item, which represents its real cost to perform such a task. The utility value from buyers', (respectively, sellers') point of view is the most (respectively, least) prices that they are willing to pay to buy (respectively, sell) the task. Although all market agents are self-interested, agents formulate their bids is based on the truthful value of the item.

A double auction could be either periodic or continuous. In a continuous double auction, buyers and sellers are matched immediately on detection of compatible bids, while in periodic double auction bids are collected over a specified period of time after which the market will be cleared.

2.2.3.3 Combinatorial Auction

A combinatorial auction is one where buyers and sellers have preferences on packages or bundles of commodities rather than only on one particular commodity at time. In this auction, bids are considered for combinations of different commodities. Consider a case where a set *S* of *n* tasks offered to the team members. Each agent calculates the cost of performing each subset of *S*, and submits a bid on that subset. After receiving all bids, the auctioneer evaluates them, and finds the partition of *S* with minimum cost. Bid calculation and winner determination are NP hard, which makes the combinatorial auctions intractable.

2.2.3.4 Parallel Auction

In parallel auctions, a set *S* of *n* tasks offered to the team members. Each agent calculates the cost of performing any of the offered tasks individually and submits a bid. The auctioneer then assigns each task to the agent that has submitted the lowest bid. Parallel auctions do not account for the dependencies among the tasks.

2.2.3.5 Sequential Auction

In sequential auctions, the set *S* of *n* tasks assigned through a sequence of *n* auctions, where only one task is sold in each auction. During each auction, each agent computes the cost of adding each unsold task to its current plan. Then, the task with the lowest cost is assigned to a corresponding bidder. Clearly, submitting bids for all the tasks offered will create communication complexity; but, since only one task is assigned during each auction cycle, it is in the best interest of each bidder to submit a bid only for the task that cost less among all auctioned tasks. This will have an impact of reducing the communication bottleneck and increase the bidder’s chance of winning the auctioned task.

Calculating marginal cost (the cost of adding a new task to the current plan) is NP hard in the MTSP case since it requires re-planning for the new set of tasks. In the heuristic used by (Ajorlou et al., 2007), a new task will be inserted between each two successive tasks into the agent plan to find the minimum cost of the new generated plan, which is the current plan. The difference between the current plan and the old one is the cost of performing the new task.

2.2.4 Auction Types

In a single attribute or a one-side auction, agents negotiate over one item, which is available by itself as a whole and not in combination. The negotiation has one dimension, usually price, and the relationship between buyers and sellers takes the form of one-to-one, one-to-many or many-to-many relationships.

2.2.4.1 Ascending-bid Auction (English Auction)

The English auction is the most common type of auction where the winning bidder receives the item at the highest price. The auction uses upward or “ascending” movement of potential buyer bids until the bidding stops. Bids may be oral, signaled, written or by third-party proxy in which one item or groups of items can be auctioned. Auction periods vary but are generally short. Items are frequently displayed to potential bidders prior to the auction with the reserve prices cited. E-Bay is a good e-commerce application of this type auction.

2.2.4.2 Descending-bid Auction (Dutch Auction)

The Dutch auction is an open, “descending” bid auction designed to handle multiple, identical items (usually in a lot). In this type auction, the seller sets an opening price. If no bids are made, the price is lowered until a bid is received. This first bidder wins the first option of buying all or a part of the lot. Other bidders have an opportunity to buy once the demand at that price is exhausted. Additional bidders may bid a lower price. This cycle continues until the lot is gone.

2.2.4.3 First-price Sealed-bid Auction

In a first price sealed bid auction (FPSB), each bidder submits a sealed bid that reflects its private valuation for the auctioned item without knowing other bidders’ valuations of the item. In this auction, the highest bidder is the winner and pays the amount of his/her bid. There are two distinctive phases (Fasli, M. 2007):

1. The bidder phase in which participants submit their bids.
2. The resolution phase in which the bids are opened and the winner is determined.

2.2.4.4 Second-price Sealed-bid Auction (Vickery Auction)

The second-price sealed-bid auction was named after William Vickery, a 1996 Nobel Prize recipient (Economics). In this type auction, each participant submits a sealed bid. The highest bidder wins the auction but only pays the price of the second highest bid. This auction fosters a bid strategy that reflects the buyer’s true valuation of the item. The Vickery approach gives all competing buyers an incentive to disclose their true best price since they can safely bid a price that would yield zero profit. The process can be used in a reverse auction method with the cheapest price winning but paying the second lowest bid price.

2.2.5 Auction Approaches

Normally, targets are scattered in the environment and the number of UAVs may be more or less than the number of targets available. In either case, an efficient task allocation method is needed for assigning UAVs to the targets. An efficient task allocation strategy should complete the mission (that is, delivering the target information to the GSs) in minimum time by direct assignment from the GSs or through negotiation with other UAVs in communication range. The classical solution for a task allocation problem would be to apply a centralized task allocation algorithm that generates the necessary commands for UAVs. However, centralized task allocations have well known limitations. Hence, there is a necessity to develop a decentralized task allocation algorithm. Here, briefly discussed are the concepts of centralized and distributed task allocation to assign tasks to UAVs.

2.2.5.1 Centralized Approaches

With centralized approaches, one agent (the leader) is responsible for planning for the entire team, while simultaneously taking into account the environment and the interactions of all team members at all times. All agents report to the leader and execute the plan. Although the centralized approach generates an optimal solution under the assumption that the information from the agents is available, it is intractable for a team of UAVs due to the complexity of operations (Karla et al., 2006, Sariel et al., 2006). Coordinating more than a few agents in the centralized approach causes a heavy communication load, and a problem with the bandwidth due to restrictions on the network. This approach is slow to incorporate new environmental information since new information must be sent back to the planner who re-computes the entire team's plan, usually at significant computational expense. Finally, a centralized approach does not allow a quick adaptation to change and tends to be brittle to failure. Primarily, centralized approaches have been used loosely in coordinated systems for task allocation (Sariel et al., 2006). Thus, centralized approaches are best suited for applications where teams are small and the environment is static or global state information is easily available.

2.2.5.2 Distributed Approaches

In a distributed approach, agents act independently and make decisions with local information about their state and their environment. For example, the UAV work with the ground stations or the central units such as TI to conduct their own plans based on available information. Here the role of the central control or ground station is for auctioning the task to the agents, evaluating received bids and awarding the winner agent. This approach tends to be more robust to failure, allowing for greater flexibility and tractability, and efficient for computation and communication. However, the solution remains sub-optimal. To emphasize the benefits of centralized approaches in distributed systems, market-based approaches have been designed to centrally plan over small subsets of the team where time and resources permit (Dias et al., 2004 and 2003).

In market-based frameworks, agents model an economy of self-interested individuals that buy and sell tasks and resources to maximize personal profit (Dias et al., 2005). This redistribution of tasks and resources simultaneously results in lower cost solutions for the team. Most of the distributed task allocations use an explicit communication message, which means that agents make decisions based on inter-agent communications transmitted at different times. This characteristic makes the algorithms more efficient, and with a higher level of fault tolerance than a centralized approach due to its distributed nature.

Negotiation over the distributed system generates a step-wise improvement. Negotiation techniques based on market rules (i.e., market-based approaches) fall within the distributed algorithms that make use of explicit communication. These techniques have received significant attention (Dias et al., 2006) since they offer a good compromise between communication requirements and the quality of the allocation.

2.2.5.3 Market-based Approaches

A task allocation algorithm can be a method of distributing common resources. Humans have dealt with similar problems for thousands of years with increasingly sophisticated market economies in which the individual pursuit of profit leads to the redistribution of resources and an efficient production of output. Therefore, market based approaches make use of the principles of the market economy and apply them to multi-agent coordination. This idea started with the Contract Net Protocol or CNP (Smith, 1980), which allocates tasks through negotiation of contracts. In this virtual economy, agents are traders, tasks are traded commodities, and virtual money acts as currency. Agents compete, despite being teammates in reality, to win tasks by participating in auctions that produce efficient distributions based on specified preferences. When the system is designed appropriately, each agent acts to maximize its individual profit, and simultaneously improves the efficiency of the team. This is the foundation of the success of the market-based approach; one engineers the costs, revenues, and auction mechanisms in such a way that individual self-interest leads to global efficient solutions.

Generally, a multi-agent coordination approach is a market-based approach if it satisfies the following requirements (Dias et al., 2006):

- The team is given a number of tasks that are achievable by individuals or sub-teams. To execute these tasks, the team has at its disposal a limited set of resources (robot capacities) that the team distributes among its members.
- A global objective quantifies the system designer's preferences for all possible solutions.
- An individual utility function specified for each agent quantifies that agent's preferences for its individual resource usage and contributions towards the team objective. Evaluating this function cannot require global or perfect information about the state of the team or team objective.
- A mapping is defined between the team objective function and individual or sub-team utilities. This mapping addresses how the individual production and consumption of resources and individuals' advancement of the team objective affect the overall solution.
- Resources and individual or sub-team objectives can be redistributed using a mechanism such as an auction.

The core of market-based approach can be observed from where the auction mechanism is. This mechanism can be divided in two phases, namely, a bidding phase and the winner determination phase. In the former, tasks are evaluated using a utility function, which does not require the use of global information. In the latter phase, after receiving the different bids, a task awarding mechanism is applied in order to choose the most suitable agent for the task under auction. Moreover, these two phases consider the participation of two roles: auctioneer and bidders. The bidding phase starts with either TI or GS offering a task to the rest of the bidders. After receiving the announcement, they should reply with their bids based on their capacity to execute that task (utility function). The bidding phase is finish when the auctioneer receives all the bids. Next, the winner determination phase starts. The auctioneer applies a mechanism that awards the task to one of the bidders. Finally, the winner will add the task to his/her execution list. Market-based task allocation algorithms do not limit the number of auctioneers and more than one can operate at the same time. The main concepts that define a task allocation mechanism based on auctions are: global objective, utility function, and task awarding mechanism.

The global objective defines the team's goal to be optimized by coordinating all agents. Different global objective functions can be considered (Tovey et al., 2005) described the sum of the utilities, the maximum of all the utilities, and the average of the utilities. The sum of utilities is used in scenarios where it is important to minimize the total energy consumed by the team of agents. The maximum of all the utilities is used in scenarios where it is fundamental to minimize the time needed to execute all tasks. Both objectives have been used in multi-agent exploration scenarios. On the other hand, the average of the utilities used in search-and-rescue scenarios where it is important to minimize how long on average it takes to execute a task.

The utility function is used to evaluate tasks and calculate bids. This function is composed of the reward and cost functions as indicated in Chapter 3. The reward function indicates the benefit of executing a task, and the cost function gives an estimate of the effort to accomplish the same task.

The most common task awarding mechanism is to allocate a task to the agent with the highest utility or lowest cost considering all received bids. As mentioned before, there is a connection between the individual utility function, the task awarding mechanism and the global objective. The system designer's responsibility is to choose a utility function, and an awarding mechanism that leads to an efficient global solution. Tovey et al., 2005 explained the systematic methods for deriving appropriate utility functions and awarding mechanisms for each of the global objectives.

Finally, other properties that allow for characterization of a market-based task allocation algorithm are described:

- Multiple Robot Single Task (MRST) algorithms and Multiple Robots Multiple Tasks (MRMT) algorithms: MRST algorithms do not make use of local execution plans, and therefore, they are suited for applications where task costs may change through time. However, the allocations are usually less efficient allocations than MRMT algorithms, which use local plans to increase the information used in the bid calculation. It can be said that MRST and

MRMT algorithms have a capacity constraint equal to one and greater than one respectively (Koenig et al., 2007).

- With and without reallocations: when reallocations are not considered, the same robots that initially allocated tasks execute tasks. On the other hand, when a task allocation algorithm considers reallocations, it means that in order to increase the efficiency of the final allocation, a robot could re-announce its already allocated task or tasks.
- Combinatorial or single-item auctions: in most of the task allocation algorithms, each auction process only considers a single task. In combinatorial auctions, each auction can involve more than one task. Therefore, bids are calculated for bundles of tasks (Zheng et al., 2007).
- Coordinated or loosely coupled tasks: when the execution of tasks is completely independent from the rest, this is termed loosely coupled. However, if the execution of tasks depends on others, tasks are coordinated. This fact should be taken into account in the task allocation algorithm in order to avoid execution deadlocks.
- Sequential and parallel auctions: when only one auction runs at a time, the task allocation algorithm is sequential. On the other hand, if more than one auction can be performed simultaneously, they are executed in parallel. When parallel auctions are used, the system's designer must be aware of the biding process since bids used in one auction process are no longer valid due to the result of another parallel auction.

2.2.5.3.1 M+ Approach - M+ (Botelho and Alami, 1999) could be, the first distributed market-based system defined within a general architecture for the cooperation among multiple robots. In this system, when a robot calculates the cost of a task, it considers one task ahead for each robot that allowed, whenever possible, an overlapping between the execution of the current task for a robot and the planning and task allocation of the next one, which increases the efficiency of the solution. In order to synchronize subtask execution, the M+ approach imposes soft time constraints on subtasks for a complex task. Costs are also associated with the quality of time-constraint satisfaction. In a multi-robot context, robots negotiate with one another to adapt its plan incrementally. Since each task has a different execution time, for future negotiations, agents optimize deviation for different execution times. Along the way, tasks can be moved from one UAV (agent) to another through negotiation.

2.2.5.3.2 MURDOCH Approach - MURDOCH is a general task allocation system based on principled, resource centric, published/subscribe communication model that makes extensive use of explicit inter-robot communication (Gerkey and Mataric, 2002). Therefore, Murdoch is a MRST task allocation algorithm, in which robots do not take part in auctions while they are executing a task. Therefore, a new task announced dynamically will be allocated to idle robots. If all robots are executing a task; the task is either discarded or re-announced after a period. Therefore, Murdoch appears as a version of Contract Net Protocol (CNP) of Smith (1980), which uses simple auctions to allocate tasks. Murdoch's approach is considered the first proven application of auction methods for the coordination of physical multi-robot systems that applied multiple tasks.

2.2.5.3.3 TraderBots Approach - TraderBots is price-based approach in which robots are considered as self-interested agents and the team of robots as an economy. Its goal is to complete tasks successfully while minimizing overall costs. The individual goal for each robot is to maximize its individual profit, which at the end will contribute to overall good. Robots have the ability to make plans and perform task assignment. Re-assignment is allowable and dependencies are taken into account.

CHAPTER 3

PROBLEM FORMULATION AND METHODOLOGY

3.1 Problem Formulation

In this research environment, multiple buyers and sellers exchange a single item at a time; the TI creates *n* sensor-visit tasks corresponding to *n* random UGSs and assigns them to GSs via auctioning. The GSs (and later on the UAVs) calculate the incremental cost for specific tasks and submit their sell (respectively, buy) bids to the TI (respectively, GS). The market system consists of multiple buyers and sellers that submit their bids for standardized units of well-defined items by stating the amount and the price they will trade, referred to as a "double auction." Each bidder expresses its subjective preference for the traded item using a utility function, which represents its estimate of its real cost to perform such a task.

Consider a set of *n* UGSs scattered in a remote area. A team of *m* UAVs designated to frequently visit these sensors collect their data and deliver it to the ground stations (GSs). Therefore, the objective of the coordination problem is to apply double auctions to reduce the overall cost while satisfying time constraints. The TI is located at the origin of a 3D-space bounded by the following ranges:

$-2\text{Comm}_{\text{UAVMax}} \leq x \leq 2\text{Comm}_{\text{UAVMax}}$

$-2\text{Comm}_{\text{UAVMax}} \leq y \leq 2\text{Comm}_{\text{UAVMax}}$, and

$0 \leq z \leq 2\text{Comm}_{\text{UAVMax}}$, where, $Comm_{UAVMax}$ is the maximum communication radius admissible by a UAV. The TI computes the cost and the deadline for each task as indicated by Equations (3.1) and (3.2).

$$TI_{TaskCost}(t) = P_0(1+u_p*t/t_H) \tag{3.1}$$

$$TIt_H(T_K) = 2*||T_k - TI||_2/UAV_i\_Speed + t \tag{3.2}$$

where, *t* is the time elapsed between the times the task is created until the time the task is re-auctioned, u_p is a parameter to determine the increment in the price, t_H is the deadline time for the task (T_K) to be received by the auctioneer, and UAV_i _Speed is the speed of the UAV_i , as also used in Equation (3.3).

$$P_{0=} 2*(|| T_k - TI||_2*(u_{dc} + u_{tcost}/UAV_i\_Speed)) \tag{3.3}$$

Ideally, as soon as a task is created it is auctioned immediately to the GSs. The TI broadcasts its tasks one at a time. Each GS broadcasts the announced task to the UAVs in its connectivity range and submits its sell bid paired with the minimum buy bid received from the UAVs to the TI. The TI ranks all received bids in descending order from maximum to minimum to generate a supply and demand profile. Then, the TI evaluates its received bids and assigns the task to the GS with the minimum sell bid, which is paired with a UAV's minimum buy bid. The GS that was paired with the winner UAV (UAV_{iWin}) prior to ranking the bids (denoted by GS_{jUAV}) will tell UAV_{iWin} to execute the task. The GS_{jUAV} will receive profit equal to the difference between its bid price and the announced winner GSs (denoted by GS_{jWin}) bid price. The winner UAV (UAV_{iWin}), and the winner GS (GS_{jWin}) will receive the profit calculated using Equation (3.4).

$$\text{Profit}_{GSi} = \text{Profit}_{UAV} = 0.5*((\underset{jWin}{GS}\_\text{SBid} - \underset{iWin}{UAV}\_\text{BBid}) + (\text{TI\_P}_0 - \underset{jWin}{GS}\_\text{SBid}) - (\underset{jUAV}{GS}\_\text{SBid} - \underset{jWin}{GS}\_\text{SBid})) \tag{3.4}$$

If at any time a UAV does not deliver the task to the original GS, a penalty is incurred that reduces its profit as in Equation (3.5).

$$\text{Penalty} = 0.5*(\underset{jUAV}{GS}\_\text{SBid} - \underset{jWin}{GS}\_\text{SBid}) \tag{3.5}$$

Further, the UAVs submit bids reflecting the incremental cost of the data-delivery task, and the GS allocates the task to the UAV submitting the minimum bid. This procedure is repeated until all tasks are sold. However, allocation of data delivery tasks to UAVs must continue to optimize the overall distance traveled and to satisfy the deadline time constraint on data delivery time as denoted by Equation (3.6).

$$T_{i+1}^{j} - T_{i}^{j} \leq t_H \qquad i \in N, \;\; 1 \leq j \leq n \tag{3.6}$$

Let T^j_i be the *ith* data delivery time for the jth sensor, t_H be the deadline time for data deliveries for any sensor, and *n* be the number of unattended ground sensors. In this case, we are dealing with a continuous task allocation mechanism that satisfies the constraint on data delivery time while optimizing the team's average distance traveled per data delivery.

To obtain a solution, we describe sensor-visit tasks. Each sensor-visit task consists of visiting a sensor, collecting its data, and returning to the GS to deliver the collected data. A task created or renewed is the task creation time. The data delivery time for a task is also the time passed from the task's creation time to the time the TI receives the task's data. By using the task creations and data delivery times, the problem statement reformulated as a problem of continuously allocating sensor-visit tasks to the UAVs so the data delivery times of all sensor-visit tasks remain lower than t_H.

3.2 Methodology

In this dynamic research environment, off-line methods are not appropriate because new UGSs maybe added to or removed from the mission. Further, the team of UAVs is prone to changes since we may lose some UAVs due to communication failure or death, and a UAV maybe added to the mission at any time. Additionally, continuously updating task allocations will produce more allocations that are efficient because allocations depend on the UAVs' positions when tasks are refreshed, and efficient allocations may change from round to round. Therefore, in this auction, the following assumptions are considered:

1. Tasks are created every two seconds.
2. Since the auction happen so quickly, UAVs wait until all tasks are auctioned before they start moving towards their tasks; otherwise, they hover around the GSs they communicate with.
3. The space is constrained to

   $-2Comm_{UAVMax} \leq x \leq 2Comm_{UAVMax},$

   $-2Comm_{UAVMax} \leq y \leq 2Comm_{UAVMax},$ and

   $0 \leq z \leq 2Comm_{UAVMax}.$
4. Three GSs are scattered around the TI and communicate with one another.

5. A UAV’s tasks are not delivered unless that UAV communicates with the GS it is supposed to deliver the tasks to.

A market-based approach uses communication efficiently since the UAVs compress information into bids. The role of the TI is limited to task creation and to holding auctions as well as matching sell bids with buy bids. In a double auction, each bidder has a private utility value for an item, which represents its real cost to perform such a task. In dealing with a minimization problem, we are trying to find a way to reduce the overall cost. Therefore, in this auction, the TI broadcasts its tasks one at a time, each GS broadcasts the announced task to UAVs in its connectivity range and submits its sell bid, paired with the minimum buy bid received from the UAVs, to the TI. GSs use the TI’s reservation price to generate their own price for the auctioned task, which is based on their linear distance to the auctioned task and the time needed for the task to be executed by a prospective UAV, which is also based on its known speed.

The UAV inserts its awarded task into its current plan where the task remains until the UAV delivers the corresponding data to the GS. When a GS receives the task's data, the TI’s information will be updated accordingly. Transaction determinations and winners (buyers and sellers) who are going to transact in the double auction are completely based on the bid price subject to the following constraints:

- The bid price must be less than or equal to the announced reservation price,
- A UAV that has a task from a previous auction cycle may submit a new buy bid if and only if executing the current bid will not make any previous won tasks in his task list time out.

The GS places a bid equal to the $\underset{j}{GS}_{\text{Cost}}$ for this task if and only if the constraint prescribed by Equation (3.7) is met.

$$\underset{j}{GS}_{\text{Cost}}(T_k) \leq TI_{\text{TaskCost}}(T_k) \tag{3.7}$$

Each GS auctions the task to all UAVs in its communication range. In other words, if the constraint prescribed by Equation (3.8) is met, a UAV qualifies to place a bid for that task from that GS.

$$\|UAV - GS\|_2 \leq CS_{UAV\text{-}GS}{*}(Comm_{UAV} + Comm_{GS}) \tag{3.8}$$

where $CS_{UAV\text{-}GS}$, in [0, 1], is the communication strength between the UAV and GS, $Comm_{UAV}$ is the communication radius of the UAV, and $Comm_{GS}$ is the communication radius of the GS.

3.3 Market-based Coordination Framework

UAVs, GSs and the TI trade tasks continuously via double auctions. The TI creates some tasks for bidding by the GSs. At this point, each GS starts broadcasting an availability message containing its ID to determine available agents within its communication range. When a GS detects a UAV inside its communication range, it sets an available flag for that UAV and adds it to its auction list. A GS offers its tasks only to members within its communication range. When a GS immediately detects an available agent within its communication range, auctioning of its task starts. Each UAV sends a bid representing its most profitable deal to the source GS. A GS, as auctioneer, evaluates all received bids, and sends its cost accompanied by the minimum buy bid to the TI.

3.4 Policy of the Market

Based on the description of our market given in Section (3.3), the market policy is introduced as follows:

- The accepting policy states that, in order for an incoming bid to be accepted, it must be less than or equal to the reservation price announced by the auctioneer. The purpose of such policy is to maintain a successful rate of transactions, and to signal to traders the current market prices.
- The matching policy defines how to match a buy bid with a sell bid. For any auctioned task, the minimum sell bid will be matched with the minimum buy bid.
- The clearing policy determines what matched bids are being executed.
- The clearing price will be equal to the won bid (M^{th}) price.

For any GS that auctions a specific task that it has paired with a UAV offering a minimum bid, the bids from all the GS-UAV pairs are ranked in ascending order of magnitude. Suppose the number of GSs submitting bids is M. Counting from the top ranked bids, the value in the M^{th} position is the clearing price that task is sold for, and the that GS and the UAV making the bid becomes the seller and the buyer (winner), respectively, for that task. Then the UAV delivers the task to that GS. A task not sold during the auction round is re-auctioned at a higher cost as determined by Equation (3.1). This procedure is repeated until all tasks are sold.

Generally, the TI can determine 'what' and 'how' an incoming bid is transacted. Briefly, we can say, for a given set of incoming orders, that the accepting policy determines what bids are to be accepted. The matching policy determines whose bid can be matched with whom, and the clearing policy specifies the transaction that should be executed.

Transaction determinations and winners (buyers and sellers) that are going to transact in double auctions have two issues that need to be well defined. For instance, given the buy and sell bids for the following example, which of these is going to transact and at what clearing price? Which will lead us to the M^{th} and the $(M+1)^{st}$ price?

3.4.1 M^{th} and $(M+1)^{st}$ Price Rules

Let X denotes the set of all buy and sells bids for a single task; M of these bids are the sell offers, and N represents the buy offers. The M^{th} price rule sets the clearing price at the M^{th} lowest price among all X bids. The $(M+1)^{st}$ price rule sets the clearing price at the $(M+1)^{st}$ lowest price among all X bids. In order to determine the bids that are going to be transacted, the transaction set proceeds as follows:

> While the lowest remaining buy bid is less than or equal to the lowest sell bid, remove these bids from the set of outstanding bids and add them to the set of matched bids (transaction set).

Note that the M^{th} price is undefined if there are no sellers, and the $(M+1)^{st}$ price is undefined if there are no buyers.

Consider the set of bids in the double auction shown in Figure 3.1. The number of total bids is X = 6, of which M (number of sell offers) = 3 and N (number of buy offers) is X – M = 3. The M^{th} clearing price is the M^{th} bid among all submitted bids while the $(M+1)^{st}$ price is the $(M+1)^{st}$ bid among all bids X = {30, 27, 25, 24, 18, 15}.

