Robust and Decentralized Task Assignment Algorithms for UAVs

by

Mehdi Alighanbari

M.Sc., Operations Research, MIT Sloan School of Management, 2004
M.Sc., Aeronautics and Astronautics, MIT, 2004
M.Sc., Electrical Engineering, NC A&T State University, 2001
B.Sc., Electrical Engineering, Sharif University of Technology, 1999

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007

Author .... [illegible]
Department of Aeronautics and Astronautics
July 31, 2007

Accepted by .... [illegible]
David L. Darmofal
Associate Professor of Aeronautics and Astronautics
Chair, Department Committee on Graduate Students

Robust and Decentralized Task Assignment Algorithms for UAVs

by

Mehdi Alighanbari

Accepted by .... [illegible]
Jonathan P. How
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .... [illegible]
Nicholas Roy
Assistant Professor of Aeronautics and Astronautics

Accepted by .... [illegible]
Emilio Frazzoli
Assistant Professor of Aeronautics and Astronautics

Robust and Decentralized Task Assignment Algorithms for UAVs

by

Mehdi Alighanbari

Submitted to the Department of Aeronautics and Astronautics
on July 31, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis investigates the problem of decentralized task assignment for a fleet of UAVs. The main objectives of this work are to improve the robustness to noise and uncertainties in the environment and improve the scalability of standard centralized planning systems, which are typically not practical for large teams. The main contributions of the thesis are in three areas related to distributed planning: information consensus, decentralized conflict-free assignment, and robust assignment.

Information sharing is a vital part of many decentralized planning algorithms. A previously proposed decentralized consensus algorithm uses the well-known Kalman filtering approach to develop the Kalman Consensus Algorithm (KCA), which incorporates the certainty of each agent about its information in the update procedure. It is shown in this thesis that although this algorithm converges for general form of network structures, the desired consensus value is only achieved for very special networks. We then present an extension of the KCA and show, with numerical examples and analytical proofs, that this new algorithm converges to the desired consensus value for very general communication networks.

Two decentralized task assignment algorithms are presented that can be used to achieve a good performance for a wide range of communication networks. These include the Robust Decentralized Task Assignment (RDTA) algorithm, which is shown to be robust to inconsistency of information across the team and ensures that the resulting decentralized plan is conflict-free. A new auction-based task assignment algorithm is also developed to perform assignment in a completely decentralized manner where each UAV is only allowed to communicate with its neighboring UAVs, and there is no relaying of information. In this algorithm, only necessary information is communicated, which makes this method communication-efficient and well-suited for low bandwidth communication networks.

The thesis also presents a technique that improves the robustness of the UAV task assignment algorithm to sensor noise and uncertainty about the environment. Previous work has demonstrated that an extended version of a simple robustness algorithm in the literature is as effective as more complex techniques, but significantly

easier to implement, and thus is well suited for real-time implementation. We have also developed a Filter-Embedded Task assignment (FETA) algorithm for accounting for changes in situational awareness during replanning. Our approach to mitigate "churning" is unique in that the coefficient weights that penalize changes in the assignment are tuned online based on previous plan changes. This enables the planner to explicitly show filtering properties and to reject noise with desired frequencies.

This thesis synergistically combines the robust and adaptive approaches to develop a fully integrated solution to the UAV task planning problem. The resulting algorithm, called the Robust Filter Embedded Task Assignment (RFETA), is shown to hedge against the uncertainty in the optimization data and to mitigate the effect of churning while replanning with new information. The algorithm demonstrates the desired robustness and filtering behavior, which yields superior performance to using robustness or FETA alone, and is well suited for real-time implementation.

The algorithms and theorems developed in this thesis address important aspects of the UAV task assignment problem. The proposed algorithms demonstrate improved performance and robustness when compared with benchmarks and they take us much closer to the point where they are ready to be transitioned to real missions.

Acknowledgments

In carrying out the research that went into this PhD dissertation, there were several key individuals that played large roles in helping me make it to the end. This was a long and difficult road at times and I thank everyone whole-heartedly for their kindness and support.