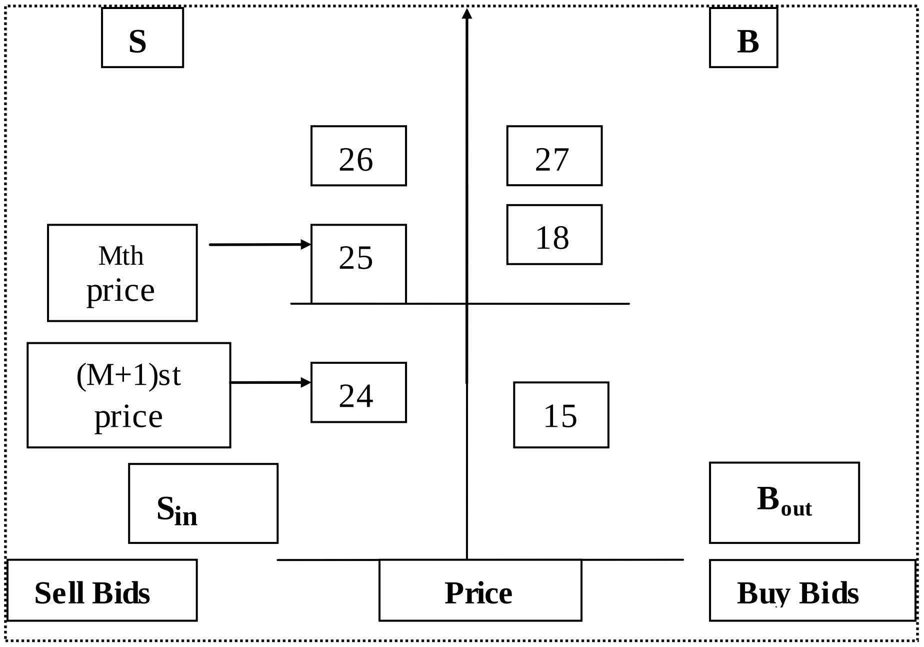

Figure 3.1. Schematic of bids in double auction with Mth and (M+1)st price

To determine the transaction set, the lowest buy bid is matched with the lowest sell bid, providing the constraints in Section 3.2 are met. This process continues until the buy bid is higher than the sell bid. The transaction set will be {(15, 24), (18, 25)}. Matched bids are removed from the outstanding bids and placed in the matched bids profile where the lowest sell bid and the lowest buy bid are transacted. For instance, the sell bid 25 cannot be transacted since only one task will be sold, which is the lowest buy and sell bids. The transaction price can either be set at 25 (the Mth) or at 24 (the (M+1)st) price.

For instance, take the first set of bids in the transaction set (24, 15). If the M^{th} price is used, then each buyer and seller will make a profit equal to ½*((24-15) + (30-24)), which is \$7.5, assuming the TI's reservation price (P_0) is \$30, using Equation (3.4).

3.4.2 Auction Structure

This implementation consists of two different kinds of auctions:

1. Auctions held by the TI, and
2. Auctions held by the GSs

3.4.2.1 Combined Auction Procedures

The auction proceeds as follows; see also the flow diagram in Figure 3.2:

1. ***Task Announcement*:** Each auction starts with an offer message sent by the auctioneer (TI) to all GSs (TI → GSs). The message contains the task's id, sensor location, task's creation times, task's deadline, and task's prices.
2. ***GS Call for Bid*:** Upon receipt of the offer message from the TI, each GS broadcasts the task and calculates its cost for that task.
3. ***GS's Buyers Bid Evaluation:*** Each GS evaluates the buy bids received from the UAVs for validity according to the accepting policy and chooses a bid with the minimum price.
4. ***Bid Submission:*** Each GS submits its bid with the winner UAV's buy bid to the TI.
5. ***Matching Result*:** The auctioneer (TI) evaluates all received bids and finds the one with the most profit. Then it matches a GS with the minimum sell bid with the ID of the minimum buy bid and the clearing price.
6. ***Win Confirmation*:** Each winner (GS or UAV) receives the result and sends a confirmation message to the auctioneer indicating notification about the result.
7. ***Offer End*:** The auctioneer sends an Offer End message to the auction participants

when it successfully receives the Win Confirmation message and closes the auction.

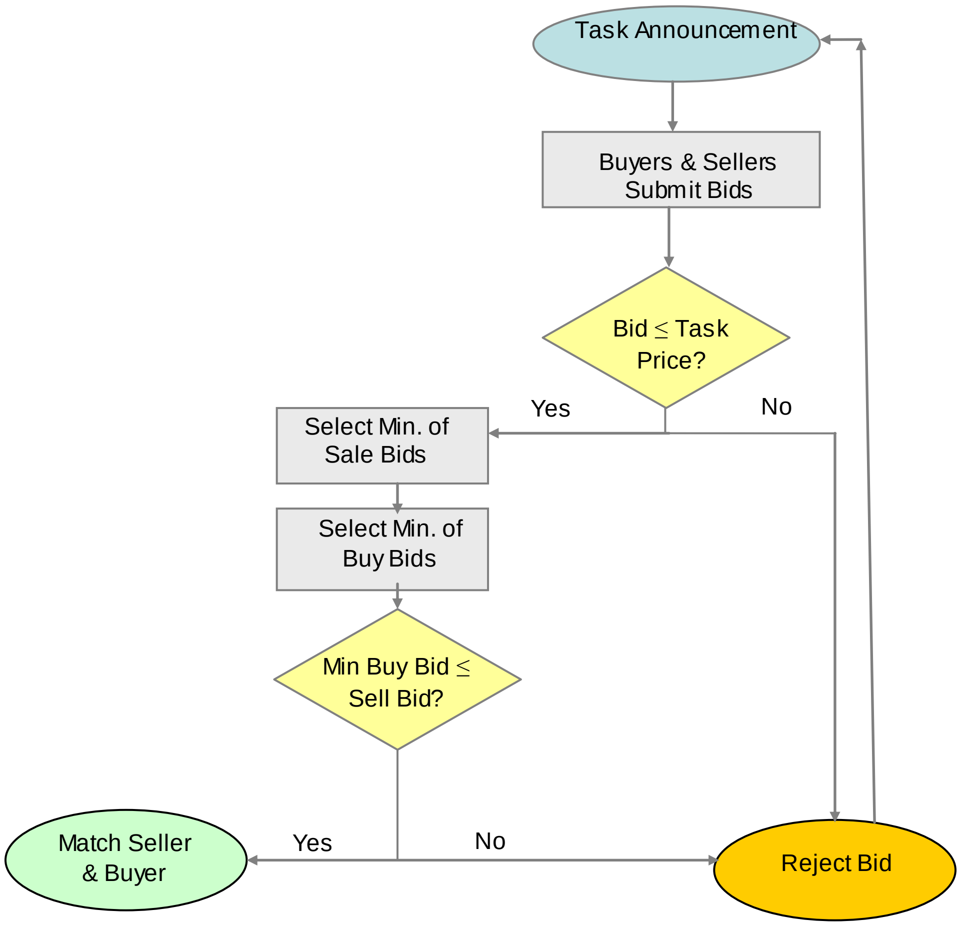

Figure 3.2. Flow diagram for the market policy

3.4.3 Cost Estimation

As previously mentioned, tasks are traded between the TI, GSs and UAVs. In order to define the allocation problem for such environments, it is necessary to specify the cost functions, as given by Equation (3.9).

$$P(T^i, GS_k) = R(T^i, GS_k) - C(T^i, GS_k) \qquad (3.9)$$

where P is the profit generated by the ground station GS_k by accepting the task T^i and R and C are revenue and cost functions, respectively. The revenue function indicates the benefit of executing a task, and the cost function provides an estimate of the cost to accomplish the same task. In this dissertation, rewards associated with tasks are not considered; therefore, the utility functions equal the cost of the tasks. Further, tasks are waypoints, and costs define an amount that reflects the distance between each GS or UAV and the location of interest, such as the traveled Euclidean distance.

The global objective of the task allocation algorithm is to minimize overall costs. An important term used in the following chapters is global cost, which is the sum of the allocated task costs. Therefore, the global objective used for this dissertation is the minimization of the global cost. The multi-ground station task allocation problem stated in terms of global costs is as follows:

Given a set of tasks, $T = \{T_1, T_2, ..., T_t\}$, a set of GSs $\{GS_1, GS_2, ..., GS_r\}$, and a function $P(T^i, GS_i)$ that specifies the cost of executing a subset T^i of the set of tasks T by GS_i, find the allocation of tasks to GSs that minimizes the global cost as given by Equation (3.10).

$$\sum_{i=1}^{r} P(T^i, GS_i) \qquad (3.10)$$

where r is the number of GSs and the subset of tasks T^j is assigned to GS_j.

The TI issues and renews tasks, and each GS submits bids for the newly issued tasks; a task is assigned to the GS with the minimum bid. The TI's algorithm for this allocation is as follows:

```
if a task is created then
    announce task
    while timer is running do
        receive bids
    end while
    calculate best bid
    match buyer with seller
    award task to best match
    remove task from announcement list
end if
```

For each auction cycle, there is only one awarded task. Upon winning a task, the winning GS broadcasts the same task(s) to the UAVs within its connectivity range. The UAVs then calculate the cost for adding the new task to their current plan. Then, the difference between the two plans (current plan and old plan) is the cost for UAV_i to execute the auctioned task.

The algorithm for the GS task allocation is as follows:

```
if a task-list is not empty then
    announce task
    while timer is running do
        calculate cost
        receive bids
    end while
    calculate best bid
    send cost and best bidder
    remove task from task-list
end if
```

When a GS assigns a task to the winning UAV, it keeps the winner's id, task id, winner's cost, and the task due time in different lists to maintain control of its awarded task(s).

In this problem, two types of costs contribute to the marginal cost of adding a new task to a plan:

1. **The distance cost** is the cost due to the additional distance that the UAV should travel, and
2. **The time cost** is the cost due to latency that performing this task will cause in the data delivery time of the other tasks already in the plan.

3.4.4 GSs Cost Estimation

In the set up simulated in this work, there are three GSs located 120˚ apart from each other around the TI. The GSs communicate with the TI all the time. When a task is created at the TI level, each GS submits a bid for that task. The bid is based on the linear distance to that task and uses the task's information provided by the TI (i.e., id, price, location, and creation time), which has a different price for each task. First, the TI computes the cost and the deadline time for that task according to Equations (3.11) and (3.12):

$$TI_{TaskCost}(t) = P_0(1+u_p*t/t_H) \tag{3.11}$$

where t_H is the deadline time for task T_K to be received by the auctioneer, as given by Equation (3.12).

$$t_H(T_K) = 2*\|T_k - TI\|_2/UAV_i\_Speed + t \tag{3.12}$$

where t is the time elapsed from the time the task is created until the time the task is re-auctioned, u_p is a parameter to determine increments in the price, t_H is the deadline time for the task T_K to be received by the auctioneer, u_{dc} is the distance unit cost, and UAV_i_Speed is the speed of UAV_i, as used in Equation (3.13).

$$P_0(T_k) = 2*\| T_k - TI\|_2*(u_{dc} + u_{tcost}/UAV_i\_Speed)) \tag{3.13}$$

By the time the task reaches its deadline, if u_p is chosen to equal 1, the price of the task will have doubled since its creation time. This will motivate the GSs to bid for the task that was not profitable during previous auction cycles. When a task T_k is created at the TI level, all GSs will use the same TI's price function to generate a new bid for the auctioned task. The GS will place a bid equal to $GS_{TaskCost}$ for this task if and only if the constraint prescribed by Equation (3.14) is met.

$$GS_{TaskCost} \leq TI_{TaskCost}(t) \tag{3.14}$$

Therefore, the GS's bid will be the GS's linear distance to T_k and back; in addition to the time the UAV needs to execute the task.

First, the GSs calculate their cost as given by Equation (3.15).

$$GS_{j\,distCost} = 2*(\|dist(GS_j, T_k)\|_2*u_{dc} \qquad 1 \leq j \leq 3, \tag{3.15}$$

where T_k is a new auctioned task to be added to GS_j current plan, and u_{dc} is the unit distance cost. Second, since the speed of the UAVs is known to the GSs, any GS_j can predict the execution time for the newly auctioned task. Therefore, the time cost to execute a new auctioned task is calculated using Equation (3.16).

$$\underset{j}{GS}_{\text{timeCost}} = 2*(\underset{j}{GS}_{\text{dist}} / \text{Speed}_{\text{UAV}})*u_{\text{tcost}}, \tag{3.16}$$

where u_{tcost} is the time unit cost, which is known to all GSs, and $\underset{j}{GS}_{\text{timeCost}}$ is the cost of the time for $\underset{j}{GS}$ to receive the task's data, which is based on the task's distance to $\underset{j}{GS}$, GS_{dist}, and the UAV's speed, $\text{Speed}_{\text{UAV}}$. The sum of Equations (3.15) and (3.16) yields the GS's estimated cost, as given by Equation (3.17).

$$\underset{j}{GS}_{\text{Cost}} = \underset{j}{GS}_{\text{distCost}} + \underset{j}{GS}_{\text{timeCost}} \tag{3.17}$$

And the GSs bids were calculated using Equation (3.18).

$$\underset{j}{GS}_{\text{Bid}} = \underset{j}{GS}_{\text{Cost}} \tag{3.18}$$

Since a GS bids according to its true valuation for a task, a bid in Equation (3.18) will be the actual cost of the auctioned task to the GS that will submit it to the TI. P_0 is a new term that the GS will use to determine the price of task(s) later. Therefore, the GS will use the P_0 from the TI, as given by Equation (3.3), in order to generate its cost for newly auctioned tasks.

As previously mentioned, the auctioneer uses a time-increasing function P(t) for assigning prices to tasks. In order for any seller to maximize all profit made, the seller must sell tasks won as soon as possible. Further, a GS is not assured of selling a task within a fixed time because bidders will only bid when assured a profit. This is the main reason for using the increasing varying price function, which encourages buyers to bid for unsold tasks in the near future. Therefore, the GS will make the task price function an increasing function of time, which will guarantee that UAVs will be encouraged to buy tasks that were not profitable to them during previous auction cycles. The price function is given by Equation (3.19).

$$GS\_P(t) = P_0*(1+u_p*t/t_H) \tag{3.19}$$

The price function also motivates bidders to bid on those tasks for which more time has passed since their creation and so have become more profitable. Clearly, the most profitable task is not necessarily the task with the lowest cost. Ajorlou et al. (2007) suggested that, since price is time varying and profit is price minus cost, this time varying function will have an important effect on the balance between a task's cost and the importance of a task in the task allocation process. Assume that the price of task T_k, offered by the TI at time t is $P(t - t_T)$, where, t_T is the creation time of T_k. The difference in price, with regard to time, should be large enough to overcome the extra cost that an expensive important task may have compared to other offered tasks, which can improve the performance by decreasing the probability of successive timeouts for a given task.

Clearly, the auctioneer associates a price with each offered task, and upon appropriate completion of the task, it pays revenue equal to the task's price to the agent that performed the task. Agents' bids reflect the profit they can make by accepting and performing tasks. In this case, the GSs and UAVs share the profit evenly after task completion. Equation 3.4 is presented again here as Equation (3.20) for clarity.

$$\text{Profit}_{GS} = \text{Profit}_{UAV} = 0.5*((\underset{jWin}{GS}\_SBid - \underset{iWin}{UAV}\_BBid)+ (TI\_P_0- \underset{jWin}{GS}\_SBid)- (\underset{jUAV}{GS}\_SBid - \underset{jWin}{GS}\_SBid)) \tag{3.20}$$

Adding any task to the bidder's current plan will result in an additional distance an executing agent needs to travel, and the delivery time of tasks will be pushed back, as expressed by Equation (3.21).

$$R(t) = \begin{cases} P0 & t \le t_H \\ 0 & otherwise \end{cases} \tag{3.21}$$

where t is the elapsed time since the task creation and P_0 is the price of the task. Note that P_0 may be different for different tasks and is equal to the price the auctioneer announced while selling that task. If a GS offers a task, its price is determined by the time-varying price function, as in Equation (3.11). As previously mentioned, for any auctioned task, constraints must be satisfied in order for the task bid to be accepted.

The auctioneer, TI, knows the price function and uses it to calculate the price announced to the GS when offering the task. However, the revenue that a UAV will receive upon performing a task depends on both the time when it accepts the task and the time when it delivers the data. Clearly, any GS will receive its revenue for any task it won and executed in a time not exceeding the task's deadline time.

3.4.5 UAVs Cost Estimation

The initial locations of the UAVs are generated randomly such that each UAV communicates with at least two GSs. In this dissertation, a given UAV_i can participate in an auction with a given GS_j if and only if UAV_i is within GS_j's range and GS_j is within UAV_i 's range. Whichever range is smaller will determine how close UAV_i must be to GS_j for them to participate together. We make the simplifying assumptions that all UAVs and GSs have the same range. Call the area around a GS_j within which it can carry on an auction with a UAV its *domain*; a UAV's *domain* is defined similarly but with respect to a GS. Whether the UAV's range is smaller than the GS's range or vice versa, the domains of all GSs will be the same size and shape.

An interesting case is where the GSs' domains overlap. When the TI announces its task(s), each GS checks for availability of UAVs before announcing a task. When a GS receives a response from any UAV in its connectivity range, the GS starts to broadcast the announced task. The UAV should deliver the data to the GS for which it sold the corresponding data-delivery task because that GS will count on that UAV returning it within a certain time. As previously stated, if it delivers the data to another GS, it incurs a large penalty. Recall that Equation (3.5) is

$$\text{Penalty} = 0.5*(GS_{jUAV}\text{\_SBid} - GS_{jWin}\text{\_SBid})$$

Upon receipt of a task announcement, the UAV computes its first bid and the estimated time taken to deliver the task to the GS according to Equations (3.22) and (3.23).

$$\text{UAV}_{\text{TaskCost}} = \text{Dist}_{\text{UAV-Tk}}*u_{\text{dcost}} + \text{Dist}_{\text{UAV-Tk}}*u_{\text{tcost}}/*UAV_i\text{\_Speed} \tag{3.22}$$

$$UAV_i\ t_H = \text{Dist}_{\text{UAV-Tk}} / UAV_i\text{\_Speed} \tag{3.23}$$

Then, the UAV places a bid for an amount equal to its cost to that GS for the task if the constraints prescribed by Equations (3.24) and (3.25) are met.

$$\underset{i}{UAV}\ _{TaskCost}(T_k) \leq TI_{TaskCost}(t) \text{ and} \tag{3.24}$$

$$\underset{i}{UAV}\ t_H(T_K) \leq TIt_H(T_K) - t_{UAVstart} \tag{3.25}$$

where $t_{UAVstart}$ is the time the UAV starts executing the task in its task list.

If the UAV already has at least one task in its task list, then calculating the marginal cost is NP hard and requires re-planning for a new set of tasks. For simplicity, we use a heuristic in which we inserted the new task in all possible positions in the current plan and chose the one that minimized the distance cost of the new plan. In addition to Equation (3.26), the constraints denoted by Equations (3.27) and (3.28) have to be met.

$$TIt_{Hnew} \leq TIt_{Hold} \tag{3.26}$$

$$t_{UAVstart} + t_{PathNew} \leq TIt_{Hnew} \tag{3.27}$$

$$UAV_{TaskCostExtra} \leq TI_{TaskCost}(T_K) \tag{3.28}$$

where

$$UAV_{TaskCostExtra} = UAV_{CostCurrentPath} - UAV_{CostPreviousPath}.$$

TIt_{Hold} and TIt_{Hnew} are the task deadlines for the immediate previous and the new task to bid for, respectively; $UAV_{TaskCostExtra}$ is the extra cost for adding on an additional task. If all these constraints are met, then the UAV places a bid for the new task according to Equation (3.29).

$$UAV_{TaskCost}(T_K) \quad = UAV_{TaskCostExtra} \tag{3.29}$$

3.5 Robustness

3.5.1 UAV Malfunction

In this dynamic research environment of market-based mechanisms, recall that, in order for a UAV to submit a bid, the UAV must first is able to deliver the task on time, meaning that the UAV's contract is time-limited, and the UAV is responsible for delivering its won task in a timely fashion. During task execution, in the event a UAV malfunction, the tasks may not be delivered as desired. All undelivered tasks are re-auctioned to existing UAVs when the UAV never recovers to complete the task in its task list. To plan for any uncertainties, undelivered tasks from a disconnected UAV are not immediately re-auctioned. First, the GS allocates an extended time. Then, after this time has elapsed, undelivered tasks are re-auctioned. This extended time is computed using Equation (3.30).

$$t_{Reauction} = (1+\gamma) * \min(TIt_{H}\text{-}UAV_i) \; 0 \leq \gamma \leq 1 \tag{3.30}$$

where $TIt_{H\text{-}}UAV_i$ contains task deadlines for all the tasks that are in the task list for a specific UAV_i .

Since the route information for each UAV is known, the location of the UAV can easily be found. Since the re-auction happens very frequently (every 2 secs), all UAVs hover around their current locations until announced re-auctioned tasks are sold. A GS will assume a UAV malfunction if a certain amount of time has passed since task(s) deadline has elapsed and the GS did not yet receive their corresponding data. γ is a constant number chosen between zero and one, which is multiplied by the deadline to allow latency in delivering data due to possible estimation errors. The task deadline triggers a UAV malfunction detection and recovery. When a GS experiences a situation where needed task data was not received for a period of at least equal to the threshold time, it auctions all tasks won by the malfunctioned UAV. Additionally, the fact that the price is an increasing function of time accelerates the recovery process by providing more profit for timed out tasks compared to the corresponding profit of its regular price.

When the re-auction begins, all the remaining tasks (undelivered tasks) from each UAV become the previous task list. Consequently, the route starting from the current location of that UAV becomes the immediate previous route. As a result, the procedures described for GS and UAV bidding for a new task still apply except the new task deadline for any re-auctioned task is determined by Equation (3.31).

$$TIt_{Hnewj} = TIt_{Holdj} + t_{jReauction} - \max(t_{Malfunctioned}, t_{UAVstart}) \tag{3.31}$$

where

TIt_{Hnewj} is the new task deadline for a given task j,

TIt_{Holdj} is the old task deadline for a given task j,

$t_{Malfunctioned}$ is the time at which the UAV get malfunctioned, and

$t_{jReauction} \geq t_{Reauction}$ is the time at which task j is re-auctioned.

3.5.2 Communication Failure

A UAV might also fail to communicate during the auction process. Frequent auctioning allows the UAV to regain its ground and is an essential part of this coordination method designed to accelerate the recovery of such a problem. As soon as a UAV recovers from the communication failure by being able to communicate with any GS, the UAV will start observing the auction rules and participate in new auctions.

Dias et al. (2003) suggested the following strategies to improve robustness:

- monitoring the communication connectivity to robots that have subcontracted tasks,
- frequent auctioning and bidding, which help reallocate tasks among robots more efficiently,
- the absence of assumptions that all agents will participate in any auction, and
- continuous scheduling of assigned tasks for execution as tasks are completed.

CHAPTER 4

MULTI-GROUND STATIONS TASK ALLOCATION WITH DOUBLE AUCTION

This chapter studies the performance of market-based coordination methods to demonstrate how a double auction can influence the quality of the solution. Double auction consists of multiple buyers and sellers participating to trade a commodity. Each buyer and seller submits a bid representing its offer to buy or sell the auctioned commodity. Submitted bids are matched, and the auction is cleared thereafter. A double auction, then, is a two-sided auction. One side represents a centralized approach while the other represents a distributed approach. This enables the market to compute the information in an efficient manner while providing quicker responses.

This study investigated the effect of increasing the UAVs' communication ranges on the distance travelled, and the time of task delivery using a different number of tasks while the UAVs participated in double auction coordination. Therefore, the global objective used in this dissertation is the minimization of the global cost (distance travelled).

4.1 Market Setup

The market setup consists of an allocation of a set T of tasks among a set of GSs partitioning T among the GSs, where $T = \{T_1, T_2, \ldots, T_n\}$ and GSs = $\{ GS_1, GS_2, \ldots, GS_m\}$. This is denoted by a tuple $[T^1, T^2, \ldots, T^{n-1}, T^n]$ where:

- Each subset of tuples represents tasks assigned to a GS, i.e., a ground station GS_i is assigned the tasks represented by $T^i = \{Ta, Tb, \dots, Tt\}$, which is a subset of the set T.
- The union of the sets of tasks in the tuple is equal to the complete set of tasks, i.e.,

  $T^1 \cup T^k \cup \dots \cup T^n = T$
- The sets in the tuple are pairwise disjoint, i.e., $T^1 \cap T^k = Ø$; for all $i \neq k$.

The purpose of using a task allocation algorithm is to minimize overall costs, which are defined as the sum of the allocated task costs. Hence, a task allocation problem for multiple GSs can be stated as follows:

Given a set of tasks, $T = [T_1, T_2, \dots, T_n]$, a set of GSs = $[GS_1, GS_2, \dots, GS_m]$, and a function $P(T^i, GS_k)$ that specifies the utility of executing a subset of tasks T^i by GS_i, $i = 1, 2, \dots, n$, find the allocation of tasks to GS_i that optimizes the overall objective.

Tasks are issued by the TI to the GSs for bidding, and allocated to the GSs with the minimum bid. In defining the allocation problem, it is important to specify the cost function given in Chapter 3, Equation 3.2, repeated here as Equation 4.1.

$$P(T^i, GS_k) = R(T^i, GS_k) - C(T^i, GS_k), \tag{4.1}$$

where P is the profit generated by the ground station GS_k for accepting tasks T^i, and R and C are the revenue and cost functions, respectively. The revenue function, $R(T^i, GS_k)$, represents the cost benefit to GS_k for executing task T_i, and the cost function, $C(T^i, GS_k)$, represents a cost estimate for GS_k for accomplishing the same task. Tasks are waypoints, and costs are numbers that reflect the distance between each GS and a waypoint of interest, such as the Euclidean travelled distance.

4.2 Simulation Setup

This section describes the simulation environment that consists of one TI, three GSs, and a different number of UAVs. The TI, GSs, and UAVs interact over tasks issued by the TI. The TI is located at the origin of a 3D-space bounded by the following ranges:

$-2\mathrm{Comm}_{\mathrm{UAVMax}} \leq x \leq 2\mathrm{Comm}_{\mathrm{UAVMax}},$

$-2\mathrm{Comm}_{\mathrm{UAVMax}} \leq y \leq 2\mathrm{Comm}_{\mathrm{UAVMax}}$, and

$0 \leq z \leq 2\ \mathrm{Comm}_{\mathrm{UAVMax}}$,

where $\mathrm{Comm}_{\mathrm{UAVMax}}$ is the maximum communication radius admissible by a UAV during the design of that UAV, and x, y, and z are the coordinate axis.

As shown in Figure 4.1, the three GSs are located 120˚ apart from each other around the TI, and can communicate with the TI at all times. The initial locations of the UAVs are generated randomly such that each UAV communicates with at least two GSs. A given UAV_i can participate with a given GS_j if and only if UAV_i is within GS_j 's range and GS_j is within UAV_i 's range. Whichever range is smallest determines how close UAV_i must be to GS_j for them to participate together. The simplifying assumption is that all UAVs and GSs have the same range. The area around GS_j in which it can carry on an auction with a UAV_i is its *domain*. Whether the UAV's range is smaller than the GS's range or vice versa, the domains of all GSs are the same size and shape.

The TI's and GSs' communication ranges are 100 meters, and 200 meters, respectively. Any GS has at least one UAV within its communication range before the start of the initial auction. Any UAV can communicate with at least two GSs at the beginning of the auction. To enhance the design's quality of solution, the UAVs' communication ranges fell between 250 meters to 2,500 meters, and their speed is 10 m/s. The unit distance cost (u_{dcost}) is set at $0.1 / m, the time unit cost (t_{ucost}) at $0.1 /s, and u_p, a parameter to determine an increase in price, is set at 1. Each task has a different value, which is determined by its distance from the TI. This implies that, using the time varying price function, the price for a task will double when it reaches its deadline, t_H, from its creation time and has not been sold.

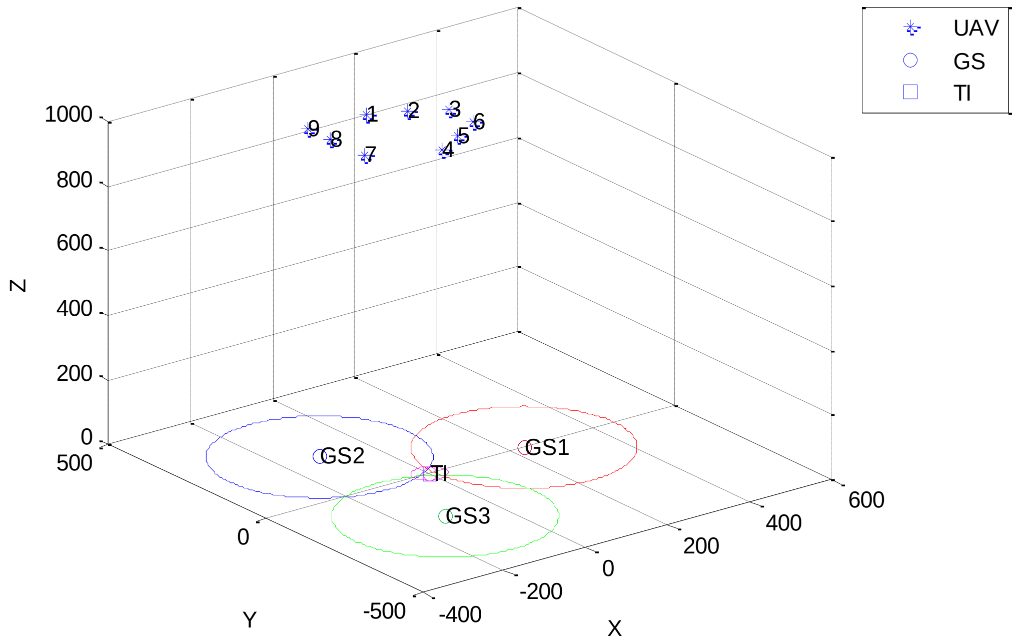

Figure 4.1. Setup of TI, three GSs and nine UAVs for double auction

Each GS submits bids for newly issued tasks. A task is assigned to a pair of GSs and the UAV with the minimum bid. A task is an act of visiting and collecting data from a specific sensor location by a UAV and delivering the collected data to the TI through a

GS later. In a practical sense, a TI could be a control room or any agent (including humans) with the ability to initiate and re-allocate tasks to the fittest pair of GSs and UAVs through double auction. In this scenario, the UAVs have the ability to buy from more than one GS and deliver tasks to a source GS.

4.3 Interaction between TI and GSs

This section describes the interaction between the TI and GSs with emphasis on optimizing task allocation in double auction. Presented are the results of five runs with relative statistics and associated profits.

4.3.1 Double Auction between GSs and UAVs

All GSs compete as self-interested agents to maximize their profits, and the UAVs aim to minimize their overall distance travelled in order to execute tasks. The distance cost of adding a new task to any GS's current plan is double the linear distance from the GS to the task location in addition to the time needed for task(s) execution. Since the UAVs' speed is known, the GSs can predict the time cost for an auctioned task. Based on this information, the GSs submit their bids to the TI (refer to Chapter 3).

Task performance is measured using the averages of at least five runs for each communication range. Initially, Tables 4.1 through 4.5 summarizes the results for each run for the GSs and UAVs, and the profits each made during the auction cycles. Table 4.1 presents the data for twelve random tasks for six UAVs at 250 meters and the transactions between the UAVs and GSs for exchanging task(s) during each run. Randomly generated tasks do not necessarily mean tasks will be bought by the same

UAV from the previous run. Instead, the cost of any task might change during each run depending on how far the task is from the UAV, and the time needed for execution.

Table 4.1. Run 1: Profit generated by GSs and six UAVs while auctioning 12 random tasksat 250 meters

Task	TI Price	Seller (GS) Bid	Seller(GS) ID	Buyer (UAV) ID	Buyer (UAV) Bid	Buyer/Seller Profits
T1	$1117	$1084	2	2	$1023	$47
T2	$310	$268	1	4	$215	$48
T3	$1271	$1233	2	1	$1172	$50
T4	$1136	$1107	2	1	$368	$384
T5	$1502	$1466	1	3	$1410	$46
T6	$832	$799	1	3	$336	$248
T7	$842	$819	1	1	$410	$216
T8	$508	$460	3	1	$216	$146
T9	$630	$612	3	2	$167	$231
T10	$1318	$1289	3	5	$1215	$51
T11	$1235	$1202	3	5	$277	$479
T12	$1702	$1664	2	6	$1616	$43

Since delivery cost is assumed to be at the intersection of the UAVs' communication range and the target sensor location, tasks executed by each UAV might also change from one run to the next, depending on whether the task(s) on the UAV's list permits executing a newly auctioned task without making any of its previous won task(s) timeout. The profit each UAV and GS generates during a double auction relies on how well the UAVs are positioned during the auctioned task with respect to the task's location and the UAV's communication ranges. The program ran for at least five times to generate profit, as denoted by Equation 4.2.

$$GS_j\_average\_profit = Average(\sum_{i=1}^{m} UAVi\_Profit) \quad 1 \le j \le 3 \tag{4.2}$$

where m is the number of UAVs that paired with GS_j during the auction and GS_j *_average_profit* is the average profit made by that specific GS. The data in Tables 4.2 through 4.5 show the profit for each GS and a list of corresponding UAVs for runs 2 through 5.

Table 4.2. Run 2: Profit generated by GSs and six UAVs while auctioning 12 random tasks at 250 meters

Seller (GS) ID	GS Total Profit	GS Task List	Buyer (UAV) ID
1	$462	4, 5, 11, 12	1, 2, 4, 6
2	$255	3, 6	1, 2
3	$873	1, 2 ,7, 8, 9, 10	1, 4, 5, 6

Table 4.3. Run 3: Profit generated by GSs and six UAVs while auctioning 12 random tasks at 250 meters

Seller (GS) ID	GS Total Profit	GS Task List	Buyer (UAV) ID
1	$736	1, 7, 8, 9	1, 3, 4
2	$1578	2, 3, 4, 5, 6, 11, 12	2, 4, 1, 5
3	$444	10	1

Table 4.4. Run 4: Profit generated by GSs and six UAVs while auctioning 12 random tasks at 250 meters

Seller (GS) ID	GS Total Profit	GS Task List	Buyer (UAV) ID
1	$1258	2, 3, 6, 9, 10	3, 4, 5
2	$935	7, 8, 11, 12	3, 4
3	$648	1, 4, 5	3, 5

Table 4.5. Run 5: Profit generated by GSs and six UAVs while auctioning 12 random tasks at 250 meters

Seller (GS) ID	GS Total Profit	GS Task List	Buyer (UAV) ID
1	$606	3, 5, 6, 10, 12	2, 3, 4
2	$202	1, 7	1, 4
3	$809	2, 4, 8, 9, 11	1, 2, 5, 6

4.3.2 Results for increasing UAVs' Communication Range

The impact of increasing the UAVs' communication ranges was investigated to enhance the quality of the solution for each GS. Profits are divided evenly between participating GSs and UAVs. Therefore, each GS's profit influenced the total distance travelled by the UAVs on the GS's / UAV list and the total execution time for its task(s), which affected the UAV's ability to reach and execute tasks at an earlier time. Further, since all UAVs' speed is constant, the only factor that affected a task's reachability is their communication ranges.

The average GSs profit generated from these runs is the profit made by all UAVs on its UAV list. The profit for the GSs at each communication range during the five runs was summed to give the average profit. As shown in Table 4.6, each GS's profit continued to increase as the UAVs' communication ranges increased. Because the UAVs share profits evenly with their GSs, they also benefitted from increasing their communication range.

As depicted in Table 4.6 and Figure 4.2, increasing the UAVs' communication range affected not only the profit each buyer and seller generated, but also the task's execution time as well. The UAV's constant speed contributed to the decrease in the distance travelled to execute a task, which was a direct result of increasing the UAV's communication range. Thus, the decrease of distance travelled automatically affected the task execution time.

Table 4.6. Average GSs profit while auctioning five sets of 12 random tasks to six UAVs

UAVs Communication Range (m)	250	500	750	1000	1250	1500	1750	2000	2250	2500
GS1	$724	$749	$774	$801	$837	$961	$981	$1070	$1081	$1101
GS2	$699	$719	$739	$758	$917	$940	$962	$984	$1001	$1025
GS3	$736	$754	$771	$788	$791	$876	$898	$903	$939	$959

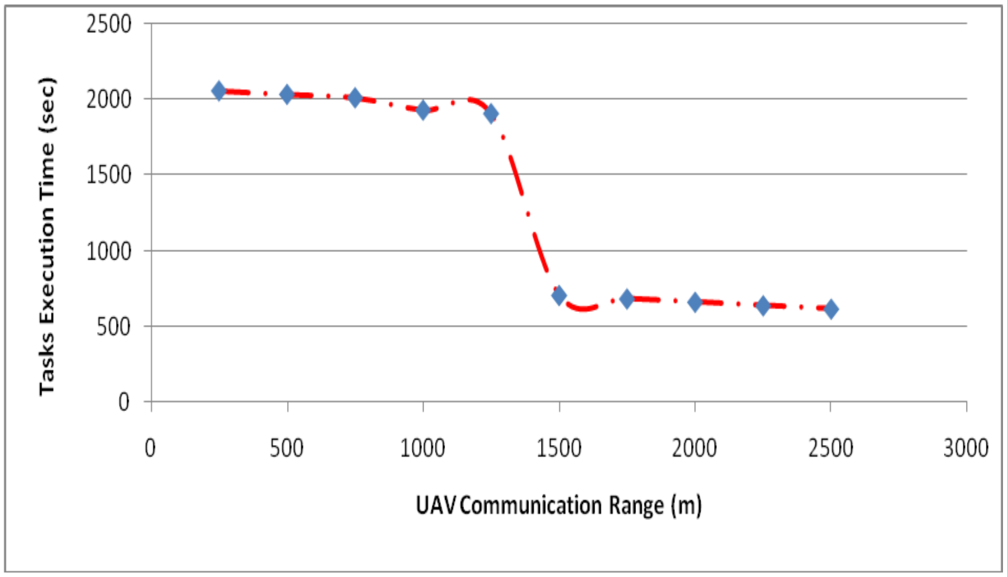

Figure 4.2. UAV1's trace of path time of task execution at different communication ranges

Figure 4.2 shows UAV1's path of tasks execution during different communication ranges that depicts the impact of increasing the UAV's communication ranges over the tasks' execution time, which reduced the UAV's ability to deliver its tasks earlier since it travelled less distance to its intended GS and target sensor. Figure 4.2 also shows a large drop in UAV1's path execution time when its communication range increased from 1250 meters to 1500 meters. At 1250 meters, UAV1 won tasks 10 and 6; while at 1500 meters UAV1 won tasks 11 and 4. Therefore, the reason for the drop in path execution time is that they have different coordinates.

Tasks must also be delivered on time such that any UAV will not bid for any auctioned task that will cause any of its current task(s) to timeout. Since tasks are generated randomly, their IDs and number of tasks won by an individual UAV may change from one run to the next; therefore, the time required to execute the task(s) may change accordingly. Table 4.7 shows the tasks path execution times and their tasks for UAV1 with varied communication ranges. The results show it took less time when the UAV's communication range increased for the same task(s) to be delivered to the same destination.

Table 4.7. UAV1's tasks execution time during different communication ranges

UAV Communication Range (m)	Tasks Path Execution Times (sec)	Task List
250	2056	11, 6, 10
500	2033	11, 6, 10
750	2011	11, 6, 10
1,000	1930	6, 10
1,250	1907	6, 10
1,500	708	11, 4
1,750	685	11, 4
2,000	663	11, 4
2,250	640	11, 4
2,500	618	11, 4

4.3.3 Results of Average Distance Travelled and Average Time to Perform Tasks

In a double auction, UAVs may submit their bids to more than one GS to increase their chances of winning an auctioned task as self-interested agents. Different cases were investigated to show the effect of increasing the buyers' communication ranges during a double auction. A base case involves nine random tasks auctioned to three GSs to sell to six UAVs bidding to execute these tasks. Table 4.8 shows the average total distance travelled to execute the tasks and the average time required to perform the tasks.

The data in Table 4.8 shows that the average time needed to execute tasks improved as the UAV's communication range increased. When the UAVs communication ranges increased from 250 meters to 2500 meters, the average distance of data delivery decreased by 16.8%. As shown in Table 4.9, this decrease resulted in all the GSs receiving a profit increase. GS1, GS2, and GS3 realized an increase in their average profit up to 39.8%, 57.9% and 20.9%, respectively.

Table 4.8. Data delivery statistics for nine random tasks by six UAVs

UAV Communication Range(m)	Mean Data Delivery Distance (m)	Mean Tasks Execution Time (sec)
250	9871	987
500	9625	962
750	9445	944
1,000	9264	927
1,250	9085	909
1,500	8905	890
1,750	8725	872
2,000	8545	855
2,250	8371	837
2,500	8213	821

Table 4.9. Average GSs profit from selling nine tasks to six UAVs

UAV Communication Range (m)	250	500	750	1000	1250	1500	1750	2000	2250	2500
GS1	$431	$454	$474	$493	$513	$533	$553	$573	$590	$602
GS2	$458	$504	$532	$559	$586	$613	$640	$668	$696	$723
GS3	$535	$548	$560	$573	$585	$597	$610	$622	$634	$647

This research used two cases to investigate the scalability of the market-based coordination. In Case-1, the number of tasks was increased from 9 to 12, and in Case-2, decreased from nine to six, respectively, while the number of UAVs that carried out the job remained unchanged at six. The task execution time decreased for each UAV delivering to the same destination as its communication range increased. As shown in Table 4.10, the average data delivery distance travelled by all six UAVs decreased.

Table 4.10. Data delivery statistics for 12 random tasks by six UAVs

UAV Communication Range(m)	Mean Data Delivery Distance (m)	Mean Tasks Time Execution (sec)
250	12676	1268
500	12489	1249
750	12301	1230
1,000	12104	1210
1,250	11821	1182
1,500	11119	1119
1,750	10924	1092
2,000	10572	1057
2,250	10341	1034
2,500	10131	1013

Even though the market is scaled up, it still benefitted from double auction, and is better off by 3% in the distance travelled when tasks are scaled up to twelve as compared to the base case; see Table 4.8. This improvement accompanied an improvement in the quality of the solution since the decrease in the average time of task execution showed the same result for the reduction in the distance travelled.

Computing the profit generated by each GS in the double auction provides a better view of these results. As the number of auctioned tasks increased for the same UAVs, the probability of increasing the profit for each participating UAV increased. By scaling the number of auctioned tasks to twelve, GS1, GS2 and GS3 received an increase in their profit by 52%, 46.6% and 30.2%, respectively.

In Case-2, the number of tasks was scaled down to six tasks to be bought by the same six UAVs who participated in double auction in Case-1. The data in Tables 4.11 and 4.12 show the results from increasing the UAVs communication range.

Table 4.11. Data delivery statistics for six random tasks by six UAVs

UAV Communication Range (m)	Mean Data Delivery Distance (m)	Mean Tasks Execution Time (sec)
250	8080	808
500	7937	794
750	7783	778
1,000	7640	764
1,250	7420	742
1,500	7258	726
1,750	6958	696
2,000	6793	679
2,250	6595	659
2,500	6430	643

To investigate the matter further, using the same tasks, the numbers of buyers were increased from six to nine to buy twelve tasks that were sold in Case-1. As in Case-

1, increasing the UAV's ranges affected the average data delivery time for an auctioned task by allowing it to be performed in a shorter time as data delivery will be completed earlier due to the increase in the UAVs' communication range. Table 4.13 and Figure 4.3 show the effect on average time of tasks execution when the UAVs' ranges increased.

Table 4.12. Average GSs profit from selling 6 tasks to 6 UAVs

UAV Communication Range (m)	250	500	750	1000	1250	1500	1750	2000	2250	2500
GS1	\$130	\$148	\$165	\$182	\$200	\$217	\$234	\$251	\$278	\$292
GS2	\$291	\$303	\$320	\$353	\$370	\$387	\$465	\$482	\$502	\$521
GS3	\$192	\$209	\$226	\$244	\$261	\$280	\$284	\$304	\$324	\$343

Table 4.13. Execution time of 12 tasks by UAVs during different communication ranges

UAV Communication Range(m)	Average Time Travelled by 6 UAVs	Average Time Travelled by 9 UAVs
250	1268	836
500	1249	824
750	1230	811
1,000	1210	798
1,250	1182	785
1,500	1119	740
1,750	1092	725
2,000	1057	700
2,250	1034	685
2,500	1013	670

Increasing the communication ranges also improved the quality of the solution by reducing the execution time, thus increasing each participant's profit. Table 4.14 shows that increasing a UAV's communication range led to a decrease in the time travelled and a decrease in the average distance travelled. Accordingly, the decrease in the distance travelled or the time of execution yielded an increase in the profit generated by participating UAVs, as shown in Table 4.15. This meant that the increase in profit would be divided evenly between the GSs and their UAVs.

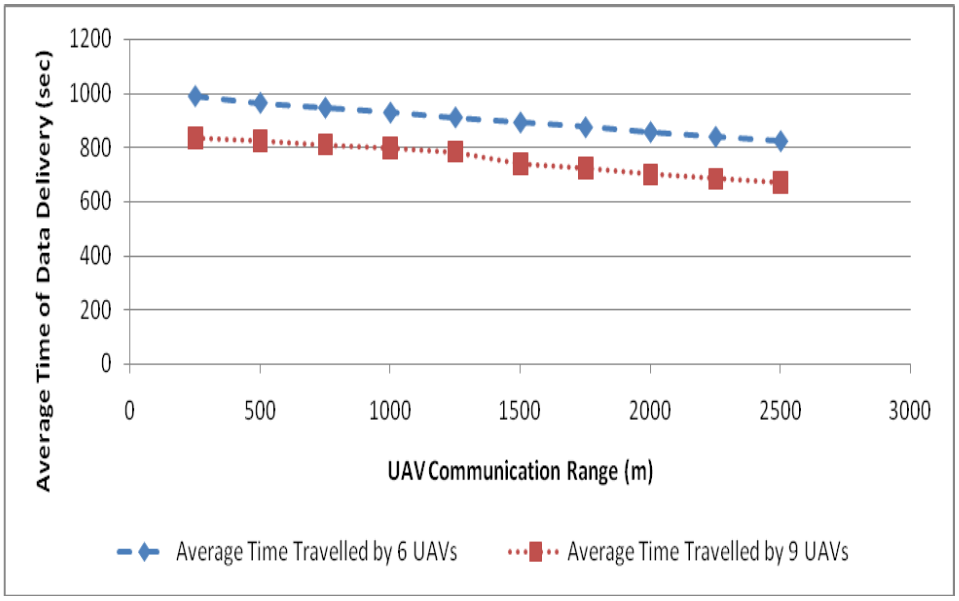

Figure 4.3. Execution time of same 12 tasks in double auction

Table 4.14. Data delivery statistics for 12 random tasks by nine UAVs

UAV Communication Range (m)	Mean Data Delivery Distance Cost	Mean Task Execution Time(sec)
250	$8364	836
500	$8239	824
750	$8114	811
1,000	$7981	798
1,250	$7854	785
1,500	$7398	740
1,750	$7253	725
2,000	$6997	700
2,250	$6851	685
2,500	$6699	670

Table 4.15. GSs average profit while auctioning five sets of 12 random tasks to nine UAVs

UAV Communication Range(m)	250	500	750	1000	1250	1500	1750	2000	2250	2500
GS1	$751	$774	$796	$822	$845	$968	$988	$1088	$1090	$1112
GS2	$832	$854	$877	$899	$921	$944	$966	$989	$1016	$1033
GS3	$722	$739	$756	$774	$792	$879	$901	$905	$948	$985

This reduction in tasks execution time meant that the distance travelled also reduced as observed earlier since the UAV's speed is constant. In fact, the decrease in the average execution time is a direct result of the decrease in the average distance travelled as depicted in Figure 4.4, where six UAVs compete to buy a different number of auctioned tasks.

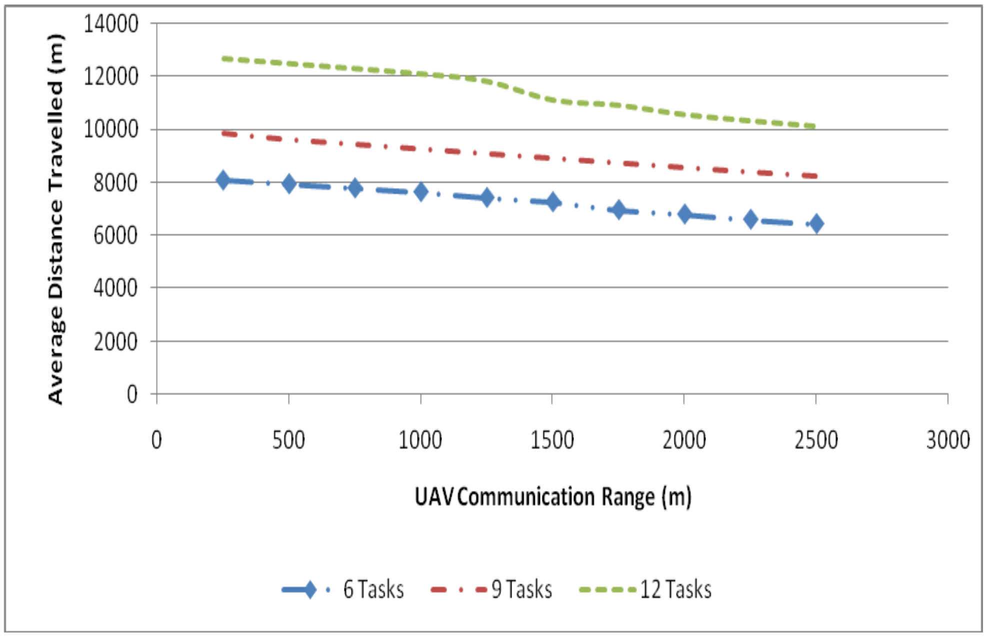

Figure 4.4. Average distance travelled by six UAVs competing over a different number of tasks

Since the average task execution time decreased as a result of increasing the UAVs' communication ranges, the profit generated by the UAVs increased as well. The increase in the UAVs' profit led to an increase in the average profit generated by GSs as their profit is the summation of all profit made by each participant UAV that bought task(s) from that specific GS, as mentioned before in Equation 4.2. As an example, refer to Figure 4.5 and Table 4.15.