Firstly I would like to thank my advisor, Professor Jonathan How, for directing and guiding me through this research. I also thank my committee members, Professor Nicholas Roy and Professor Emilio Frazzoli, for their input and oversights. Next, I would like to thank my research colleagues at the Aerospace Controls Laboratory, among them, Yoshiaki Kuwata, Louis Breger, and Luca Bertuccelli. Thanks also to Professor How's administrative assistant, Kathryn Fischer, for her support throughout this time.

A special warm thank you to all my friends in Boston for their support and assistance.

In appreciation for a lifetime of support and encouragement, I thank my parents, Javaad and Shayesteh Alighanbari, my sisters, Jila, Jaleh and Laleh and my uncle Reza Tajalli.

This research was funded under AFSOR contract # F49620-01-1-0453.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis investigates the problem of Robust and Decentralized Task Assignment for Unmanned Aerial Vehicles (UAV). In particular, it addresses the limitations of the centralized planning algorithms and develops new decentralized algorithms to eliminate these limitations.

The introduction will continue with the motivation of the work in Section 1.1, which defines the problems of interest and presents previous works in these areas along with the challenges faced. Section 1.2 provides a brief background on the tools that are used in this thesis, and finally, Section 1.3 presents the outline of the thesis and the summary of contributions of each chapter.

1.1 Motivation

UAV planning and control have recently been given much attention from different research communities due to their extensive predicted role in the future of air combat missions [4, 5, 10, 21, 23, 24, 25, 33, 34, 41, 50, 52, 72]. With the current degree of autonomy, today's UAVs typically require several operators, but future UAVs will be designed to autonomously make decisions at every level of planning and will be integrated into teams that cooperate to achieve any mission goals, thereby allowing one operator to control a fleet of many UAVs.

Achieving autonomy for UAVs is a complex problem and the degree of complexity is different for different levels of decision making and is directly related to the degree of cooperation required between UAVs. For instance, the low level control (i.e. waypoint follower) requires almost no cooperation between UAVs and can be easily automated. The highest level of cooperation is required in the task assignment level, where UAVs need to share information, divide tasks and assign tasks to UAVs with the appropriate task timing and ordering. This level of cooperation and information sharing makes the autonomous task assignment problem very complex.

1.1.1 Decentralized Task Assignment

Cooperative Task Assignment for UAVs has been the topic of much research; many different algorithms have been proposed to solve the task assignment problem [4, 5, 10, 21, 23, 24, 25, 33, 34, 41, 50, 52, 72], but most of these are centralized algorithms and require a central planing agent [10, 34, 52, 72]. This agent can be a ground station that receives all the information from all the UAVs, calculates the optimal plan, and sends each UAV its plan. It can also be one of the UAVs in the team that acts as the central planner. In this setup, the planner UAV is also called the leader. There are also variations of this method, such as sub-team allocations, in which the fleet is divided into smaller teams and each team has a leader [10] or emerging (dynamic) leader in which each UAV can become a leader under certain circumstances [30].

Although the leader approach eliminates the need for a ground planner from the task assignment algorithm, most of the issues associated with the centralized planning (i.e. lack of autonomy, scalability, high level of communication, and robustness) still exist. The desired level of autonomy in which each UAV can contribute to the overall mission objective independently can only be accomplished when each UAV creates its own plan while cooperating with other UAVs by means of communication. This can only be achieved with a decentralized task assignment scheme. There are several important questions that need to be answered in the decentralized task assignment. An important part of the decentralized task assignment is the communication. What should be communicated, when should it be communicated and to whom? What is the optimal communication scheme, given the limitations and objectives? What is the trade off between communication effort and performance? How much communication bandwidth is enough? These are the core questions that need to be answered. There are also important algorithmic questions that need to be answered. What is a good distributed algorithm for UAV task assignment? Will the algorithm work in different environments with different communication structures? How much communication is needed? Is it robust to changes in the network structure? Does it always create a feasible plan?