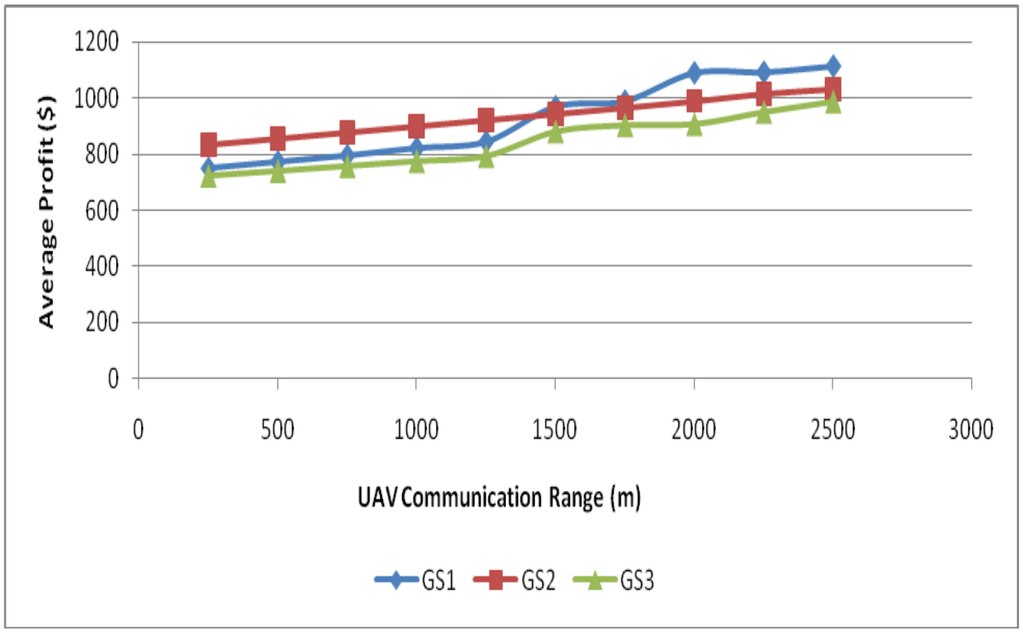

Figure 4.5. GSs average profit of selling 12 tasks to nine UAVs

Scaling up the number of UAVs from six to nine competing over the same number of tasks resulted in the same improvement as the reduction in the average of task execution time and the distance travelled (Figures 4.3 and 4.4, respectively) as well as the profit generated due to increasing the UAVs' communication ranges (Figure 4.5).

Scaling up the number of tasks to twelve resulted in a decrease of 31.8% in the average task execution times, while the UAVs ranges increased from 250 meters to 2500 meters; see Table 4.13.

Increasing the number of UAVs competing for the same number of tasks resulted in finding the fittest UAV, and a quicker response, which decreased the average distance travelled as well as the average task execution times, as depicted by Figure 4.6.

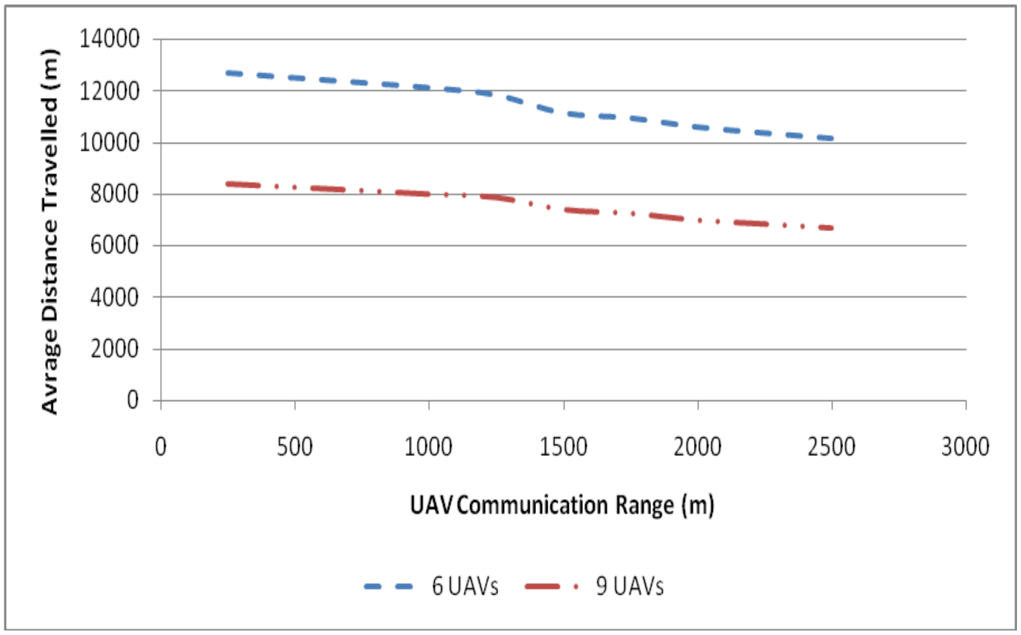

Figure 4.6. UAVs average distance travelled to execute 12 tasks

4.3.4 System Robustness

In a dynamic research environment, system robustness is one of the most important criteria for reliability. The availability of task data to the TI is very crucial, particularly in military operations or rescue missions; therefore, the system must be robust. To validate the system's robustness, UAV1, which had four tasks in its current plan, {9, 10, 11, and 12} assumed to fail before delivering any of its tasks to GS2, as shown in Figure 4.7.

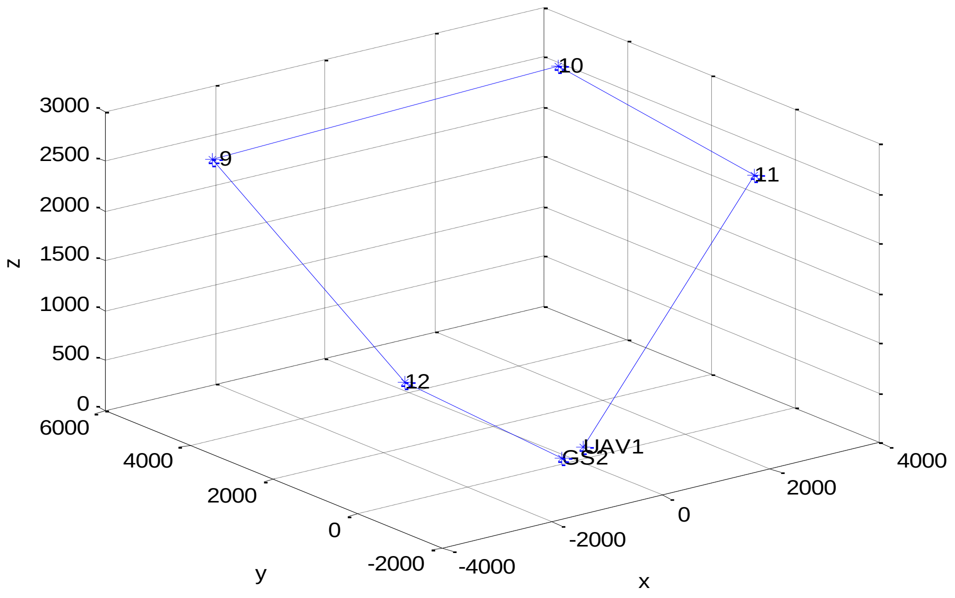

Figure 4.7. Path for executing tasks by UAV1

As a result, GS2 detected the absence of UAV1's task data after γ time passed because the deadline due time, t_H, for the first task was due by UAV1. To recover from this failure, using the time-increasing function with higher prices to accelerate recovering these tasks, GS2 re-auctioned the undelivered tasks to the UAVs within its communication range. The eligible UAVs submitted their bids for those tasks from where they were at the beginning of re-auctioning, as shown in Table 4.16. During re-auctioning, UAV2 won tasks 9 and 12 in addition to those delivered earlier, and on time. For clarity, the execution path of tasks won by UAV2 before and after re-auctioning is shown in Figure 4.8. While UAV4 won task10 and delivered it to GS1 and the fourth re-auctioned task was delivered by UAV1, it recovered from its failure and was able to win this task during re-auctioning because at that time the task was already on its task list.

Table 4.16. UAVs coordinates at re-auctioning tasks

UAV_i	x	y	z
1	313	-61	315
2	-115	200	0
3	-115	200	0
4	219	13	139
5	-115	-180	134
6	-115	-200	0

This robustness affected the time of data delivery due to the extended time due to tasks not delivered on time. Therefore, the average time for tasks delivered increased from 1005 seconds to 1095 seconds. The difference in these two numbers represents the cost for making the system robust to uncertainty, and is the cost for assuring data delivery.

Since re-auctioning in this case takes place only when tasks are not delivered at their expected due time, re-auctioning undelivered tasks will cause some delay on the average data delivery time. As shown in Figure 4.9, when 12 tasks was delivered by six UAVs, at a range of 250 meters, re-auctioning caused a 295 seconds delay in data delivery; while at a range of 2500 meters, re-auctioning caused the same data to be delivered 127 seconds late.

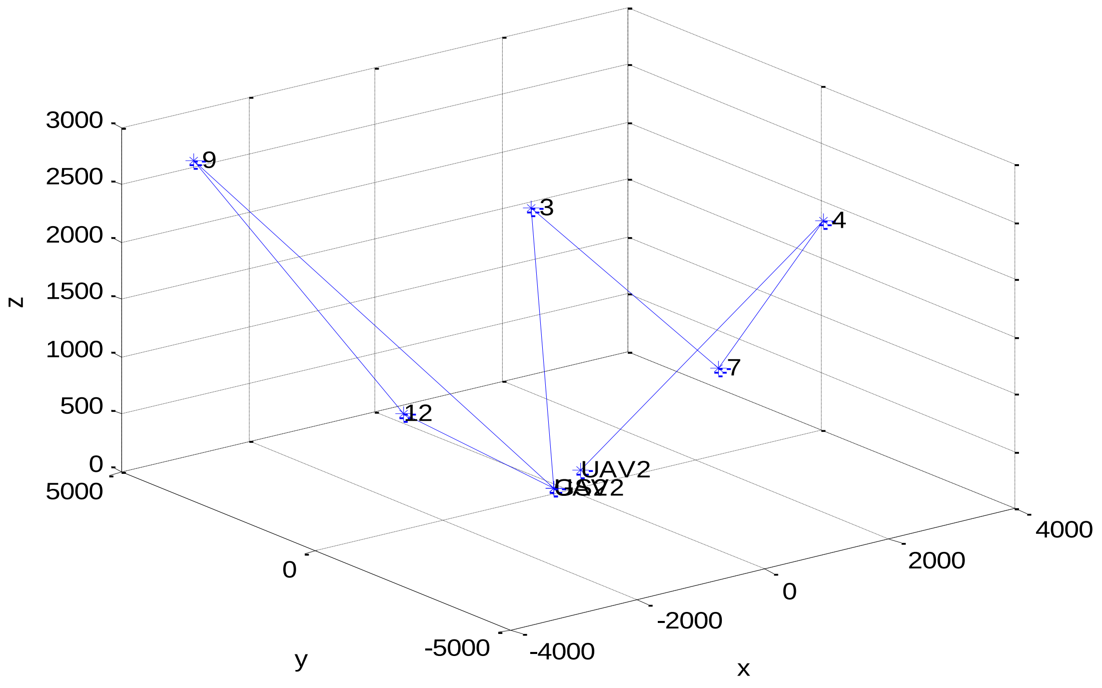

Figure 4.8. Path for executing tasks by UAV2 before and after validating robustness

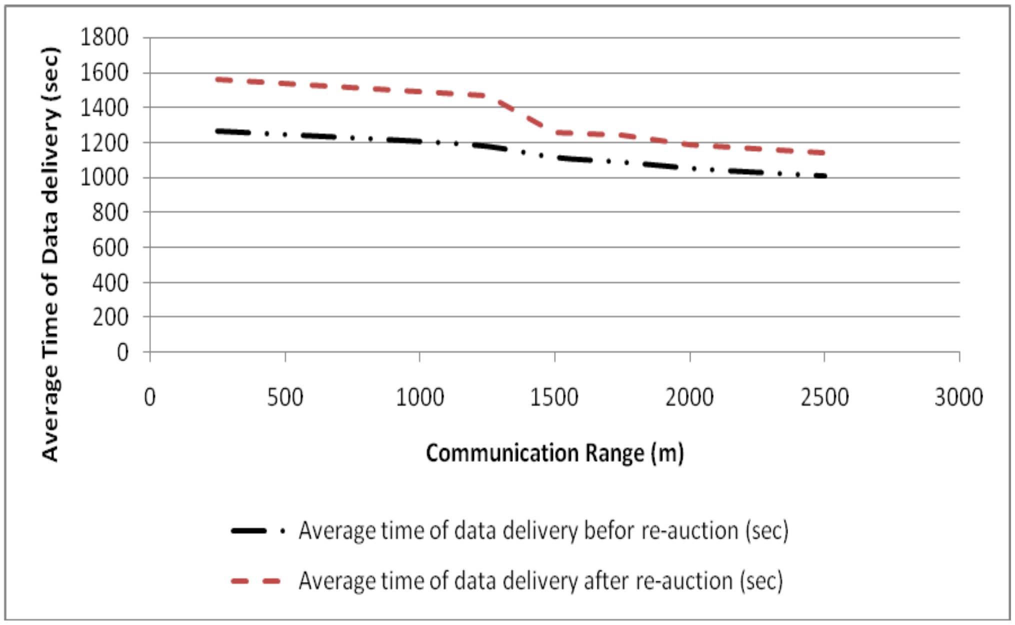

Figure 4.9. Data delivery time history for 12 tasks by six UAVs before and after re-auctioning

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

This dissertation studied the problem of collecting data from a series of unmanned ground sensors (UGSs) using a team of unmanned aerial vehicles (UAVs), and delivering the information to a task initiator (TI) through the ground stations (GSs). The problem formulated into a procurement auction between the TI and GSs and a double auction between the GSs and UAVs, which was formalized as a continuous time-constraint version of a multi-traveling salesperson problem.

A market-based coordination method was simulated using double auction methods applying concepts of price, revenue, and cost to trade task allocations between the UAVs and GSs. A double auction enabled the TI to benefit from 1) having different ground stations reach farther locations in a timely manner, and 2) cost efficiency by allowing task executions through a different agent in the system, which led to a decrease in time and cost.

The market-based method used communication efficiently because it compressed the UAVs' data into bids. The role of the TI, in addition to task creation and holding auctions, was as a market matcher that matched the sell bids and purchased bids during the double auctions. The GSs are buyers that participated with the TI and sellers in the double auctions. This allowed the UAVs to control the allocation process by the bids they submitted. All GSs participated in auctioning tasks received from the TI to UAVs that submitted bids to one or more GSs. This gave the UAVs more chances to win the auctioned tasks. Therefore, the buyer's cost for executing the task was considered in a bid that reflected the maximum price the buyer can pay for that task. Thus, the seller's bid is considered the minimum price acceptable to trade the task.

As a global objective, this dissertation used double auctions to minimize the overall cost. During the auctioning process, the UAV's cost for executing tasks was considered a bid that did not exceed the auctioneer's reserve price. Therefore, more than one agent may submit the same bid, and the auctioneer's role was as tiebreaker. GSs were bounded by the TI's reservation price when submitting their bids to the TI; hence, they benefitted from the buyers that bid for less in case they did not win the task. Yet there was still an opportunity for them to receive partial profit in case their buyer agent became the winner. The GSs used the TI's reserve price to come up with their own price for the auctioned task that was based on their linear distance to the auctioned task, and the time needed to execute a task by a prospective winner, which was based on the known speed of the UAVs. UAVs submitted their buy offers to the market (GS in this situation) according to the market rules; the seller's decision was based on the minimum bid.

For any auctioned task, there is a deadline time for the task data to be received by the auctioneer. When execution of a candidate task would cause the execution of any task in the UAV's current plan to miss its deadline, the UAV did not submit a bid for that auctioned task. Thus, data delivery time affected the amount of profit a UAV could earn; the sooner the task(s) were delivered to the auctioneer, the more profit the UAV generated, which was a direct result of the UAV's communication range. Increasing the UAVs' communication ranges enabled them to deliver the needed data sooner.

This double auction operated differently than a regular double auction. The overall objective was to minimize overall costs. Bids were ranked from minimum to the maximum; the agent with the minimum bid was awarded the task assuming all constraints were met.

Using the increasing price function, re-auctioning the GSs' tasks improved the quality of the solution because all tasks at the end were bid on and thereafter assigned. Re-auctioning made the system robust and maintained strong control over all auctioned tasks regardless of any uncertainty agents might have faced (e.g., UAV communication malfunction) during task execution. When a UAV was unable to perform its tasks, these tasks were assigned to other agents through a new auction held by the GS that was supposed to receive the data on time.

This double auction generated a random number of tasks to be executed by a different number of UAVs to determine the effect on the quality of the solution. For the same number of tasks, having more UAVs participate in the auction resulted in a quicker response by reducing the average distance traveled, and the average time of tasks execution decreased as well. Increasing the communication ranges also improved the quality of the solution by reducing the execution time and increasing the profit for each participant.

Assuming each UAV is self-interested, the total time to complete all the tasks assigned to it was minimized. Tasks were assigned through bidding according to double auction rules. Additionally, communication malfunctions of the UAVs were simulated.

In this research, UAVs could become disabled and so fail to deliver their task(s). Therefore, UAV malfunctioning introduces new challenges for this system. Since each UAV is responsible for delivering its won tasks, it will have to assure delivering the completed tasks to the source GS without any delay to maximize its won profit. Otherwise, the UAV suffers a severe penalty. The robustness issue was addressed by introducing a point of failure for some UAVs during task execution. This was done by disconnecting a UAV before it delivered all or some of its tasks to their final destination. Since the tasks are time limited, the system's robustness was validated by re-auctioning undelivered tasks(s) by the prospective receiver GS after gamma (γ) time passed since their due time. However, a disabled UAV could recover from its failure during re-auctioning, and if it participated in the bidding of any task(s) on its task list, the UAV wins those tasks if the UAV was already on its way to deliver the tasks' data when it was disabled.

The GS malfunctioning issue is an area for future research, since the current constraints made it difficult to handle at this time. Going forward, UAVs may have different capabilities such as different speeds and different communication ranges; therefore, research is needed to explore the effect of these on the quality of the solution and the profit that can be generated. To address relevant issues in double auctions, future research can also expand on this research by exploring double auctions between GSs in addition to the combinations (GSs and UAVs) investigated in this research.

REFERENCES

Ajorlou, A., Homaifar, A., Esterline, A., Moore, J. G., and Bamberger, R. J. Market-based coordination of UAVs for time-constrained remote data collection and relay. In *Proceedings of the International Conference on Intelligent Systems* (2007).

Ajorlou, A., Homaifar, A., Esterline, A., Moore, J. G., and Bamberger, R. J. Robust multi-UAV data collection. In *Proceedings of the AIAA Infotech@Aerospace Conference and Exhibit* (2007).

Botelho, da Costa, S. S., and Alami, R. M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the International Conference on Robotics and Automation* (1999).

Deneubourg, J.-L., Theraulaz, G., and Beckers, R. Swarm-made architectures. In *Proceedings of the European Conference on Artificial Life* (1991).

Dias, M. B., and Stentz, A. T. Traderbots: A market-based approach for resource, role, and task allocation in multi-robot coordination. Tech. Rep. CMURI -TR-03-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2003.

Dias, M. B., Zlot, R. M., Kalra, N., and Stentz, A. T. Market-based multi-robot coordination: A survey and analysis. Tech. Rep. CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2005.

Dias, M., Zlot, R., Karla, N., and Stentz, A. (2006). Market-based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE*, 94(7):1257–1270.

Dias, M., Zlot, R., Zinck, M., and Stentz, A. Robust multi-robot coordination in dynamic environments. In *Proceedings of the International Conference on Robotics and Automation* (2004).

Fasli, Maria, "Agent Technology for e-Commerce." John Wiley & Sons Ltd, The Atrium, Southern Gate, Chester, West Sussex PO19 8SQ, England (2007).

Gerkey Brian P. and Maja J. Mataric´, "Sold!: Auction methods for multi-robot coordination." *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, 18**,** 5(2002), 758–768.

Kalra, N., Ferguson, D., and Stentz, A. T. Constrained exploration for studies in multirobot coordination. In *Proceedings of the IEEE International Conference on Robotics and Automation* (2006).

Kalra, N., Stentz, A., and Ferguson, D. (2005). Hoplites: A Market Framework for Complex Tight Coordination in Multi-Agent Teams. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1170–1177, New Orleans, USA.

Koenig, S., Tovey, C., Zheng, X., and Sungur, I. (2007). Sequential Bundle-Bid Single-Sale Auction Algorithms for Decentralized Control. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1359–1365.

Kube, C. R., and Zhang, H. Collective robotic intelligence. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior* (1992).

Lemaire, Thomas, Rachid Alami, Simon Lacroix. A Distributed Tasks Allocation Scheme in Multi-UAV Context. *In Proceedings of the 2004 IEEE International Conference on ROWW h Automation (2004)*

Moore, K. L., White, M. J., Bamberger, R. J., and Watson, D. P. Cooperative UAVs for remote data collection and relay. In *Proceedings of AUVSI Unmanned Systems Conference and Expo* (2005).

Parker, L. Alliance: An architecture for fault-tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation 14,* 2 (1998), 220 - 240.

Retrieved from http://www.wordiq.com/definition/Procurement_auction on April 15, 2010.

Sandholm Thomas, "An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations." *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, 1993, pp. 295—308.

Sandholm, T. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence* (1993).

Sandholm, T., and Lesser, V., "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework." *Proceedings of the first International Conference on Multiagent Systems (ICMAS-95),* 1995.

Sariel, S., Balch, T., and Erdogan, N. (2006). Robust Multi-Robot Cooperation Through Dynamic Task Allocation and Precaution Routines. In *The 3rd International Conference on Informatics in Control, Automation and Robotics (ICINCO),* Minneapolis, USA.

Smith, G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

Smith, R., “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver.” *IEEE Transactions on Computers*, Vol. C-29, No. 12, 1980.

Steels, L. Cooperation between distributed agents through self-organization In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems '90* (1990).

Stentz, A. and Dias, M. B., “A Free Market Architecture for Coordinating Multiple Robots.” *Technical Report, CMU-RI-TR-99-42, Robotics Institute, Carnegie Mellon University*, 1999.

Tovey, C., Lagoudakis, M. G., Jain, S., and Koenig, S. (2005). *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III, Proceedings from the 2005 International Workshop on Multi-Robot Systems*, chapter The Generation of Bidding Rules for Auction-based Robot Coordination, pages 3–14. Springer Netherlands.

Zheng, X., Koenig, S., and Tovey, C. (2006). Improving Sequential Single-Item Auctions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 2238–2244, Beijing, China.

APPENDIX A

DOUBLE AUCTION TRACES

```
Actual GS communication radius for the overlap case is: 400
TI communication radius: 231


GSs and UAVs bid pairs:
       GS     GS_Bid     UAV     UAV_Bid
   1  475    3  406
   2  Inf    3  433
   3  Inf    3  447

GSs and UAVs bid pairs:
       GS     GS_Bid     UAV     UAV_Bid
       1      1105        1      1034
       2      1101        1      1032
       3       Inf        1      1069

GSs and UAVs bid pairs:
       GS     GS_Bid     UAV     UAV_Bid
       1      1077        1       907
       2       Inf        1       945
       3      1085        1       911

GSs and UAVs bid pairs:
       GS     GS_Bid     UAV     UAV_Bid
   1  Inf    1  307
   2  524    1  307
   3  Inf    1  307

GSs and UAVs bid pairs:
       GS     GS_Bid     UAV     UAV_Bid
   1  Inf    2  766
   2  805    2  735
   3  Inf    2  764

GSs and UAVs bid pairs:
       GS     GS_Bid     UAV     UAV_Bid
   1  407    2   64
```

```
    2   380     2    64
    3   Inf     2    64

GSs and UAVs bid pairs:
         GS       GS_Bid       UAV      UAV_Bid
    1   Inf     5   939
    2   980     5   920
    3   Inf     5   942

GSs and UAVs bid pairs:
         GS       GS_Bid       UAV      UAV_Bid
    1   561     5   285
    2   586     5   297
    3   Inf     5   297

GSs and UAVs bid pairs:
         GS       GS_Bid       UAV      UAV_Bid
         1         Inf          9       1343
         2         Inf          9       1331
         3        1374          9       1304

GSs and UAVs bid pairs:
         GS       GS_Bid       UAV      UAV_Bid
         1        1158          9        947
         2        1179          9        957
         3         Inf          9        970

GSs and UAVs bid pairs:
         GS       GS_Bid       UAV      UAV_Bid
    1   599     9   133
    2   Inf     9   133
    3   602     9   133

GSs and UAVs bid pairs:
         GS       GS_Bid       UAV      UAV_Bid
         1        1077          4        993
         2         Inf          4       1014
         3        1052          4        981


UAV_TaskList =

  Columns 1 through 5
    [1x3 double]    [1x2 double]    [1]    [12]    [1x2 double]
  Columns 6 through 9
     []     []     []    [1x3 double]

Ground Stations coordinates (in rows):
   231     0     0
  -115   200     0
  -115  -200     0

UAVs Initial Coordinates (in rows):
  -114   219   121
   -78   199   106
   218    28   132
   -65  -151   144
   -24  -127   169
   -52  -144   111
  -115  -180   106
  -115  -180   119
  -115  -180   132

Tasks Coordinates (in rows):
        2073         832         759
        2297        4431        1144
        2802       -3897        1470
       -2120        1407         433
       -1624        2492        2422
         261        1538        1028
       -1035        2227        3860
        1926        1837         503
       -4385       -3825        2759
        2814        2401        3910
        1079       -1712        1941
         567       -3680        3210

RECORDS FOR UAV1:
Tasklist:
     2     3     4

Minimum Route:
     1     4     2     3     1

StartTime  DeliveryTime  PathTime LatestDeliveryTime
          22        2063        2041        2581
RECORDS FOR UAV2:
```

```
Tasklist:
    5     6

Minimum Route:
    2     6     5     2
StartTime  DeliveryTime  PathTime LatestDeliveryTime
        22         749         727        1868
RECORDS FOR UAV3:
Tasklist:
    1

Minimum Route:
    3     1     1

StartTime  DeliveryTime  PathTime LatestDeliveryTime
        22         391         369        2359

RECORDS FOR UAV4:
Tasklist:
   12

Minimum Route:
    4    12     3

StartTime  DeliveryTime  PathTime LatestDeliveryTime
        22         914         892        4916

RECORDS FOR UAV5:
Tasklist:
    7     8

Minimum Route:
    5     7     8     1

StartTime  DeliveryTime  PathTime LatestDeliveryTime
        22        1117        1095        2709

RECORDS FOR UAV9:
Tasklist:
    9    10    11

Minimum Route:
    9     9    11    10     1
```

```
StartTime  DeliveryTime  PathTime LatestDeliveryTime
        22        2189        2167        2804

UAV LIST FOR GS1:
    3     1     5     9     9

UAV LIST FOR GS2:
    1     1     2     2     5

UAV LIST FOR GS3:
    9     4

   AvgDist(m)   AvgTime(s)  Dist_std(m)  Time_std(s)
      8101         810        8378         838




RESULTS FROM THE ROBUSTNESS
Records for the disconnected UAV:
UAV Disconnected = 1
Time it got eliminated = 50
Time it recovered = 4400
Its latest task delivery time = 2581
Tasks in its tasklist are:
    4     2     3

Tasks delivered are:
None
Tasks undelivered are:
    4     2     3



TIME AT WHICH THE REAUCTION OF TASKS FROM THE DISCONNECTED
UAV BEGINS: 2586
THE REMAINING TASKS TO BE DONE BY ALL THE OTHER UAV JUST
BEFORE THE REAUCTION OF TASKS FROM THE DISCONNECTED, UAV1:
The remaining tasks for UAV 2  are: None
The remaining tasks for UAV 3  are: None
The remaining tasks for UAV 4  are: None
The remaining tasks for UAV 5  are: None
The remaining tasks for UAV 6  are: None
The remaining tasks for UAV 7  are: None
The remaining tasks for UAV 8  are: None
The remaining tasks for UAV 9  are: None
UAVs current coordinates just before the reauction of tasks from the eliminated UAV:
  -353   361   158
  -115   200     0
   232     1     1
  -115  -200     0
   232     1     0
   -52  -144   111
  -115  -180   106
  -115  -180   119
   232     1     2

GSs and UAVs bid pairs:
        GS      GS_Bid      UAV     UAV_Bid
    1   Inf    2   503
    2   524    1   425
    3   Inf    1   450

GSs and UAVs bid pairs:
        GS      GS_Bid      UAV     UAV_Bid
        1       1105          2       1038
        2       1101          2       1036
        3        Inf          2       1074

GSs and UAVs bid pairs:
        GS      GS_Bid      UAV     UAV_Bid
        1       1077          2        907
        2        Inf          2        945
        3       1085          2        911

RECORDS FOR UAVs WHICH WON REAUCTIONED TASK(S):
(Note: Those UAVs which did not win any reacuctioned tasks have the same records as
shown before)
UAV1:
TASKLIST:
Task completed before REAUCTION:
Task completed after REAUCTION:
    4

Minimum Route from start to end:
    1    4    2
```

```
StartTime  DeliveryTime  PathTime LatestDeliveryTime
        22       2977        415       5117

UAV2:
TASKLIST:
Task completed before REAUCTION:
    6     5
Task completed after REAUCTION:
    2     3

Minimum Route from start to end:
    2     6     5     2     3     1

StartTime  DeliveryTime  PathTime LatestDeliveryTime
        22       4356       2493       1868

UAV LIST FOR GS1:
    3     5     9     9     2

UAV LIST FOR GS2:
    2     2     5     1     2

UAV LIST FOR GS3:
    9     4

   AvgDist(m)   AvgTime(s)  Dist_std(m)  Time_std(s)
      8257        826       9407        941



Ground Stations coordinates (in rows):
   231      0      0
  -115    200      0
  -115   -200      0

UAVs and Tasks coordinates:

9 UAVs

UAVs Initial Coordinates (in rows):
    69      114   138
   -97      210   100
   -89      205   146
    -8      -118  136
   -58      -147  155
    68      -74   160
  -115      -180  143
  -115      -180  166
  -115      -180  112


20 Tasks
Tasks coordinates (in rows):
        3148        -613        4615
        4706        -544        2990
       -3731        2547        3855
        4134        1551         405
        4158        2655        1679
        2922         853        2995
        4058       -3374        1985
        4595        -102         702
         469        2094        4694
        1324        4598        2906
       -4025       -3132         954
       -2215       -1185        3973
        4576        2952        1917
        3003       -2762        2119
        -146       -3810        2533
        -782        1463        2818
        4572         -16         600
       -3581       -2240        3871
       -3424        1797        3072
        4649       -1596        4214

Ground Stations coordinates (in rows):
   231      0     0
  -115    200     0
  -115   -200     0



6 UAVs

UAVs Initial Coordinates (in rows):
    17   144   115
    37   132   139
   131   -38   113
   146   -29   153
  -115  -180   177
  -115  -180   127


12 Tasks

Tasks coordinates (in rows):
        2073         832         759
        2297        4431        1144
```

```
 2802       -3897        1470
-2120        1407         433
-1624        2492        2422
  261        1538        1028
-1035        2227        3860
 1926        1837         503
-4385       -3825        2759
 2814        2401        3910
 1079       -1712        1941
  567       -3680        3210
```

APPENDIX B

CODE

File CalUAVDistance

```
function DistTravelled = CalUAVDistance(UAV_Speed,
TimeTaskStart,NegTIME,TaskDeliveryTime)
            % Computes the instaneous distance travelled by UAV
            % NegTIME is the instance at which negotiation is to happen

            if NegTIME<=TimeTaskStart
                DistTravelled = 0;
            else
                DistTravelled =
UAV_Speed*(min(NegTIME,TaskDeliveryTime)-TimeTaskStart);
            end
            %%%%%%%%%%%%%%%%%%%%
```

File CalUAVsNegBids

```
function [UAVNegPath,PathCostNeg] =
CalUAVsNegBids(DistTravel_UAVi,UAVi_MovingTo,...

UAVj_MovingTo,GS_UAVi_RouteCords,GS_UAVj_RouteCords,UAVi_NewLocation,..
.

UAVj_NewLocation,DistUnitCost,TimeUnitCost,UAV_Speed,TimeHistory,ROUTEi
d_j,Deadlines,GS_UAViRoutes)

%%% UAVi negotiating a task
global TaskName
UAVNegPath = [];
PathCostNeg = inf;

if DistTravel_UAVi>0
    TaskToNeg_i = GS_UAVi_RouteCords(UAVi_MovingTo,:);
    TaskName = GS_UAViRoutes(UAVi_MovingTo);
    %%% compute the cost of the UAVs to the tasks
   UAVi_NegBid = norm(TaskToNeg_i-UAVi_NewLocation)*DistUnitCost +
norm(TaskToNeg_i-UAVi_NewLocation)*TimeUnitCost/UAV_Speed; % in USD
   UAVj_NegBid = norm(TaskToNeg_i-UAVj_NewLocation)*DistUnitCost +
norm(TaskToNeg_i-UAVj_NewLocation)*TimeUnitCost/UAV_Speed; % in USD
   if UAVj_NegBid<UAVi_NegBid
       TempUAVjRoute = [GS_UAVj_RouteCords(1:UAVj_MovingTo-
1,:);UAVj_NewLocation;TaskToNeg_i;GS_UAVj_RouteCords(1:UAVj_MovingTo,:)
];
       %%% computing the cost of the new path
       NegPathDist_j = ComputeNegPathDistance(TempUAVjRoute);  %% a
function called to compute path distance

       PathTimeNeg_j =  NegPathDist_j/UAV_Speed;
       PathCostNeg =  NegPathDist_j*DistUnitCost +
PathTimeNeg_j*TimeUnitCost; % in USD
       TaskCompletionTimeNeg_j = TimeHistory(ROUTEid_j,1)+
PathTimeNeg_j;
       if
TaskCompletionTimeNeg_j<=TimeHistory(ROUTEid_j,3)&&TaskCompletionTimeNe
g_j<=Deadlines(TaskName)
           UAVNegPath = TempUAVjRoute;
          % GS_UAVi_RouteCords(UAVi_MovingTo,:) = [];
       end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   %%% UAVj negotiating a task
% if DistTravel_UAVj>0
%     TaskToNeg_j = GS_UAVj_RouteCords(UAVj_MovingTo,:);
%     %%% compute the cost of the UAVs to the tasks
%    UAVj_NegBid = norm(TaskToNeg_j-UAVj_NewLocation)*DistUnitCost +
norm(TaskToNeg_j-UAVj_NewLocation)*TimeUnitCost/UAV_Speed; % in USD
%    UAVi_NegBid = norm(TaskToNeg_j-UAVi_NewLocation)*DistUnitCost +
norm(TaskToNeg_j-UAVi_NewLocation)*TimeUnitCost/UAV_Speed; % in USD
%    if UAVi_NegBid<UAVj_NegBid
%        TempUAViRoute = [GS_UAVi_RouteCords(1:UAVi_MovingTo-
1,:);UAVi_NewLocation;TaskToNeg_j;GS_UAVi_RouteCords(1:UAVi_MovingTo,:)
];
%        %%% computing the cost of the new path
%        NegPathDist_i = ComputeNegPathDistance(TempUAViRoute);  %% a
function called to compute path distance
%
%        PathTimeNeg_i =  NegPathDist_i/UAV_Speed;
%        PathCostNeg_i =  NegPathDist_i*DistUnitCost +
PathTimeNeg_i*TimeUnitCost; % in USD
%        TaskCompletionTimeNeg_i = TimeHistory(ROUTEid_i,1)+
PathTimeNeg_i;
%        if TaskCompletionTimeNeg_i<=TimeHistory(ROUTEid_j,3)
%            GS_UAVi_RouteCords = TempUAViRoute;
%            GS_UAVj_RouteCords(UAVi_MovingTo,:) = [];
%        end
%    end
% end
```

File

```
function NegPathDist = ComputeNegPathDistance(TempUAVRoute)
% computes the distance of the new path formed by adding a new task
from
% another UAV
```

```
NegPathDist = 0;
for i =1:size(TempUAVRoute,1)-1
    NegPathDist = NegPathDist + norm(TempUAVRoute(i+1,:)-
TempUAVRoute(i,:));
end
```

File

```
function [RouteLenghts,CumRouteLenghts] =
ComputeSegmentDistance(GS_UAVRouteCordAll,GS_UAVRoutesAll)
global GS_UAVCommStrength GS_CommRange UAV_CommRange;
RouteLenghts = [];
CumRouteLenghts = [];
NumberOfRoutes = length(GS_UAVRoutesAll);
for u=1:NumberOfRoutes
    RouteLenghts{u}(:,1) = 0;
    NumOfPoints = size(GS_UAVRouteCordAll{u},1);
    %if NumOfPoints>2 % Avoid dealing with a route with no task
    for r = 1:NumOfPoints-1
    RouteLenghts{u}(:,r+1) = norm(GS_UAVRouteCordAll{u}(r+1,:)-
GS_UAVRouteCordAll{u}(r,:));
    end
    RouteLenghts{u}(:,NumOfPoints)= RouteLenghts{u}(:,r+1)-(1-
GS_UAVCommStrength)*(GS_CommRange+UAV_CommRange);
    CumRouteLenghts{u} = cumsum(RouteLenghts{u}); % zero included for
reference purposes
    %end

end
```

File

```
function GS_Cord = CreateAllGroundStations(NumberOfGS, TI,
TI_CommRange, GS_CommRange,Scenario)

GS_AngleChange = 2*pi/NumberOfGS;  % Angular displacement between anay
two GS from TI
GS_Angle = [1:NumberOfGS].*GS_AngleChange; % The angles at which the
Manager distributes the GS

if Scenario ==1
   GS_Angle = [0,pi/2,pi];
    % FAR APART
    RADIUS = 0.9 *(TI_CommRange+GS_CommRange);
    GS_X = RADIUS.*cos(GS_Angle)+ TI(:,1);
    GS_Y = RADIUS.*sin(GS_Angle) + TI(:,2);
elseif Scenario ==2
    % INTERSECT
   GS_X = [0, GS_CommRange,  -GS_CommRange ]+ TI(:,1);
   GS_Y = [2*GS_CommRange, GS_CommRange,  GS_CommRange ] + TI(:,2);
```

```
elseif Scenario ==3
    % TANGENT
RADIUS = 2*GS_CommRange/sqrt(3);
    GS_X = RADIUS.*cos(GS_Angle)+ TI(:,1);
    GS_Y = RADIUS.*sin(GS_Angle) + TI(:,2);
end

GS_Z = zeros(1,NumberOfGS);
GS_Cord = [GS_X',GS_Y',GS_Z'];
```

File

```
function UAVsAll = CreateAllUAVs(NumberOfUAVs,TI_CommRange,
UAV_CommRange,UAVCommStrength,UAVAngleFromTI,TI)


% This function creates Unmanned Aerial Vehicles and displays them in
such
% a way that at each Ground Station(i.e. GS) has at least one UAV in
its
% communication range
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % The initial angles at which the UAVs are distributed from the TI
    ChangeInTeta_TI = 2*pi/NumberOfUAVs;
    Teta_TI_UAVs = [1:NumberOfUAVs].* ChangeInTeta_TI; % X-axis angle

    Dist_TI_UAVs = UAVCommStrength*(TI_CommRange + UAV_CommRange);
    UAV_X = cos(UAVAngleFromTI)*Dist_TI_UAVs.*cos(Teta_TI_UAVs) +
TI(1,1);
    UAV_Y = cos(UAVAngleFromTI)*Dist_TI_UAVs.*sin(Teta_TI_UAVs) +
TI(1,2);
    Z_Value = Dist_TI_UAVs.*sin(UAVAngleFromTI) + TI(1,3);
    UAV_Z(:,1:NumberOfUAVs) = Z_Value;
    UAVsAll = [UAV_X',UAV_Y',UAV_Z'];
```

File

```
function [GS_Cord, GS_CommRange,TI_CommRange] =
CreateGroundStations(NumberOfGS, TI,TangGS_CommRange,Scenario,
GS_GS_CommStrength)

GS_Angle = [0,2*pi/3, 4*pi/3];
RADIUS = 2*TangGS_CommRange/sqrt(3); % radius at which all three GSs
are tangential
GS_X = RADIUS.*cos(GS_Angle)+ TI(:,1);
GS_Y = RADIUS.*sin(GS_Angle) + TI(:,2);
GS_Z = zeros(1,NumberOfGS)+  TI(:,3);
GS_Cord = [GS_X',GS_Y',GS_Z'];
```

```
if Scenario ==1      % FAR APART
    GS_CommRange = GS_GS_CommStrength*TangGS_CommRange;
elseif Scenario ==2  % INTERSECT
    GS_CommRange = (1 + GS_GS_CommStrength)*TangGS_CommRange;
elseif Scenario ==3 % TANGENT
    GS_CommRange = TangGS_CommRange;
end

TI_CommRange = RADIUS; %(RADIUS/GS_TI_CommStrength) - GS_CommRange;
```

File

```
function [GS_UAVs_Route,UAVs_Route,UAVRouteCord,ID_UAVs,
UAVTaskStartTime, UAVTaskEndTime] =
DisplayUAVTaskList(WinnerUAV_Route,UAVs, GS_Cord,
Task,WinnerPathTime,TaskCreationTimes)
global UAVsNames Count GS_UAVRoutesAll Counter GS_UAVRouteCordAll
Deadlines TimeHistory
Route = [];
Count = 0;
UAVRouteCord = [];
TimeForTaskExecution = [];
ID_UAVs  = [];
for m=1:length(WinnerUAV_Route)
    if ~isempty(WinnerUAV_Route{m})
        Count = Count + 1;
        Counter = Counter + 1;
   UAVs_Route{Count} = WinnerUAV_Route{m};
   Route = UAVs_Route{Count};
   GS_UAVs_Route{Count} = UAVs_Route{Count};
   GS_UAVs_Route{Count}(1,1) = UAVsNames(Route(1));
   GS_UAVRoutesAll{Counter} = GS_UAVs_Route{Count};
   TimeForTaskExecution(Count) = round(WinnerPathTime(Route(1),:));
   TaskList =  Route(:,2:end-1);
   UAVTaskStartTime(Count,:) = TaskCreationTimes(TaskList(1)); % time
at which UAV begins executing task
   UAVTaskEndTime(Count,:) = UAVTaskStartTime(Count,:) +
TimeForTaskExecution(Count); % time at which UAV ends its task

   disp(sprintf('The Task List for UAV %s  is: %s, Tasks start time is
%d and Task delivery time is %.0d'...
       ,num2str(UAVsNames(Route(1))),
num2str(TaskList),UAVTaskStartTime(Count,:), UAVTaskEndTime(Count,:)))
   UAVRouteCord{Count} = [UAVs(Route(1),:); Task(Route(2:end-
1),:);GS_Cord(Route(end),:)];
   ID_UAVs = [ID_UAVs;UAVsNames(Route(1))];

   LatestDeliveryTime = min(Deadlines(Route(2:end-1)));

   GS_UAVRouteCordAll{Counter} = UAVRouteCord{Count};
   TimeHistory(Counter,:) = [UAVTaskStartTime(Count,:),
UAVTaskEndTime(Count,:),LatestDeliveryTime];
    end
end
```

File

```
function [UAVNewLocation, PointMovingTo, LastTaskDone]=
FindUAVnewLocation(UAVCumRouteLenghts, GS_UAVRouteCords,
RouteLenghts,DistTravelled)

global  GS_CommRange UAV_CommRange GS_UAVCommStrength

UAVNewLocation = [];
LastTaskDone = 0; % no task is done
if DistTravelled==0
    UAVNewLocation = GS_UAVRouteCords(1,:); % UAV initial coordinate
     PointMovingTo = 1;
else
for i =1:length(UAVCumRouteLenghts)
    if DistTravelled < UAVCumRouteLenghts(i)
        PointMovingTo = i;

        DistFromPreviousPoint = DistTravelled -
UAVCumRouteLenghts(PointMovingTo-1);
        DirectionVector = GS_UAVRouteCords(PointMovingTo,:)-
GS_UAVRouteCords(PointMovingTo-1,:);

        UAVNewLocation = GS_UAVRouteCords(PointMovingTo-1,:) +
(DistFromPreviousPoint/RouteLenghts(PointMovingTo))*DirectionVector;

     if PointMovingTo > 2
     LastTaskDone = PointMovingTo - 2; % number of tasks done (Note:
not necesarily task IDs)
      end
        break;
    end
end

if isempty(UAVNewLocation)


     DirectionVector = GS_UAVRouteCords(end,:)-GS_UAVRouteCords(end-
1,:);
     DistFromPreviousPoint = norm(GS_UAVRouteCords(end,:)-
GS_UAVRouteCords(end-1,:))-(1-
GS_UAVCommStrength)*(GS_CommRange+UAV_CommRange);

     UAVNewLocation = GS_UAVRouteCords(end-1,:) +
(DistFromPreviousPoint/DistFromPreviousPoint)*DirectionVector; % UAV
has finished its tasks and arrived at the GS
```

```
%       UAVNewLocation = GS_UAVRouteCords(end,:); % UAV has finished its
tasks and arrived at the GS
     LastTaskDone = length(UAVCumRouteLenghts) - 2; % 2, because we
have one UAV and one GS coordinates
     PointMovingTo = length(UAVCumRouteLenghts);
end

end
```

File

```
function RandUAVs =
GenerateRandUAVs(NumberOfGS,NumberOfUAVs,GS_Cord,TangGS_CommRange,UAV_C
ommRangeMin)
Lines = nchoosek(1:NumberOfGS,2); % lines connecting the centre of the
GSs
NumOfLines = size(Lines,1);

NumOfRandUAV = zeros(1,NumOfLines);
NumOfRandUAV(1:NumOfLines-1) =
repmat(floor(NumberOfUAVs/NumberOfGS),1,NumOfLines-1);
NumOfRandUAV(NumOfLines) = NumberOfUAVs -
sum(NumOfRandUAV(1:NumOfLines-1)); %% Last GS pair takes the remaining
UAVs
TempUAVs  = [];
for i=1:NumOfLines
    x1 = GS_Cord(Lines(i,1),1);
    y1 = GS_Cord(Lines(i,1),2);
    x2 = GS_Cord(Lines(i,2),1);
    y2 = GS_Cord(Lines(i,2),2);
    if max(x1,x2)-min(x1,x2)< NumOfRandUAV(i)
    UAV_X = repmat(min(x1,x2),1,NumOfRandUAV(i));
        else
    UAV_X = randsample(min(x1,x2):max(x1,x2),NumOfRandUAV(i));
    end
    UAV_Y  = ((y2- y1)/(x2-x1)).*(UAV_X-x1)+ y1 + 0.1*TangGS_CommRange;
    UAV_Z  = randsample(0.5*TangGS_CommRange:0.4*(TangGS_CommRange +
UAV_CommRangeMin),NumOfRandUAV(i));
    TempUAVs = [TempUAVs ;[UAV_X',UAV_Y',UAV_Z']];
end
RandUAVs = TempUAVs;
FileName =
strcat(num2str(NumberOfUAVs),'RandUAVs_',num2str(UAV_CommRangeMin));
save (FileName, 'RandUAVs')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

UAVsCommWithGS{NumberOfGS} = [];
for g=1:NumberOfGS

DistOfUAVfromGS = sqrt(sum((bsxfun(@minus, GS_Cord(g,:),
RandUAVs)).^2,2));
```

```
% Temp = find(DistOfUAVfromGS<=(TangGS_CommRange + UAV_CommRangeMin));
for j=1:NumberOfUAVs
        if DistOfUAVfromGS(j)<=(TangGS_CommRange + UAV_CommRangeMin)
            UAVsCommWithGS{g} = [UAVsCommWithGS{g},j];
%               break
        end
end
end

disp('COMMUNICATION BETWEEN GSs and UAVs')
for g=1:NumberOfGS
disp(strcat('UAV communicating with GS', num2str(g)))
disp(UAVsCommWithGS{g})
end
```

File

```
function RandTasks = GenerateTask()
UAV_CommRangeMax = 2500;
NumOfTasks = 20;
Xmin= -2*UAV_CommRangeMax;
Xmax= 2*UAV_CommRangeMax;
Ymin= -2*UAV_CommRangeMax;
Ymax = 2*UAV_CommRangeMax;
Zmin = 0;%TangGS_CommRange+UAV_CommRangeMin;
Zmax  = 2*UAV_CommRangeMax;
X_Cords = randsample(Xmin:Xmax, NumOfTasks);
Y_Cords = randsample(Ymin:Ymax, NumOfTasks);
Z_Cords = randsample(Zmin:Zmax, NumOfTasks);
RandTasks = [X_Cords',Y_Cords',Z_Cords'];
FileName = strcat('RandTasks_',num2str(NumOfTasks));
save (FileName, 'RandTasks')
```

File

```
function GS_Bids = GS_Bidding(NumberOfGS, TI, Task,
GS_Cord,TI_CommRange,GS_CommRange,UAV_Speed,DistUnitCost,TimeUnitCost,
TI_TaskCost,Task_id)

% This function computes the the cost for Ground Station(GS) to perform
a given
% task and makes a list of the tasks won by each GS
GS_Bids = [];
Po = [];
Alpha = 0.5; % 0<Alpha<1
for i=1:NumberOfGS
    %      CommCheck = norm(TI- GS_Cord(i,:)); % checking to see if GS
and TI are in communication range
    %      if CommCheck<=(TI_CommRange + GS_CommRange)
%     GS_TaskDist = norm(Task-GS_Cord(i,:)); % distance between the
task and the ground station
```

```
%     GS_TaskDistCost = DistUnitCost*(GS_TaskDist); % (in USD) % Take
note
%     GS_TaskTime = GS_TaskDist/UAV_Speed;
%     TGS_TaskTimeCost = GS_TaskTime*TimeUnitCost;  % (in USD)
%     Po(:,i) =  2*(GS_TaskDistCost + TGS_TaskTimeCost);  %2*norm(Task-
GS_Cord(i,:))*DistUnitCost; % price from the task
    Po(:,i) = 2*(norm(Task-GS_Cord(i,:))*DistUnitCost + norm(Task-
GS_Cord(i,:))*TimeUnitCost/UAV_Speed); % cost in USD

%     GS_Cost = Alpha*Po(:,i) + (GS_TaskDistCost +
TGS_TaskTimeCost)*(1-Alpha); % GS bid for a task

    GS_Cost  = Po(:,i);

    if GS_Cost<=TI_TaskCost(Task_id)
        GS_Bids(:,i) =  GS_Cost; %GS_Bids(:,i); %GS_Cost;
    else
        GS_Bids(:,i) = Inf;
    end
end
% [Po TI_TaskCost(Task_id)]
```

File

```
function PlotTheSpaceWithTheAgents(GS_Cord, GS_CommRange,
NumberOfGS,TI_CommRange, TI,UAVs,NumberOfUAVs)
figure('Name','GS and TI communication ranges, Tasks and UAVs')
plot3(UAVs(:,1),UAVs(:,2),UAVs(:,3),'*')
uavnames = 1:NumberOfUAVs;
text(UAVs(:,1),UAVs(:,2),UAVs(:,3),num2str(uavnames'))
hold on
plot3(GS_Cord(:,1),GS_Cord(:,2),GS_Cord(:,3),'o')
hold on
plot3(TI(:,1),TI(:,2),TI(:,3),'s')
% hold on
% plot3(Task(:,1),Task(:,2),Task(:,3),'')
legend('UAV','GS','TI')
xlabel('x')
ylabel('y')
zlabel('z')
hold on

COL =['r';'b';'g';'m'];
for i =1:NumberOfGS
    % use to plot a circle for the GS communication ranges
    N =256;
t = (0:N)*2*pi/N;
Z = zeros(1,N+1);
plot3(GS_CommRange*cos(t)+GS_Cord(i,1),
GS_CommRange*sin(t)+GS_Cord(i,2),Z,COL(i,:))
hold on
plot(GS_Cord(i,1),GS_Cord(i,2),strcat(COL(i,:),'o'))
```

```
text(GS_Cord(i,1),GS_Cord(i,2),strcat('GS',num2str(i)))
hold on
end

plot3(TI_CommRange*cos(t)+TI(:,1),
TI_CommRange*sin(t)+TI(:,2),Z,COL(4,:))
plot(TI(:,1),TI(:,2),strcat(COL(4,:),'o'))
text(TI(:,1),TI(:,2),'TI')
grid on
hold on
```

File

```
% The function tries to distribute the tasks for a given UAV to other
UAVs
function Robustness(UAV_ID_Eliminate, UAV_Elimination_Time,
UAVRecoveryTime, GS_UAVRoutesAll,...
    TimeHistory, GS_UAVRouteCordAll,
UAVTasksCompletionDurations,AllTaskDeadlines)
% GS_UAVRoutesAll contains all the UAVs' routes (i.e. from UAV->Task-
>GS)
% TimeHistory contains the task start time , delivery time and the
latest
% time the task could be delivered without timeout
% UAV_ID_Eliminate is UAV to be eliminated due to injury
% UAVRecoveryTime > UAV_Disconnection_Time
global TI_CostScalar InitNumberOfTask CycleAuctionCloseTime
tH_Robust = AllTaskDeadlines;
% Making sure the  UAVRecoveryTime and UAV_Disconnection_Time are valid
while (UAVRecoveryTime < UAV_Elimination_Time)
    disp('UAVRecoveryTime must be more than UAV_Elimination_Time and
both must be greater than zero')
    UAV_Elimination_Time = input('Please enter UAV_Elimination_Time:
');
    UAVRecoveryTime = input('Please enter the UAVRecoveryTime:   ');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global GS_CommRange NumberOfGS TI_CommRange TI UAV_CommRange GS_History
Task UAV_Speed WinnerUAV_RouteOld TangGS_CommRange
GS_UAVCommStrength...
    DistUnitCost TimeUnitCost NumberOfUAVs TaskAuctionTimesOld GS_Cord
Up AuctionTimeStep TI_id WinnerPathTimeOld GS_UAVList UAVs
 global SellAndBuyBids M_BuyBid SellAndBuyWinners SellAndBuyWinnerBids
TI_TaskCost GS_jUAV_SBids TaskCreationTime Scenario Gamma OriginalUAV
OriginalGS

% Note: Gamma is a fraction of latest delivery time of task by
eliminated UAV
NumberOfUAVsAll = NumberOfUAVs;
% Find the UAV to be disconnected
UAVs_CurrentLocations = zeros(NumberOfUAVsAll,3);
```

```
UAVsWithTasks = [];
RemainingPathTime = UAVTasksCompletionDurations;
%zeros(NumberOfUAVsAll,1);
DeadTimes = zeros(NumberOfUAVsAll,1);
UAV_TaskListRemaining = [];  % tasks unvisisted prior to the UAV's
disconnection
 NumOfTasks = zeros(NumberOfUAVsAll,1);
for j = 1: length(GS_UAVRoutesAll)
        if ~isempty(GS_UAVRoutesAll{j})
    NumOfTasks(j,:) = length(GS_UAVRoutesAll{j}) - 2; % stores the
number of tasks in the tasklist of each UAV
%     UAVsWithTasks   = [UAVsWithTasks; GS_UAVRoutesAll{j}(1)];  % UAV
IDS basically

    DeadTimes(GS_UAVRoutesAll{j}(1),:) =
min(AllTaskDeadlines(GS_UAVRoutesAll{j}(2:end-1))); % stores the time
for the last auctioned tasks for each route
    %     GS_id(j,:) = GS_UAVRoutesAll{j}(end);
        else
          UAVs_CurrentLocations(j,:) =  UAVs(j,:);  % UAV  does not
move if it wins no task
        end
end



% Checking to make sure the UAV entered is valid and has some task(s)
ValidUAV_IDChecker = 0;
while (ValidUAV_IDChecker==0)
%     ROUTEidToDelete = find( UAVsWithTasks == UAV_ID_Eliminate);
    if isempty(GS_UAVRoutesAll{UAV_ID_Eliminate})
        disp('The UAV you have entered does not exist or has no task;
taking it out has no impact on the system.')
        UAV_ID_Eliminate = input('Please enter another UAVid (any
number from 1 through 9):   '); % There are 9 UAVs
        ValidUAV_IDChecker = 0;
    else
        ValidUAV_IDChecker = 1;
        ROUTEidToDelete = UAV_ID_Eliminate;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RouteToDelete = GS_UAVRoutesAll{ROUTEidToDelete}; % contains the UAV,
its tasks, and the GS to deliver to

TasksForUAV_ID_Eliminate = RouteToDelete(2:end-1); % Task IDs from the
eliminated UAV
% NumOfTasks = length(TasksForUAV_ID_Eliminate);
LatestTasksDeliveryTime = TimeHistory(ROUTEidToDelete, 4);
TimeToReAuctionTasks = round((1 + Gamma)*LatestTasksDeliveryTime); %
time to restart auctioning taks from the eliminated UAV
```

```
TimeToBeginReAuctionTasks = TimeToReAuctionTasks;
% Compute the new location of the UAV prior to its elimination (i.e.
deletion)
DistByUAV_ID_Eliminate = CalUAVDistance(UAV_Speed,
TimeHistory(ROUTEidToDelete,1),UAV_Elimination_Time,TimeHistory(ROUTEid
ToDelete,2));
Temp1{1} = GS_UAVRouteCordAll{ROUTEidToDelete};  % take note of
{RouteToDelete};
Temp2{1} = GS_UAVRoutesAll{ROUTEidToDelete};     % take note of
{RouteToDelete};

[RouteLenghts,UAVCumRouteLenghts] = ComputeSegmentDistance(Temp1,
Temp2);

Temp3  = GS_UAVRouteCordAll{ROUTEidToDelete};    % take note of
{RouteToDelete};
Temp4 = UAVCumRouteLenghts{1};
Temp5 = RouteLenghts{1};
[UAVNewLocation1, PointMovingTo, LastTaskDone] =
FindUAVnewLocation(Temp4, Temp3, Temp5,DistByUAV_ID_Eliminate);

% Compute UAV new location after the UAV has recovered and moved
towards
% the remaining tasks
if UAVRecoveryTime < TimeToReAuctionTasks
    EarliestExpDeliveryTime =
TimeHistory(ROUTEidToDelete,2)+(UAVRecoveryTime-UAV_Elimination_Time);
    DistByUAV_ID_Eliminate2 = CalUAVDistance(UAV_Speed,
UAVRecoveryTime, TimeToReAuctionTasks,EarliestExpDeliveryTime);
    TotalDist = DistByUAV_ID_Eliminate + DistByUAV_ID_Eliminate2;
    [UAVNewLocation2, PointMovingTo, LastTaskDone] =
FindUAVnewLocation(Temp4, Temp3, Temp5, TotalDist);
    UAVs_CurrentLocations(UAV_ID_Eliminate, :) = UAVNewLocation2;
else
    UAVs_CurrentLocations(UAV_ID_Eliminate, :) = UAVNewLocation1;
    UAVNewLocation2 = UAVNewLocation1;
end


UAV_GS_CommChecker = UAV_CommRange + GS_CommRange;
Store1 = GS_UAVRouteCordAll{ROUTEidToDelete}(end,:); % GS co-ordinates
if (LastTaskDone>0) && (LastTaskDone < NumOfTasks(ROUTEidToDelete))
&&((norm(Store1- UAVNewLocation2)<=(UAV_GS_CommChecker))) % check
communication
    TasksUnDelivered = TasksForUAV_ID_Eliminate(LastTaskDone+1:end); %
these tasks have to be re-auctioned to the existing UAVs
elseif LastTaskDone < 1
    TasksUnDelivered = TasksForUAV_ID_Eliminate;
elseif (LastTaskDone==NumOfTasks(ROUTEidToDelete)) &&
((norm(GS_UAVRouteCordAll{ROUTEidToDelete}(end,:)-
UAVNewLocation2)<=(UAV_GS_CommChecker)))
    TasksUnDelivered = []; % all tasks are delivered.
else
```

```
    TasksUnDelivered = TasksForUAV_ID_Eliminate;
end

TasksDelivered =
TasksForUAV_ID_Eliminate(1:length(TasksForUAV_ID_Eliminate)-
length(TasksUnDelivered));
%TasksForUAV_ID_Eliminate(length(TasksUnDelivered)+1:end);

% Display some statistics for the disconnected UAV
disp('Records for the disconnected UAV:')

disp(sprintf('UAV Disconnected = %d', UAV_ID_Eliminate))
disp(sprintf('Time it got eliminated = %d', UAV_Elimination_Time))
disp(sprintf('Time it recovered = %d', UAVRecoveryTime))
disp(sprintf('Its latest task delivery time = %d',
LatestTasksDeliveryTime))
disp('Tasks in its tasklist are:')
disp(TasksForUAV_ID_Eliminate)
disp('Tasks delivered are:')
if isempty(TasksDelivered)
    disp('None')
else
    disp(TasksDelivered)
end


disp('Tasks undelivered are:')
if isempty(TasksUnDelivered)
    disp('none')
else
    disp(TasksUnDelivered)
end
disp(sprintf('\n'))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




% Computing the remaining tasks to be done by the existing UAVs at the
point of reauctioning of the tasks undelivered by the disconnected UAV

% Initialize the task lists
 UAV_TaskListRemaining{NumberOfUAVsAll} = [];
%  UAV_TaskListRemaining{UAV_ID_Eliminate} = TasksUnDelivered;
 RemainingPathTime = zeros(NumberOfUAVsAll,1); %initialization
%  WinnerUAV_Route{NumberOfUAVsAll} = [];
 UAV_TaskListCompleted{NumberOfUAVsAll} = [];
  UAV_TaskListCompleted{UAV_ID_Eliminate} = TasksDelivered;
%%%%%%%%%%%%%%%%%%%%
```

```
% Find the remaining tasks for UAVs with tasks
 TaskAuctionTimes = zeros(InitNumberOfTask,1);
 WinnerPathTimeOld = zeros(NumberOfUAVsAll,1);
 last =[];
for j = 1: NumberOfUAVsAll %length(UAVsWithTasks)
    if j~=UAV_ID_Eliminate % avoid doing for eliminated UAV
        if ~isempty(GS_UAVRoutesAll{j})
            Distance = CalUAVDistance(UAV_Speed, TimeHistory(j,1),
TimeToReAuctionTasks,TimeHistory(j,2));

            Temp6{1} = GS_UAVRouteCordAll{j};
            Temp7{1} = GS_UAVRoutesAll{j};

            [RouteLenghts,UAVCumRouteLenghts] =
ComputeSegmentDistance(Temp6, Temp7);
%               [j,Distance,UAVCumRouteLenghts{1}(end)]
            [UAVs_CurrentLocations(j,:), PointMovingTo, LastTaskDone] =
FindUAVnewLocation(UAVCumRouteLenghts{1}, GS_UAVRouteCordAll{j},
RouteLenghts{1},Distance);

            if LastTaskDone>0 && LastTaskDone <= NumOfTasks(j)
                UAV_TaskListCompleted{j} =
GS_UAVRoutesAll{j}(2:LastTaskDone+1);
                GS_UAVRoutesAll{j}(2:LastTaskDone+1) = []; % delete
tasks IDs visited
                UAV_TaskListRemaining{j} = GS_UAVRoutesAll{j}(2:end-1);
% UAV_TaskListRemaining contains remaining task to be done
                GS_UAVRouteCordAll{j}(2:LastTaskDone+1, :) = []; %
delete the coordinates of tasks visited
                WinnerPathTimeOld(j) =
min(TimeToReAuctionTasks,TimeHistory(j,2))-TimeHistory(j,1); % time
already travelled.
%                  last = [last; [j  NumOfTasks(j) LastTaskDone
UAV_TaskListRemaining{j}]]
            elseif LastTaskDone==0
                UAV_TaskListRemaining{j} = GS_UAVRoutesAll{j}(2:end-1);

                 UAV_TaskListCompleted{j} = [];
%                  WinnerPathTimeOld(j) = TimeToReAuctionTasks-
TimeHistory(j,1); % time elapsed.
            end

            if TimeToReAuctionTasks < TimeHistory(j,2) % less than
earliest delivery time
                RemainingPathTime(j,:) =  TimeHistory(j,2) -
TimeToReAuctionTasks; % time remaining to be travelled.
            end

        else
            UAV_TaskListRemaining{j} = [];
        end
    end

end


%  WinnerPathTimeOld(4)
RemainingPathTime(UAV_ID_Eliminate,:) =
TimeHistory(UAV_ID_Eliminate,2) - UAV_Elimination_Time; % time
remaining to be travelled.

UAV_TaskListRemaining{UAV_ID_Eliminate} = []; % the remaining tasks for
the deleted UAV are the undelivered tasks
WinnerPathTimeOld( UAV_ID_Eliminate) = UAV_Elimination_Time -
TimeHistory(UAV_ID_Eliminate,1); % time already travelled.
% WinnerPathTimeOld( UAV_ID_Eliminate) =
TimeHistory(UAV_ID_Eliminate,2) - TimeHistory(UAV_ID_Eliminate,1);

disp(sprintf('TIME AT WHICH THE REAUCTION OF TASKS FROM THE
DISCONNECTED UAV BEGINS: %s',
num2str(round(TimeToBeginReAuctionTasks))))
disp(sprintf('THE REMAINING TASKS TO BE DONE BY ALL THE OTHER UAV JUST
BEFORE THE REAUCTION OF TASKS FROM THE DISCONNECTED, UAV%s:',
num2str(UAV_ID_Eliminate)))
for i=1:NumberOfUAVsAll
    if i~=UAV_ID_Eliminate % avoid doing for eliminated UAV
        if isempty(UAV_TaskListRemaining{i})
            disp(sprintf('The remaining tasks for UAV %s  are: %s',
num2str(i), 'None'))
            GS_UAVRoutesAll{i} = [];
        else
            disp(sprintf('The remaining tasks for UAV %s  are: %s',
num2str(i), num2str(UAV_TaskListRemaining{i})))
            TaskAuctionTimes(UAV_TaskListRemaining{i}) =
TimeToReAuctionTasks + (0:length(UAV_TaskListRemaining{i})-
1).*AuctionTimeStep;
        end
    end
end
GS_UAVRoutesAll{UAV_ID_Eliminate} = []; % deleted UAV has no route
until it recovers and wins some tasks.
disp(sprintf('\n'))


disp('UAVs current coordinates just before the reauction of tasks from
the eliminated UAV:')
disp(round(UAVs_CurrentLocations))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% The reauctioning of the undelivered tasks from the disconnected UAVs

PreviousGSTaskAuctionTimes = TaskAuctionTimes; % keeps the times the
tasks were originally sold to the UAVs.
for gs=1:NumberOfGS
    TempGS_UAVlist = GS_UAVList{gs};
    GS(gs).UAV_Route = GS_UAVRoutesAll;
    TempGS_UAVlist(find(TempGS_UAVlist==UAV_ID_Eliminate))=[];
    GS_UAVList{gs} = TempGS_UAVlist; % deleted UAV eliminated
end
WinnerUAV_Route = GS_UAVRoutesAll; % New UAV routes ( initialization )
WinnerPathTime  = RemainingPathTime; % initialization

UAV_TaskListNew = UAV_TaskListRemaining;

if isempty(TasksUnDelivered)
    ReauctionCloseTime = 0;
    disp('All the tasks have already been delivered by the disconnected
UAV before the reauctioning time')
else
    Start_t = TimeToBeginReAuctionTasks - AuctionTimeStep; %
initialization
%     GS_UAVList{NumberOfGS} =  []; % initialization
    WinnerUAVsIDs = [];
    UAVsBids = [];
    CycleCount = 0;

    UnsoldTaskIDs = TasksUnDelivered;
    InitReAuctioneTimes(UnsoldTaskIDs) = TimeToReAuctionTasks +
(0:length(UnsoldTaskIDs)-1).*AuctionTimeStep; % For initial undelivered
tasks
    while(1) % do until all tasks are re-auctioned
        CycleCount = CycleCount + 1;
        NumberOfTask(CycleCount) = length(UnsoldTaskIDs);
        UnsoldTaskIDsTemp = [];

        for i = 1: NumberOfTask(CycleCount)
            TaskID = UnsoldTaskIDs(i);
            NewTask_id = TaskID;
            Start_t = Start_t + AuctionTimeStep;  % note, Start_t
starts from TimeToReAuctionTasks and it is basically task re-auctioned
time
            TaskAuctionTimes(TaskID) = Start_t; %

            AllTaskDeadlines(TaskID,:) =  AllTaskDeadlines(TaskID) +
Start_t - max(UAV_Elimination_Time, TimeHistory(ROUTEidToDelete,1));
% task deadlines have been readjusted
            NewTaskDeadline =  AllTaskDeadlines(TaskID);

            %Compute the cost of the task to be auctioned
            t = TaskAuctionTimes(TaskID)-InitReAuctioneTimes(TaskID);
            TI_Po(:,TaskID) = 2*(norm(TI-Task(TaskID,:))*DistUnitCost +
norm(TI-Task(TaskID,:))*TimeUnitCost/UAV_Speed); % initial price from
the task initiator

            TI_TaskCost(TaskID,:) = TI_CostScalar*TI_Po(:,TaskID)*(1 +
Up*t/tH_Robust(TaskID));
            GS_Bids{TaskID} = GS_Bidding(NumberOfGS, TI,
Task(TaskID,:),GS_Cord,TI_CommRange,GS_CommRange,UAV_Speed,DistUnitCost
,TimeUnitCost,TI_TaskCost,TaskID); % sell bids

            % Computing the UAVs Biddings and the optimize path as well
            UAVwinnerBids  = zeros(1,NumberOfGS);
            WinnerUAVnames = zeros(1,NumberOfGS);
            GSs_UAVs_BidPairs = zeros(NumberOfGS,4);
            GSnames = 1:NumberOfGS;

            AllUAVNames =  1: NumberOfUAVsAll; % initialization
            if UAVRecoveryTime < Start_t
                for gs = 1:NumberOfGS
                    UAV_Bids{TaskID}(UAV_ID_Eliminate) = inf;
                    GS(gs).UAV_Route{UAV_ID_Eliminate} = [];
                    GS(gs).PathTime(UAV_ID_Eliminate,:) = inf;
                end
                AllUAVNames(UAV_ID_Eliminate)=[];  % to prevent the
deleted UAV from bidding
            end

            for gs = 1:NumberOfGS
                UAV_Bid = [];
                for j=1: length(AllUAVNames) %NumberOfUAVsAll
                    UAVID =  AllUAVNames(j);

%                     if UAVID==UAV_ID_Eliminate && UAVRecoveryTime <
Start_t
%                         UAV_Bids{TaskID}(UAVID) = inf;
GS(gs).UAV_Route{UAVID} = []; GS(gs).PathTime(UAVID,:) = inf; % to
prevent the deleted UAV from bidding
%                     else

                        if isempty(UAV_TaskListNew{UAVID})
                            UAV_TaskListPrevious = [];
                            NewLeastPathTime = 0;
                            OldTaskDeadline = 0;
                            oldGS_ID = 0;

[UAV_Bids{TaskID}(UAVID),GS(gs).UAV_Route{UAVID},GS(gs).PathTime(UAVID)
] =  UAV_Bidding(UAV_TaskListPrevious, TI,
TI_id,UAVID,UAVs_CurrentLocations,Task,UAV_Speed,TI_TaskCost,...

TaskID,DistUnitCost,TimeUnitCost,NewLeastPathTime,NewTaskDeadline,...
```

```
OldTaskDeadline,GS_CommRange,UAV_CommRange,
TaskAuctionTimes,GS_Cord,gs,oldGS_ID);
                       elseif ~isempty(UAV_TaskListNew{UAVID})
                           UAV_TaskListPrevious =
UAV_TaskListNew{UAVID};
                           NewLeastPathTime = WinnerPathTime(UAVID);
                           %                         OldTaskDeadline =
DeadTimes(UAVID);
                           oldGS_ID = WinnerUAV_Route{UAVID}(end);
%                           bb =  GS(gs).UAV_Route{UAVID}(end-1)
                           OldTaskDeadline =
min(AllTaskDeadlines(UAV_TaskListNew{UAVID}));
%AllTaskDeadlines(GS(gs).UAV_Route{UAVID}(end-1));

[UAV_Bids{TaskID}(UAVID),GS(gs).UAV_Route{UAVID},GS(gs).PathTime(UAVID)
] =  UAV_Bidding(UAV_TaskListPrevious, TI,
TI_id,UAVID,UAVs_CurrentLocations,Task,UAV_Speed,TI_TaskCost,...

TaskID,DistUnitCost,TimeUnitCost,NewLeastPathTime,NewTaskDeadline,...

OldTaskDeadline,GS_CommRange,UAV_CommRange,
TaskAuctionTimes,GS_Cord,gs,oldGS_ID);
                       end

%                     end
                   end
                   [UAVwinnerBids(gs), WinnerUAVnames(gs)] =
min(UAV_Bids{TaskID});
               end

               GSs_UAVs_BidPairs = [GSnames',round(GS_Bids{TaskID}'),
WinnerUAVnames',round(UAVwinnerBids')];
               disp('GSs and UAVs bid pairs:')
               disp('           GS       GS_Bid       UAV      UAV_Bid')
               disp(GSs_UAVs_BidPairs)
               %       UAV_Bids{i}


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

               % Matching the GS_Bids (i.e. sell bids) and the UAV_Bids
(i.e. buy
               % bids
               Sell_in_Bids = [];
               Buy_in_Bids  = [];
               TransactionSet = [];
               UAV_Indx = [];
               GS_Indx = [];
               UAV_Indx1 = [];
               GS_Indx1 = [];

               % Determine eligible sellers and buyers
```

```
            for m=1:NumberOfGS
                if GS_Bids{TaskID}(m)~=Inf && UAVwinnerBids(m)~=Inf
                    TransactionSet = [TransactionSet;
[GS_Bids{TaskID}(m),UAVwinnerBids(m)]];
                    UAV_Indx1 = [UAV_Indx1; WinnerUAVnames(m)];
                    GS_Indx1 = [GS_Indx1; m];
                end
            end
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            NumOfSellers = size(TransactionSet,1);
            if ~isempty(TransactionSet) % There are seller(s) and
buyer(s)
                [ignore1 OrigIndx] = min(TransactionSet(:,1));
                OriginalUAVtemp = UAV_Indx1(OrigIndx);
                OriginalGStemp  = GS_Indx1(OrigIndx);

                Rank = sort([TransactionSet(:,1),
TransactionSet(:,2)],'ascend');
                M_BuyBid(TaskID,:) = Rank(NumOfSellers);  % Mth bid
                M_nextBuyBid(TaskID,:) = Rank(NumOfSellers+1); % M+1st
bid

                [Sell_Bids{TaskID}, GS_IndxTemp{TaskID}] =
min(TransactionSet(:,1));
                GS_Indx(1)   = GS_Indx1(GS_IndxTemp{TaskID}); % winner
GS(i.e. GS with minimum bid)
                UAV_Indx(1)  = UAV_Indx1(GS_IndxTemp{TaskID});  % winner
UAV (i.e. UAV with minimum bid)

                Buy_Bids{TaskID} =
TransactionSet(GS_IndxTemp{TaskID},2);
                %                 GS_Winner =
GS_Indx1(GS_IndxTemp{TaskID}); % actual winner
                                GS_jUAV_SBids(TaskID) =
GS_Bids{TaskID}(GS_Indx(1)); % cost of GS to which task is delivered
%                 GS_TaskDelivered(TaskID) = GS_Indx(1);

                %%%Determine the winner
                UAV_TaskListNew{UAV_Indx(1)}    =
[UAV_TaskListNew{UAV_Indx(1)}, TaskID];
                GS_UAVList{GS_Indx(1)}       = [GS_UAVList{GS_Indx(1)},
UAV_Indx(1)];
                WinnerUAV_Route{UAV_Indx(1)} =
GS(GS_Indx(1)).UAV_Route{UAV_Indx(1)};
                WinnerPathTime(UAV_Indx(1))   =
GS(GS_Indx(1)).PathTime(UAV_Indx(1));
                SellAndBuyBids(TaskID,:)      =
SellAndBuyWinnerBids(TaskID,:); %winning sell and buy bids pair
                SellAndBuyWinners(TaskID,:) =  [GS_Indx(1),
UAV_Indx(1)]; % UAVs and GSs that win the tasks
                SellAndBuyWinnerBids(TaskID,:) = [Sell_Bids{TaskID},
Buy_Bids{TaskID}];
```

```
                SellAndBuyBids(TaskID,:) =
SellAndBuyWinnerBids(TaskID,:);

                OriginalUAV(TaskID) = OriginalUAVtemp; % UAV with
minimum cost
                OriginalGS(TaskID)  = OriginalGStemp;  % GS paired with
UAV with minimu cost
            else % there are no sellers and/or buyers
                UnsoldTaskIDsTemp = [UnsoldTaskIDsTemp,TaskID];
            end
        end
        TransactionSet = [];

        % Check if all tasks are sold
        if isempty(UnsoldTaskIDsTemp)
            break; % break from the while loop
        else
            UnsoldTaskIDs = UnsoldTaskIDsTemp; % tasks unsolds
        end
%         UnsoldTaskIDs
    end
    ReauctionCloseTime =  Start_t;  % time at which the reauction of
the tasks from the disconnected UAV is over
end


%Display the final routes for UAVs which won some reauctioned tasks
UAVTasksStartTimes = [];

% disp('THE ROUTES FOR THE BUSY (THOSE WHICH HAD TASKS) UAVs FROM THE
TIME THE UNDELIVERED TASKS ARE REAUCTIONED:')
% for i=1: NumberOfUAVsAll
%     if ~isempty(WinnerUAV_Route{i})
%         %      UAVTasksStartTimes = [UAVTasksStartTimes;
GSTaskAuctionedTimes(WinnerUAV_Route{i}(2))];
%         disp(WinnerUAV_Route{i})
%         disp(sprintf('Tasks begin at %s,  delivered at %s',
num2str(TaskAuctionTimes(WinnerUAV_Route{i}(2))),...
%             num2str(round(WinnerPathTime(i)+
TaskAuctionTimes(WinnerUAV_Route{i}(2))))))
%         disp(sprintf('\n'))
%     end
% end

TaskAuctionTimesOld(TasksUnDelivered) =
TaskAuctionTimes(TasksUnDelivered); % Replace the aution times for the
re-autioned tasks

 %%% Detail Results

TaskNames = 1:InitNumberOfTask;
% Calculate the profit using the clearing price (i.e. M_BuyBid)
```

```
SellerProfit1  = 0.5*(SellAndBuyWinnerBids(:,1)-
SellAndBuyWinnerBids(:,2) + TI_TaskCost - SellAndBuyWinnerBids(:,1) -(
GS_jUAV_SBids-SellAndBuyWinnerBids(:,1)));
BuyerProfit1 = SellerProfit1;
OriginalGS_Profit = 0.5*(GS_jUAV_SBids - SellAndBuyWinnerBids(:,1)); %
profit of original GS that was paired with the winner UAV

AvgSellBid = mean(SellAndBuyWinnerBids(:,1));
AvgBuyBid = mean(SellAndBuyWinnerBids(:,2));
% AvgGS_DelivererBid = mean(GS_jUAV_SBids);
AvgOriginalGS_Bid = mean(GS_jUAV_SBids);
AllAvgBids = [AvgOriginalGS_Bid,AvgSellBid,AvgBuyBid];

AvgClearingPrice = mean(M_BuyBid);
AvgOriginalGS_Profit = mean(OriginalGS_Profit);
AvgSellProfit = mean(SellerProfit1);
AvgBuyProfit = mean(BuyerProfit1);
% AvgGS_DelivererProfit = mean(GS_Profit);
AllAvgProfits = [AvgClearingPrice,AvgOriginalGS_Profit,AvgSellProfit,
AvgBuyProfit];


% Compute the total profits for all GS

GS_TotalProfits = zeros(NumberOfGS,1);
GS_Names = 1:NumberOfGS;
for g = 1:NumberOfGS
    GS_Originalfrequency = find(OriginalGS==g); % how often GS
originally paired with the winner UAV
    GS_Sellfrequency = find(SellAndBuyWinners(:,1)==g); % how often GS
got tasks delivered to it

    if ~isempty(GS_Originalfrequency)
    GS_TotalProfits(g) = GS_TotalProfits(g)+
sum(OriginalGS_Profit(GS_Originalfrequency));
    end
        if ~isempty(GS_Sellfrequency)
    GS_TotalProfits(g) = GS_TotalProfits(g)+
sum(SellerProfit1(GS_Sellfrequency));
        end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Compute the total for all UAVs

UAV_TotalProfits = zeros(NumberOfUAVs,1);
UAV_Names = 1:NumberOfUAVs;
for u = 1:NumberOfUAVs
    UAV_Buyfrequency = find(SellAndBuyWinners(:,2)==u); % how often UAV
buys tasks from Winner GS
    if ~isempty(UAV_Buyfrequency)
    UAV_TotalProfits(u) = UAV_TotalProfits(u)+
sum(BuyerProfit1(UAV_Buyfrequency));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Write the results into excel sheeet
TITLE = {'TaskName','Creation Time(sec)','Auction
Time(sec)','Deadline(sec)','Task Cost($) (from TI)', 'Original Seller
Bid($)','Winning Seller Bid($)', 'Winning Buyer Bid($)',...
    'Original_GS', 'Seller(GS)','Buyer(UAV)', 'Clearing Price($)',
'Original_GS Profit($)', 'Seller Profit($)','Buyer Profit($)'};

xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
TITLE, 'A1:O1')
BuySellRecords = [TaskNames',TaskCreationTime',TaskAuctionTimesOld,
AllTaskDeadlines,TI_TaskCost,GS_jUAV_SBids,SellAndBuyWinnerBids(:,1),Se
llAndBuyWinnerBids(:,2),...
OriginalGS, SellAndBuyWinners(:,1), SellAndBuyWinners(:,2),M_BuyBid,
OriginalGS_Profit,SellerProfit1,BuyerProfit1];
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
BuySellRecords, strcat('A2:O',num2str(InitNumberOfTask+1)))
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
{'AVERAGE'},
strcat('E',num2str(InitNumberOfTask+2),':','E',num2str(InitNumberOfTask
+2)))
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
AllAvgBids,
strcat('F',num2str(InitNumberOfTask+2),':','H',num2str(InitNumberOfTask
+2)))
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
AllAvgProfits,
strcat('L',num2str(InitNumberOfTask+2),':','O',num2str(InitNumberOfTask
+2)))

SubTitle = {'GS', 'GS Total Profit($)'};
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
SubTitle,
strcat('I',num2str(InitNumberOfTask+5),':','J',num2str(InitNumberOfTask
+5)))
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
GS_Names',
strcat('I',num2str(InitNumberOfTask+6),':','I',num2str(InitNumberOfTask
+6+NumberOfGS-1)))
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
GS_TotalProfits,
strcat('J',num2str(InitNumberOfTask+6),':','J',num2str(InitNumberOfTask
+6+NumberOfGS-1)))


SubTitle = {'UAV', 'UAV Total Profit($)'};
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
SubTitle,
strcat('M',num2str(InitNumberOfTask+5),':','N',num2str(InitNumberOfTask
+5)))
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
UAV_Names',
strcat('M',num2str(InitNumberOfTask+6),':','M',num2str(InitNumberOfTask
+6+NumberOfUAVs-1)))
xlswrite(strcat('BuySellResults_Robust_Scenario',num2str(Scenario)),
UAV_TotalProfits,
strcat('N',num2str(InitNumberOfTask+6),':','N',num2str(InitNumberOfTask
+6+NumberOfUAVs-1)))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Display the UAV tasklists
% TimeHistory = zeros(NumberOfUAVs,4); %initialization
GS_UAVRouteCordAll{NumberOfUAVs} = []; % initialization
GS_UAVRouteCordAllTemp{NumberOfUAVs} = []; % initialization  to hold
coordinates without UAV current locations
UAVTasksCompletionDurations  = zeros(NumberOfUAVs,1);
GS_UAVRoutesAll{NumberOfUAVs} = [];
TaskCompletionTimes = [];
TaskCompletionDists = [];
 WinnerUAV_RouteNew =  WinnerUAV_Route;
UAV_TaskList{NumberOfUAVsAll} = []; % initialization
 WinnerUAV_Route{NumberOfUAVsAll} = [];
 GS = 0;
 DelayTimes = zeros(1,NumberOfUAVsAll);
 DelayTimes(UAV_ID_Eliminate) = UAVRecoveryTime-UAV_Elimination_Time;
 disp('RECORDS FOR UAVs WHICH WON REAUCTIONED TASK(S): ')
 disp('(Note: Those UAVs which did not win any reacuctioned tasks have
the same records as shown before)')
 for UAVname=1:NumberOfUAVsAll
    UAV_TaskList{UAVname} = [UAV_TaskListCompleted{UAVname},
UAV_TaskListNew{UAVname}];
    if ~isempty(WinnerUAV_RouteNew{UAVname})
        GS = WinnerUAV_RouteNew{UAVname}(end);
    elseif ~isempty(WinnerUAV_RouteOld{UAVname})
        GS = WinnerUAV_RouteOld{UAVname}(end);
    end

    if ~isempty(UAV_TaskList{UAVname})&&
~isempty(UAV_TaskListNew{UAVname})
        WinnerUAV_Route{UAVname} = [UAVname,
UAV_TaskListCompleted{UAVname},WinnerUAV_RouteNew{UAVname}(2:end-
1),GS]; % from beginning to end
        disp(strcat('UAV',num2str(UAVname),':'))
        disp('TASKLIST:'),
        disp('Task completed before REAUCTION:')
        disp(UAV_TaskListCompleted{UAVname})
        disp('Task completed after REAUCTION:')
        disp(UAV_TaskListNew{UAVname})
        disp('Minimum Route from start to end:'),
disp(WinnerUAV_Route{UAVname}) % from start to after reauction
```

```
        disp('StartTime  DeliveryTime  PathTime LatestDeliveryTime')

        DeliveryTime = round(ReauctionCloseTime) +
round(WinnerPathTime(UAVname));
%       DeliveryTime  = round(CycleAuctionCloseTime)+
round(WinnerPathTimeOld(UAVname))+ round(WinnerPathTime(UAVname));
        LatestDeliveryTime =
min(AllTaskDeadlines(UAV_TaskList{UAVname}));
        UAVPathTime =  round(WinnerPathTimeOld(UAVname))+
round(WinnerPathTime(UAVname)); % time taken by UAV from start to end
        TimeHistory(UAVname,:) =
round([CycleAuctionCloseTime(end),DeliveryTime, UAVPathTime,
LatestDeliveryTime]);
%

%         DeliveryTime =
round(TaskAuctionTimes(WinnerUAV_Route{UAVname}(2)))+
max(round(WinnerPathTime(UAVname)), round(WinnerPathTimeOld(UAVname)))+
DelayTimes(UAVname);
%         LatestDeliveryTime =
min(AllTaskDeadlines(UAV_TaskList{UAVname}));
%         UAVPathTime = DeliveryTime -
round(TaskAuctionTimes(WinnerUAV_Route{UAVname}(2)))-
DelayTimes(UAVname);
%         TimeHistory(UAVname,:) =
round([TaskAuctionTimes(WinnerUAV_Route{UAVname}(2)),DeliveryTime,
UAVPathTime, LatestDeliveryTime]);
%

        disp(TimeHistory(UAVname,:))
        GS_UAVRouteCordAll{UAVname} =
[UAVs(UAVname,:);Task(UAV_TaskListCompleted{UAVname},:);
UAVs_CurrentLocations(UAVname,:);...

Task(WinnerUAV_RouteNew{UAVname}(2:end-1),:);GS_Cord(GS,:)];
         GS_UAVRouteCordAllTemp{UAVname} =
[UAVs(UAVname,:);Task(UAV_TaskListCompleted{UAVname},:);...

Task(WinnerUAV_RouteNew{UAVname}(2:end-1),:);GS_Cord(GS,:)];
        TaskNames = WinnerUAV_Route{UAVname}(2:end-1);
        UAVTasksCompletionDurations(UAVname) = WinnerPathTime(UAVname);
        GS_UAVRoutesAll{UAVname} = WinnerUAV_Route{UAVname};
        TaskCompletionTimes = [TaskCompletionTimes,
WinnerPathTime(UAVname) + WinnerPathTimeOld(UAVname)];
        figure('Name', strcat('Robustness:  Path for executing task by
UAV', num2str(UAVname)))

plot3(GS_UAVRouteCordAll{UAVname}(:,1),GS_UAVRouteCordAll{UAVname}(:,2)
,GS_UAVRouteCordAll{UAVname}(:,3),'*-')

text(GS_UAVRouteCordAll{UAVname}(1,1),GS_UAVRouteCordAll{UAVname}(1,2),
GS_UAVRouteCordAll{UAVname}(1,3), strcat('UAV',num2str(UAVname)))
```

```
        text(GS_UAVRouteCordAllTemp{UAVname}(2:end-
1,1),GS_UAVRouteCordAllTemp{UAVname}(2:end-
1,2),GS_UAVRouteCordAllTemp{UAVname}(2:end-1,3), num2str(TaskNames'))

text(UAVs_CurrentLocations(UAVname,1),UAVs_CurrentLocations(UAVname,2),
UAVs_CurrentLocations(UAVname,3), strcat('UAV',num2str(UAVname)))

text(GS_UAVRouteCordAll{UAVname}(end,1),GS_UAVRouteCordAll{UAVname}(end
,2),GS_UAVRouteCordAll{UAVname}(end,3), strcat('GS',num2str(GS)))
        grid on
        xlabel('x')
        ylabel('y')
        zlabel('z')
    else

         TaskCompletionTimes = [TaskCompletionTimes,
WinnerPathTime(UAVname) + WinnerPathTimeOld(UAVname)];
    end
 end

TaskCompletionDists = UAV_Speed.*TaskCompletionTimes;
% Display the GS and the UAVs  it solds tasks to
for GS=1:NumberOfGS
    if ~isempty(GS_UAVList{GS})
        disp(strcat('UAV LIST FOR GS',num2str(GS),':'))
        disp(GS_UAVList{GS})
    end
end

disp('    AvgDist(m)   AvgTime(s)  Dist_std(m)  Time_std(s)')
disp(round([mean(TaskCompletionDists), mean(TaskCompletionTimes),
std(TaskCompletionDists), std(TaskCompletionTimes)]))

% plot the space just before negotiation
% UAVs_CurrentLocations
PlotTheSpaceWithTheAgents(GS_Cord, GS_CommRange,
NumberOfGS,TI_CommRange, TI,UAVs_CurrentLocations,NumberOfUAVsAll)
```

File

```
function [UAV_Bid,UAV_Route,PathTime] = RouteOptimizer(UAV_TaskList,
GS_Cord, GS_id, UAV_id, UAVs, Task,UAV_Speed,Task_Cost,...

NewTask_id,DistUnitCost,TimeUnitCost,NewLeastPathTime,NewTaskDeadline,.
..

OldTaskDeadline,GS_CommRange,UAV_CommRange,TASK_auctionTime,
GSTaskAuctionedTimes, GS_WinningCosts)

 global GS_UAVCommStrength
 UAV_Bid = [];
 UAV_Route =[];
 PathTime = [];
if ~isempty(UAV_TaskList)
    PreviousLeastPathTime = NewLeastPathTime;
    Task_Order = perms([UAV_TaskList,NewTask_id]);
    [NumOfPath,NumOfCordPerPath] = size(Task_Order);
    UAV_Path = [repmat(UAV_id,NumOfPath,1),
Task_Order,repmat(GS_id,NumOfPath,1)];

    NumOfCordPerPath = NumOfCordPerPath + 2;
    Path_Cost = [];
    for m = 1:NumOfPath
        Path_Dist(m,:)= 0;
        UAV_Path_Cord = [UAVs(UAV_id,:); Task(Task_Order(m,:),:);
GS_Cord(GS_id,:)];
        for n = 1:NumOfCordPerPath-1
            Path_Dist(m,:) = Path_Dist(m,:) +
norm(UAV_Path_Cord(n+1,:)-UAV_Path_Cord(n,:));
        end
    end
    Path_Dist = Path_Dist -
GS_UAVCommStrength*(GS_CommRange+UAV_CommRange);
    Path_Cost = Path_Dist*DistUnitCost; % in USD
    PathTime = Path_Dist./UAV_Speed; % (in seconds)
    PathTime_Cost =  PathTime *TimeUnitCost; % in USD

   [Path_LeastCost,Path_Idx] =  min(Path_Cost + PathTime_Cost);
   UAV_Route = UAV_Path(Path_Idx,:);
   NewLeastPathTime = PathTime(Path_Idx,:);
   PathTime =  NewLeastPathTime;
   FirstTaskAuctionTime =  GSTaskAuctionedTimes(UAV_Route(2)); % the
time at which the first task in that UAV task list is auctioned
   UAV_BidCost = UAV_BidChecker(NewLeastPathTime,
Task_Cost,PreviousLeastPathTime,NewTaskDeadline,
FirstTaskAuctionTime,...
       OldTaskDeadline,DistUnitCost,UAV_Speed,TimeUnitCost);

  UAV_Bid = UAV_BidCost;
elseif isempty(UAV_TaskList)
  [UAV_Bid,PathTime] = UAV_LocalBid(GS_Cord,UAVs,GS_id,
NewTask_id,Task,UAV_id,DistUnitCost,UAV_Speed,TimeUnitCost,GS_CommRange
,UAV_CommRange,NewTaskDeadline,TASK_auctionTime,GS_WinningCosts);
  UAV_Route = [UAV_id,NewTask_id,GS_id];
end
```

File

```
function [UAV_BidCost,PathTime] = UAV_BidChecker(NewLeastPathTime,
Task_Cost,PreviousLeastPathTime,NewTaskDeadline,FirstTaskAuctionTime,Ol
dTaskDeadline,DistUnitCost,UAV_Speed,TimeUnitCost,NewTask_id,PathTime1.
..
    ,GS_Cord,GS_ID,UAV_id, UAVs,GS_CommRange,UAV_CommRange)

UAV_ExtraTime = NewLeastPathTime - PreviousLeastPathTime;
NewLeastPathDueTime = NewLeastPathTime+FirstTaskAuctionTime; % Expected
time for UAV to deliver its tasks

ExtraDistCost = UAV_ExtraTime*UAV_Speed*DistUnitCost;
ExtraTimeCost =  UAV_ExtraTime*TimeUnitCost;
UAV_CostCheck = ExtraDistCost + ExtraTimeCost;

 UAVCommCheck = norm(UAVs(UAV_id,:)- GS_Cord(GS_ID,:)); % checking to
see if UAV and GS are in communication range

if (NewTaskDeadline<=OldTaskDeadline) &&(NewLeastPathDueTime<=
NewTaskDeadline)&& (UAV_CostCheck<=Task_Cost(NewTask_id)) &&
UAVCommCheck <=(GS_CommRange + UAV_CommRange)
% if (NewLeastPathDueTime<= OldTaskDeadline)&&(NewTaskDeadline<=
OldTaskDeadline)&&(UAV_CostCheck>=Task_Cost(NewTask_id))
    UAV_BidCost = UAV_CostCheck;
    PathTime = PathTime1;
else
    UAV_BidCost = Inf;
    PathTime = Inf;
end
% [(NewTaskDeadline-OldTaskDeadline),(NewLeastPathDueTime-
NewTaskDeadline), (UAV_CostCheck-Task_Cost(NewTask_id)), (UAVCommCheck
-(GS_CommRange + UAV_CommRange))]
```

File

```
function [UAV_BidCost,PathTime] = UAV_BidChecker(NewLeastPathTime,
Task_Cost,PreviousLeastPathTime,NewTaskDeadline,FirstTaskAuctionTime,Ol
dTaskDeadline,DistUnitCost,UAV_Speed,TimeUnitCost,NewTask_id,PathTime1.
..
    ,GS_Cord,GS_ID,UAV_id, UAVs,GS_CommRange,UAV_CommRange)

UAV_ExtraTime = NewLeastPathTime - PreviousLeastPathTime;
NewLeastPathDueTime = NewLeastPathTime+FirstTaskAuctionTime; % Expected
time for UAV to deliver its tasks

ExtraDistCost = UAV_ExtraTime*UAV_Speed*DistUnitCost;
ExtraTimeCost =  UAV_ExtraTime*TimeUnitCost;
UAV_CostCheck = ExtraDistCost + ExtraTimeCost;

 UAVCommCheck = norm(UAVs(UAV_id,:)- GS_Cord(GS_ID,:)); % checking to
see if UAV and GS are in communication range
```

```
if (NewTaskDeadline<=OldTaskDeadline) &&(NewLeastPathDueTime<=
NewTaskDeadline)&& (UAV_CostCheck<=Task_Cost(NewTask_id)) &&
UAVCommCheck <=(GS_CommRange + UAV_CommRange)
% if (NewLeastPathDueTime<= OldTaskDeadline)&&(NewTaskDeadline<=
OldTaskDeadline)&&(UAV_CostCheck>=Task_Cost(NewTask_id))
    UAV_BidCost = UAV_CostCheck;
    PathTime = PathTime1;
else
    UAV_BidCost = Inf;
    PathTime = Inf;
end

% [(NewTaskDeadline-OldTaskDeadline),(NewLeastPathDueTime-
NewTaskDeadline), (UAV_CostCheck-Task_Cost(NewTask_id)), (UAVCommCheck
-(GS_CommRange + UAV_CommRange))]
```

File

```
function [UAV_Cost,PathTime] = UAV_LocalBid(TI,UAVs,TI_id,
Task_id,Task,UAV_id,DistUnitCost,UAV_Speed,...

TimeUnitCost,GS_CommRange,UAV_CommRange,NewTaskDeadline,TASK_auctionTim
es, Task_Cost,GS_Cord,GS_ID)

% This function computes the bid for UAV that communicates with a given
GS
% having a task list
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global GS_UAVCommStrength
UAV_Cost = [];
PathTime = [];
Alpha  = 0.5;
%
UAVCommCheck = norm(UAVs(UAV_id,:)- GS_Cord(GS_ID,:)); % checking to
see if UAV and GS are in communication range
if UAVCommCheck <=(GS_CommRange + UAV_CommRange)
%
    Dist = (norm(UAVs(UAV_id,:)-Task(Task_id,:)) +
norm(Task(Task_id,:)-GS_Cord(GS_ID,:)))-(1-
GS_UAVCommStrength)*(GS_CommRange+UAV_CommRange);

%     Dist = 2*norm(UAVs(UAV_id,:)-Task(Task_id,:));
    DistCost = Dist*DistUnitCost;
    Time = Dist/UAV_Speed;
    TimeCost = Time*TimeUnitCost;
    Cost = DistCost + TimeCost; % measured in USD
    if Time<=(NewTaskDeadline-TASK_auctionTimes(Task_id))&&
Cost<=Task_Cost(Task_id)
%         UAV_Cost = Alpha*Task_Cost(Task_id) + Cost*(1-Alpha); % the
bid from UAV for  a task
        PathTime = Time;
        UAV_Cost = Cost;
    else
        UAV_Cost = Inf;
        PathTime = Inf;
     end
else
    UAV_Cost =  Inf;
    PathTime =  Inf;
%      UAV_id
end
```

File

```
clear all;
close all;
clc;

global Task UAVsAll GS_Cord GS_CommRange NumberOfGS TI_CommRange TI
UAV_CommRange UAV_Speed WinnerUAV_RouteOld TangGS_CommRange
GS_UAVCommStrength...
DistUnitCost TimeUnitCost NumberOfUAVs GSTaskAuctionedTimes Up
AuctionTimeStep TI_id TaskAuctionTimesOld WinnerPathTimeOld GS_UAVList
UAVs
 global SellAndBuyBids M_BuyBid SellAndBuyWinners SellAndBuyWinnerBids
TI_TaskCost GS_jUAV_SBids InitNumberOfTask TaskCreationTime Scenario
global TI_CostScalar CycleAuctionCloseTime Gamma OriginalUAV OriginalGS
% load tasks from file
   load('RandTasks_12')
   Task = RandTasks; % Don't edit this line
%  Task = load ('-ascii', 'TaskCord.DAT'); %// old
%  %load randomized UAVs
% load('RandUAVs'); % 10 set of initial UAVs location
% SampleNumber = 1; % Choose from 1 through 10
% UAVs = RandUAVs{SampleNumber};

TangGS_CommRange = 200; % radius when GSs are tangent to each other
UAV_CommRange = 250;%750; %2500; %150; %2500; % 150, 250,750,1000,1500
UAV_CommRangeMin = 250;
Scenario = 2;   % 1 =>GSs Disjoint,  2 => GSs Overlapped, 3 => GSs at
Tangency
NumberOfUAVs = 9;
UAV_Speed = 10; % in m/s
GS_UAVCommStrength = 0.1;  % comm strength between GS and UAV. The
bigger this number the weaker the communication
GS_GS_CommStrength = 1; % comm strength between two GSs if they
intersect
% Note: (1)The bigger the communication radius, the stronger the
% communication. (2) This value lies between 0 and 1

Gamma = 0.002; % Note: Gamma is a fraction of latest delivery time of
task by eliminated UAV
% Cost constant
DistUnitCost = 0.1;
```

```
TimeUnitCost = 0.1;
Up = 1;
InitNumberOfTask = size(Task,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initiator's Coordinates
X_TI = 0;
Y_TI = 0;
Z_TI = 0;
TI = [X_TI, Y_TI, Z_TI];
TI_id = 1; % There is only one Task initiator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Generate the Ground Stations (GS)
NumberOfGS = 3;
[GS_Cord, GS_CommRange,TI_CommRange] = CreateGroundStations(NumberOfGS,
TI,TangGS_CommRange,Scenario, GS_GS_CommStrength);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp(sprintf('Actual GS communication radius for the overlap case is:
%s', num2str(GS_CommRange)))
disp(sprintf('TI communication radius: %s',
num2str(round(TI_CommRange))))
disp(sprintf('\n')) % print an empty space for clarity purposes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Generate Randomized the Unmanned Aerial Vehicles (UAVs) such that each
communicate
% with at least two GS

%  RandUAVs =
GenerateRandUAVs(NumberOfGS,NumberOfUAVs,GS_Cord,TangGS_CommRange,UAV_C
ommRangeMin);
% Note: comment the line above once you're satisfied with the initial
randomized locations of the UAVs
load(strcat(num2str(NumberOfUAVs),'RandUAVs_',num2str(UAV_CommRangeMin)
)) % don't edit this line
UAVs = RandUAVs; % Don't edit this line

%     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


UAV_TaskList{NumberOfUAVs} =  []; % initialization
GS_UAVList{NumberOfGS} =  []; % initialization
SellAndBuyBids = [];
SellAndBuyBidsLimit = [];
TaskAuctionTimes = [];
tH = [];
AuctionTimeStep = 2; %in seconds
DeadLineScalar = 10; % To give good estimation of the task's deadline
TI_CostScalar = 1; % To give good estimation of the task's cost
```

```
TaskCreationTime = 0:AuctionTimeStep:(InitNumberOfTask-
1)*AuctionTimeStep;  % Task is created every 2 seconds
UnsoldTaskIDs = 1:InitNumberOfTask;  % all tasks are unsold initially
CycleCount = 0;
SellAndBuyBids = zeros(InitNumberOfTask,2);% initialization for winning
sell and buy bids pair
GS_jUAV_SBids = zeros(InitNumberOfTask,1); % initialization
M_BuyBid = zeros(InitNumberOfTask,1); % initialization
SellAndBuyWinners = zeros(InitNumberOfTask,2); % initialization
SellAndBuyWinnerBids = zeros(InitNumberOfTask,2); % initialization
GS_TaskDelivered = zeros(InitNumberOfTask,1); % initialization
WinnerUAV_Route{NumberOfUAVs} = []; % initialization
WinnerPathTime = zeros(NumberOfUAVs,1); % initialization
OriginalUAV = zeros(InitNumberOfTask,1);
OriginalGS = zeros(InitNumberOfTask,1);
tH_Original = DeadLineScalar*(sqrt(sum((bsxfun(@minus, TI,
Task)).^2,2)))/UAV_Speed;% all original task deadlines
while(1)
CycleCount = CycleCount + 1;
NumberOfTask(CycleCount) = length(UnsoldTaskIDs);
UnsoldTaskIDsTemp = [];
CycleAuctionCloseTime(CycleCount) = sum(NumberOfTask -
1)*AuctionTimeStep;

for i=1:NumberOfTask(CycleCount)
    TaskID = UnsoldTaskIDs(i);
    if CycleCount>1
        TaskAuctionTimes(TaskID) = i*AuctionTimeStep +
CycleAuctionCloseTime(CycleCount-1);
    else
        TaskAuctionTimes(TaskID,:) =  TaskCreationTime(TaskID);
    end
    t =  TaskAuctionTimes(TaskID) - TaskCreationTime(TaskID);  % Time
ellasped before task gets sold
    tH(TaskID,:) = tH_Original(TaskID) + t; % Task deadline

    %Compute the cost of the task to be auctioned
    TI_Po(:,TaskID) = 2*(norm(TI-Task(TaskID,:))*DistUnitCost +
norm(TI-Task(TaskID,:))*TimeUnitCost/UAV_Speed); % initial price from
the task initiator
    TI_TaskCost(TaskID,:) = TI_CostScalar*TI_Po(:,TaskID)*(1 +
Up*t/tH_Original(TaskID));
    GS_Bids{TaskID} = GS_Bidding(NumberOfGS, TI,
Task(TaskID,:),GS_Cord,TI_CommRange,GS_CommRange,UAV_Speed,DistUnitCost
,TimeUnitCost,TI_TaskCost,TaskID); % sell bids


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% Compute the bids for the UAVs
    UAVwinnerBids  = zeros(1,NumberOfGS);
    WinnerUAVnames = zeros(1,NumberOfGS);
    GSs_UAVs_BidPairs = zeros(NumberOfGS,4);
    GSnames = 1:NumberOfGS;
    for gs = 1:NumberOfGS
        UAV_BidsTemp = Inf*ones(1,NumberOfUAVs); % initialization
        for k=1:NumberOfUAVs
            if isempty(UAV_TaskList{k})
                UAV_TaskListPrevious = [];
                NewLeastPathTime = 0;
                OldTaskDeadline = 0;
                oldGS_ID = 0;

[UAV_Bids{TaskID}(k),GS(gs).UAV_Route{k},GS(gs).PathTime(k)] =
UAV_Bidding(UAV_TaskListPrevious, TI, TI_id, k, UAVs,
Task,UAV_Speed,TI_TaskCost,...

TaskID,DistUnitCost,TimeUnitCost,NewLeastPathTime,tH(TaskID),...
                    OldTaskDeadline,GS_CommRange,UAV_CommRange,
TaskAuctionTimes,GS_Cord,gs,oldGS_ID);
            else
                UAV_TaskListPrevious = UAV_TaskList{k};
                NewLeastPathTime = WinnerPathTime(k,:);
                OldTaskDeadline = min(tH(UAV_TaskList{k}));
%tH(GS(gs).UAV_Route{k}(end-1));
                oldGS_ID = WinnerUAV_Route{k}(end);

[UAV_Bids{TaskID}(k),GS(gs).UAV_Route{k},GS(gs).PathTime(k)] =
UAV_Bidding(UAV_TaskListPrevious, TI, TI_id, k, UAVs,
Task,UAV_Speed,TI_TaskCost,...

TaskID,DistUnitCost,TimeUnitCost,NewLeastPathTime,tH(TaskID),...
                    OldTaskDeadline,GS_CommRange,UAV_CommRange,
TaskAuctionTimes,GS_Cord,gs,oldGS_ID);
            end
        end
        %          UAV_Bids{TaskID}
        [UAVwinnerBids(gs), WinnerUAVnames(gs)] =
min(UAV_Bids{TaskID});
        UAV_BidsTemp(WinnerUAVnames(gs)) = UAVwinnerBids(gs);
    end

    GSs_UAVs_BidPairs = [GSnames',round(GS_Bids{TaskID}'),
WinnerUAVnames',round(UAVwinnerBids')];
    disp('GSs and UAVs bid pairs:')
    disp('          GS        GS_Bid       UAV       UAV_Bid')
    disp(GSs_UAVs_BidPairs)
    %       UAV_Bids{i}


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Matching the GS_Bids (i.e. sell bids) and the UAV_Bids (i.e. buy
    % bids
    Sell_in_Bids = [];
    Buy_in_Bids  = [];
    TransactionSet = [];
    UAV_Indx = [];
    GS_Indx = [];
    UAV_Indx1 = [];
    GS_Indx1 = [];

    % Determine eligible sellers and buyers
    for m=1:NumberOfGS
        if GS_Bids{TaskID}(m)~=Inf && UAVwinnerBids(m)~=Inf
                    TransactionSet = [TransactionSet;
[GS_Bids{TaskID}(m),UAVwinnerBids(m)]];
                    UAV_Indx1 = [UAV_Indx1; WinnerUAVnames(m)];
                    GS_Indx1 = [GS_Indx1; m];

        end
    end

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    NumOfSellers = size(TransactionSet,1);
    if ~isempty(TransactionSet) % There are seller(s) and buyer(s)

        [ignore1 OrigIndx] = min(TransactionSet(:,1));
        OriginalUAVtemp = UAV_Indx1(OrigIndx);
        OriginalGStemp  = GS_Indx1(OrigIndx);

        Rank = sort([TransactionSet(:,1),
TransactionSet(:,2)],'ascend');
        M_BuyBid(TaskID,:) = Rank(NumOfSellers);   % Mth bid
        M_nextBuyBid(TaskID,:) = Rank(NumOfSellers+1); % M+1st bid

        [Sell_Bids{TaskID}, GS_IndxTemp{TaskID}] =
min(TransactionSet(:,1));
        GS_Indx(1)   = GS_Indx1(GS_IndxTemp{TaskID}); % winner GS(i.e.
GS with minimum bid)
        UAV_Indx(1)  = UAV_Indx1(GS_IndxTemp{TaskID});  % winner UAV
(i.e. UAV with minimum bid)

        Buy_Bids{TaskID} = TransactionSet(GS_IndxTemp{TaskID},2);
%         GS_Winner = GS_Indx1(GS_IndxTemp{TaskID}); % actual winner
        GS_jUAV_SBids(TaskID) = GS_Bids{TaskID}(GS_Indx(1)); % cost of
GS to which task is supposed to be delivered
        GS_TaskDelivered(TaskID) = GS_Indx(1);

        %%%Determine the winner
        UAV_TaskList{UAV_Indx(1)}    = [UAV_TaskList{UAV_Indx(1)},
TaskID];
        GS_UAVList{GS_Indx(1)}       = [GS_UAVList{GS_Indx(1)},
UAV_Indx(1)];
        WinnerUAV_Route{UAV_Indx(1)} =
GS(GS_Indx(1)).UAV_Route{UAV_Indx(1)};
```

```
        WinnerPathTime(UAV_Indx(1))  =
GS(GS_Indx(1)).PathTime(UAV_Indx(1));
        SellAndBuyBids(TaskID,:)      = SellAndBuyWinnerBids(TaskID,:);
%winning sell and buy bids pair
        SellAndBuyWinners(TaskID,:) =  [GS_Indx(1), UAV_Indx(1)]; %
UAVs and GSs that win the tasks
        SellAndBuyWinnerBids(TaskID,:) = [Sell_Bids{TaskID},
Buy_Bids{TaskID}];
        SellAndBuyBids(TaskID,:) =     SellAndBuyWinnerBids(TaskID,:);

        OriginalUAV(TaskID) = OriginalUAVtemp; % UAV with minimum cost
        OriginalGS(TaskID)  = OriginalGStemp;  % GS paired with UAV
with minimu cost
    else % there are no sellers and/or buyers
        UnsoldTaskIDsTemp = [UnsoldTaskIDsTemp,TaskID];
    end
end
TransactionSet = [];

% Check if all tasks are sold
if isempty(UnsoldTaskIDsTemp)
            break; % break from the while loop
else
    UnsoldTaskIDs = UnsoldTaskIDsTemp; % tasks unsolds
end
%     UnsoldTaskIDs
end

 UAV_TaskList % without robustness
TaskAuctionTimesOld = TaskAuctionTimes;
WinnerPathTimeOld = WinnerPathTime;
WinnerUAV_RouteOld = WinnerUAV_Route;
TaskNames = 1:InitNumberOfTask;
% Calculate the profit using the clearing price (i.e. M_BuyBid)
SellerProfit1  = 0.5*(SellAndBuyWinnerBids(:,1)-
SellAndBuyWinnerBids(:,2) + TI_TaskCost - SellAndBuyWinnerBids(:,1) -(
GS_jUAV_SBids-SellAndBuyWinnerBids(:,1))); % profit of GS to which task
is delivered to
BuyerProfit1   = SellerProfit1;  %profit of the winner UAV
OriginalGS_Profit      = 0.5*(GS_jUAV_SBids -
SellAndBuyWinnerBids(:,1)); % profit of original GS that was paired
with the winner UAV

AvgSellBid = mean(SellAndBuyWinnerBids(:,1));
AvgBuyBid = mean(SellAndBuyWinnerBids(:,2));
% AvgGS_DelivererBid = mean(GS_jUAV_SBids);
AvgOriginalGS_Bid = mean(GS_jUAV_SBids);
AllAvgBids = [AvgOriginalGS_Bid,AvgSellBid,AvgBuyBid];

AvgClearingPrice = mean(M_BuyBid);
AvgOriginalGS_Profit = mean(OriginalGS_Profit);
AvgSellProfit = mean(SellerProfit1);
AvgBuyProfit = mean(BuyerProfit1);
```

```
% AvgGS_DelivererProfit = mean(GS_Profit);
AllAvgProfits = [AvgClearingPrice,AvgOriginalGS_Profit,AvgSellProfit,
AvgBuyProfit];


% Compute the total profits for all GS

GS_TotalProfits = zeros(NumberOfGS,1);
GS_Names = 1:NumberOfGS;
for g = 1:NumberOfGS
    GS_Originalfrequency = find(OriginalGS==g); % how often GS
originally paired with the winner UAV
    GS_Sellfrequency = find(SellAndBuyWinners(:,1)==g); % how often GS
got tasks delivered to it

    if ~isempty(GS_Originalfrequency)
    GS_TotalProfits(g) = GS_TotalProfits(g)+
sum(OriginalGS_Profit(GS_Originalfrequency));
    end
        if ~isempty(GS_Sellfrequency)
    GS_TotalProfits(g) = GS_TotalProfits(g)+
sum(SellerProfit1(GS_Sellfrequency));
        end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% Compute the total for all UAVs

UAV_TotalProfits = zeros(NumberOfUAVs,1);
UAV_Names = 1:NumberOfUAVs;
for u = 1:NumberOfUAVs
    UAV_Buyfrequency = find(SellAndBuyWinners(:,2)==u); % how often UAV
buys tasks from Winner GS
    if ~isempty(UAV_Buyfrequency)
    UAV_TotalProfits(u) = UAV_TotalProfits(u)+
sum(BuyerProfit1(UAV_Buyfrequency));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Write the results into excel sheeet
TITLE = {'TaskName','Creation Time(sec)','Auction
Time(sec)','Deadline(sec)','Task Cost($) (from TI)', 'Original Seller
Bid($)','Winning Seller Bid($)', 'Winning Buyer Bid($)',...
    'Original_GS', 'Seller(GS)','Buyer(UAV)', 'Clearing Price($)',
'Original_GS Profit($)', 'Seller Profit($)','Buyer Profit($)'};
```

```
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)), TITLE,
'A1:O1')
BuySellRecords =
[TaskNames',TaskCreationTime',TaskAuctionTimes,tH,TI_TaskCost
,GS_jUAV_SBids, SellAndBuyWinnerBids(:,1),SellAndBuyWinnerBids(:,2),...
OriginalGS, SellAndBuyWinners(:,1), SellAndBuyWinners(:,2),M_BuyBid,
OriginalGS_Profit, SellerProfit1,BuyerProfit1];
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
BuySellRecords, strcat('A2:O',num2str(InitNumberOfTask+1)))
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
{'AVERAGE'},
strcat('E',num2str(InitNumberOfTask+2),':','E',num2str(InitNumberOfTask
+2)))
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
AllAvgBids,
strcat('F',num2str(InitNumberOfTask+2),':','H',num2str(InitNumberOfTask
+2)))
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
AllAvgProfits,
strcat('L',num2str(InitNumberOfTask+2),':','O',num2str(InitNumberOfTask
+2)))

SubTitle = {'GS', 'GS Total Profit($)'};
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)), SubTitle,
strcat('I',num2str(InitNumberOfTask+5),':','J',num2str(InitNumberOfTask
+5)))
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
GS_Names',
strcat('I',num2str(InitNumberOfTask+6),':','I',num2str(InitNumberOfTask
+6+NumberOfGS-1)))
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
GS_TotalProfits,
strcat('J',num2str(InitNumberOfTask+6),':','J',num2str(InitNumberOfTask
+6+NumberOfGS-1)))


SubTitle = {'UAV', 'UAV Total Profit($)'};
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)), SubTitle,
strcat('M',num2str(InitNumberOfTask+5),':','N',num2str(InitNumberOfTask
+5)))
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
UAV_Names',
strcat('M',num2str(InitNumberOfTask+6),':','M',num2str(InitNumberOfTask
+6+NumberOfUAVs-1)))
xlswrite(strcat('BuySellResults_Scenario',num2str(Scenario)),
UAV_TotalProfits,
strcat('N',num2str(InitNumberOfTask+6),':','N',num2str(InitNumberOfTask
+6+NumberOfUAVs-1)))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('Ground Stations coordinates (in rows):')
disp(round(GS_Cord))
disp('UAVs Initial Coordinates (in rows):')
disp(round(UAVs))
disp('Tasks coordinates (in rows):')
disp(Task)

%Display the UAV tasklists
TimeHistory = zeros(NumberOfUAVs,4); %initialization
GS_UAVRouteCordAll{NumberOfUAVs} = []; % initialization
UAVTasksCompletionDurations  = zeros(NumberOfUAVs,1);
GS_UAVRoutesAll{NumberOfUAVs} = [];
TaskCompletionTimes = zeros(1,NumberOfUAVs);
TaskCompletionDists = [];
for UAVname=1:NumberOfUAVs
    if ~isempty(UAV_TaskList{UAVname})
        disp(strcat('RECORDS FOR UAV',num2str(UAVname),':'))
        disp('Tasklist:'), disp(UAV_TaskList{UAVname})
        disp('Minimum Route:'), disp(WinnerUAV_Route{UAVname})
        disp('StartTime  DeliveryTime  PathTime LatestDeliveryTime')
        DeliveryTime = round(CycleAuctionCloseTime(end))+
round(WinnerPathTime(UAVname));
        LatestDeliveryTime = min(tH(UAV_TaskList{UAVname}));
        TimeHistory(UAVname,:) =
round([CycleAuctionCloseTime(end),DeliveryTime,
WinnerPathTime(UAVname), LatestDeliveryTime]);
        disp(TimeHistory(UAVname,:))
        GS_UAVRouteCordAll{UAVname} =
[UAVs(UAVname,:);Task(WinnerUAV_Route{UAVname}(2:end-1),:);
GS_Cord(WinnerUAV_Route{UAVname}(end),:)];
        TaskNames = WinnerUAV_Route{UAVname}(2:end-1);
        UAVTasksCompletionDurations(UAVname) = WinnerPathTime(UAVname);
        GS_UAVRoutesAll{UAVname} = WinnerUAV_Route{UAVname};
        TaskCompletionTimes(UAVname) = WinnerPathTime(UAVname);
        figure('Name', strcat('Path for executing task by UAV',
num2str(UAVname)))

plot3(GS_UAVRouteCordAll{UAVname}(:,1),GS_UAVRouteCordAll{UAVname}(:,2)
,GS_UAVRouteCordAll{UAVname}(:,3),'*-')

text(GS_UAVRouteCordAll{UAVname}(1,1),GS_UAVRouteCordAll{UAVname}(1,2),
GS_UAVRouteCordAll{UAVname}(1,3), strcat('UAV',num2str(UAVname)))
        text(GS_UAVRouteCordAll{UAVname}(2:end-
1,1),GS_UAVRouteCordAll{UAVname}(2:end-
1,2),GS_UAVRouteCordAll{UAVname}(2:end-1,3), num2str(TaskNames'))

text(GS_UAVRouteCordAll{UAVname}(end,1),GS_UAVRouteCordAll{UAVname}(end
,2),GS_UAVRouteCordAll{UAVname}(end,3),
strcat('GS',num2str(WinnerUAV_Route{UAVname}(end))))
        grid on
        xlabel('x')
        ylabel('y')
        zlabel('z')
    end
end
```

```
% WinnerPathTime(4)
TaskCompletionDists = UAV_Speed.*TaskCompletionTimes;
% Display the GS and the UAVs  it solds tasks to
for GS=1:NumberOfGS
    if ~isempty(GS_UAVList{GS})
        disp(strcat('UAV LIST FOR GS',num2str(GS),':'))
        disp(GS_UAVList{GS})
    end
end

% TaskCompletionTimesNew = [TaskCompletionTimes,zeros(1, NumberOfUAVs-
length(TaskCompletionTimes))];
% TaskCompletionDists = UAV_Speed.*TaskCompletionTimesNew;

disp('    AvgDist(m)   AvgTime(s)  Dist_std(m)  Time_std(s)')
disp(round([mean(TaskCompletionDists), mean(TaskCompletionTimes),
std(TaskCompletionDists), std(TaskCompletionTimes)]))

% plot the space
PlotTheSpaceWithTheAgents(GS_Cord, GS_CommRange,
NumberOfGS,TI_CommRange, TI,UAVs,NumberOfUAVs)

%  % Running the robustness
 % Inputs required from the user:
 disp(sprintf('\n'))
 disp(sprintf('\n'))
 disp('RESULTS FROM THE ROBUSTNESS')
 UAV_ID_Eliminate = 1;
 UAV_Elimination_Time = 50;
 UAVRecoveryTime = 4400;
 Robustness(UAV_ID_Eliminate, UAV_Elimination_Time, UAVRecoveryTime,
GS_UAVRoutesAll,...
 TimeHistory, GS_UAVRouteCordAll,TaskCompletionTimes,tH)
```

File

```
function [UAV_GSRoutesNeg,UAV_GSRoutesCordNeg]=
UAVNegotiator(GS_UAVRouteCordAll,GS_UAVRoutesAll,UAV_CommRange,...

UAV_Speed,TimeHistory,Deadlines,NegoTimeStep,DurationForNegotiation,Dis
tUnitCost,TimeUnitCost)

global TaskName
%%%%%%%% Computes the segmental lengths of the UAVs routes
[RouteLenghts,UAVCumRouteLenghts] =
ComputeSegmentDistance(GS_UAVRouteCordAll,GS_UAVRoutesAll);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NumberOfRoutes = length(GS_UAVRoutesAll);

for k =1:NegoTimeStep:DurationForNegotiation % first 3 seconds
```

```
        for i=1:NumberOfRoutes
            RoutesIDs = 1:NumberOfRoutes;
            RoutesIDs(find(RoutesIDs==i))=[]; % avoid self-negotiation
            UAVNegPath{i} = [];
            PathCostNeg(i) = inf;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            DistTravel_UAVi = CalUAVDistance(UAV_Speed,
TimeHistory(i,1),k);

            [UAVNewLocation(i,:),UAVi_MovingTo] =
FindUAVnewLocation(UAVCumRouteLenghts{i}, GS_UAVRouteCordAll{i},
RouteLenghts{i},DistTravel_UAVi);

            UAVNegPath = []; PathCostNeg = [];
            for kk=1:length(RoutesIDs)
                j = RoutesIDs(kk);
                UAVNegPath{j} = [];

            DistTravel_UAVj = CalUAVDistance(UAV_Speed,
TimeHistory(kk,1),k);
            [UAVNewLocation(j,:),UAVj_MovingTo, LastTaskDone] =
FindUAVnewLocation(UAVCumRouteLenghts{j}, GS_UAVRouteCordAll{j},
RouteLenghts{j},DistTravel_UAVj);
        NegoCommCheck = norm(UAVNewLocation(i,:)-UAVNewLocation(j,:));
% UAVs new locations used
        if (NegoCommCheck <= (UAV_CommRange
+UAV_CommRange))&&(length(RouteLenghts{i})>2&&
length(RouteLenghts{j})>2) % UAVs in comm range

%              disp('Communicate')
            [UAVNegPath{j},PathCostNeg(j)] =
CalUAVsNegBids(DistTravel_UAVi,UAVi_MovingTo,...

UAVj_MovingTo,GS_UAVRouteCordAll{i},GS_UAVRouteCordAll{j},UAVNewLocatio
n(i,:),...

UAVNewLocation(j,:),DistUnitCost,TimeUnitCost,UAV_Speed,TimeHistory,j,D
eadlines,GS_UAVRoutesAll{i});
        else
            UAVNegPath{j} = [];
            PathCostNeg(j) = inf;
        end
            end
            %%%% Determine the winner from the negotiation
            [ignore, NegWinner]= min(PathCostNeg);
            if PathCostNeg~=inf
                GS_UAVRouteCordAll{NegWinner} = UAVNegPath{NegWinner};
% winner route updated by adding a task coordinate
```

```
                GS_UAVRoutesAll{NegWinner}=
[GS_UAVRoutesAll{NegWinner}(1:UAVj_MovingTo-
1),TaskName,GS_UAVRoutesAll{NegWinner}(UAVj_MovingTo:end)];

                GS_UAVRouteCordAll{i}(UAVi_MovingTo,:) = []; % delete
the task coordinate from tasklist
                GS_UAVRoutesAll{i}(UAVi_MovingTo) = []; % delete the
task id from tasklist

                % Uspdate the segmental lengths
                [RouteLenghts{NegWinner},CumRouteLenghts{NegWinner}] =
ComputeSegmentDistance(GS_UAVRouteCordAll{NegWinner},GS_UAVRoutesAll{Ne
gWinner});
                [RouteLenghts{i},CumRouteLenghts{i}] =
ComputeSegmentDistance(GS_UAVRouteCordAll{i},GS_UAVRoutesAll{i});

            end
        end
end

   UAV_GSRoutesNeg =    GS_UAVRoutesAll;
   UAV_GSRoutesCordNeg  = GS_UAVRouteCordAll;
```

File

```
function UAVsRoutePlot(UAVRouteCord,GS_ID,UAVs_Route)
global UAVsNames
% This function plots the routes for the UAVs
figure('Name',strcat('Ground Station',num2str(GS_ID)))
COLORS = ['r';'b';'g';'y';'k';'m';'c'];

for i = 1: length(UAVRouteCord)
    UAVRouteCord{i} =real(UAVRouteCord{i});
    if i<length(COLORS)+1

plot3(UAVRouteCord{i}(:,1),UAVRouteCord{i}(:,2),UAVRouteCord{i}(:,3),st
rcat(COLORS(i),'o-'))

text(UAVRouteCord{i}(1,1),UAVRouteCord{i}(1,2),UAVRouteCord{i}(1,3),str
cat('UAV',num2str(UAVsNames(UAVs_Route{i}(1,1)))))

text(UAVRouteCord{i}(end,1),UAVRouteCord{i}(end,2),UAVRouteCord{i}(end,
3),'GS')
    hold on
    else

plot3(UAVRouteCord{i}(:,1),UAVRouteCord{i}(:,2),UAVRouteCord{i}(:,3),'h
-')
```

```
text(UAVRouteCord{i}(1,1),UAVRouteCord{i}(1,2),UAVRouteCord{i}(1,3),
strcat('UAV',num2str(UAVsNames(UAVs_Route{i}(1,1)))))

text(UAVRouteCord{i}(end,1),UAVRouteCord{i}(end,2),UAVRouteCord{i}(end,
3),'GS')
    hold on
    end
end
grid on
hold off
```