This thesis investigates the decentralized task assignment problem and addresses these questions by introducing new approaches and analyzing many different aspects, including:

- Information sharing is an essential and important part of the UAV task assignment problem. This problem is straightforward in the centralized schemes in which every UAV communicates with the central planner. However, in any decentralized method, information sharing becomes an important and very complex problem. Most of the decentralized algorithms rely on the assumption of consistent information among the fleet, and therefore convergence of the information sharing algorithms becomes very important. In this thesis, a Kalman filtering based consensus algorithm will be addressed and analytical and simulation results will be presented to prove the convergence of the proposed algorithms.

- The second part of this thesis will deal with the decentralized task assignment algorithms. Different existing decentralized algorithms will be analyzed and their advantages and disadvantages will be discussed. We further introduce two new decentralized task assignment algorithms that address the issues associated with the existing methods.

1.1.2 Robust Planning

UAVs in the near future will require an ever increasing number of higher-level planning capabilities in order to successfully execute their missions. These missions will be complex, requiring multiple heterogeneous vehicles to successfully cooperate in order to achieve the global mission objective. The vehicles will also have to rely on their sensor information to successfully classify true targets, reject false targets, and make a coherent series of decisions to achieve their objectives. Unfortunately, the vehicles' situational awareness will typically be impacted by the imperfections in the sensors and/or adversarial strategies, all of which may lead the vehicles to falsely conclude that a target is present in the environment, or that the target has a higher value than it actually does. The vehicles will nonetheless have to use the information at their disposal to autonomously come up with the actions, whether through a centralized planner, or in a decentralized fashion.

An important component of these planning capabilities will be the ability to incorporate uncertainty in the mission plans, and conduct missions that are robust to this uncertainty. The concept of robustness is an issue that mainly addresses the performance objective, and notionally the goal of robust optimization is to maximize the worst-case realization of the planner's objective function. At the same time, the vehicles will also need to update their information on the environment, and respond to true changes in the battlespace, while correctly rejecting adversarial false information. Failure to do so may result in a phenomenon called churning, whereby the vehicles constantly replan based on their latest noisy information, and oscillate between targets without ever reaching any of the targets. Planner robustness to decisions, and planner adaptiveness to the environment are thus two key components that must be included in higher-level planners.

1.2 Background

The following sections briefly define the UAV task assignment problem and consensus problem, which are the problems of interest throughout this thesis.

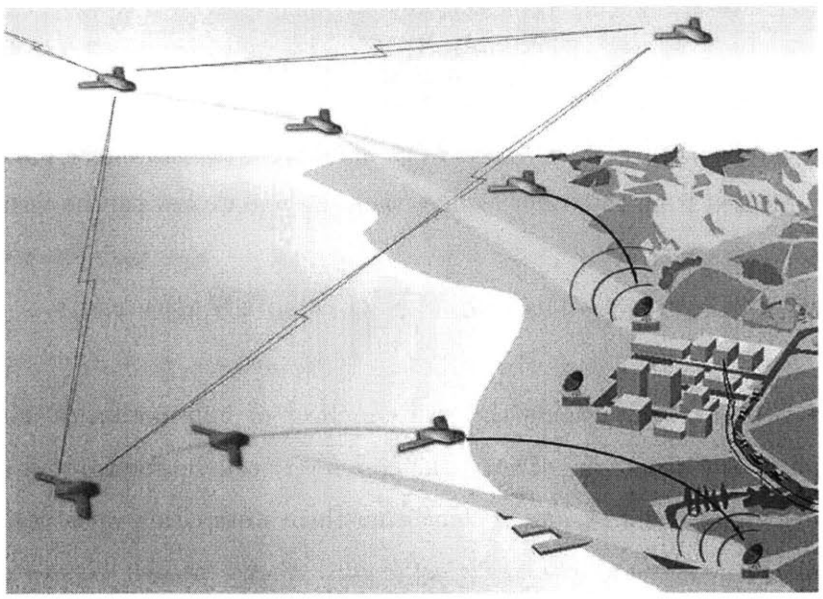


Figure 1.1: Typical UAV Mission.

1.2.1 UAV Task Assignment Problem

Figure 1.1 shows a simple UAV mission. In developing task assignment algorithms, several assumptions are made. The set of tasks and waypoints associated with them have been identified. Each team is made up of several UAVs with known starting points, speed, and capability (*i.e.,* strike, reconnaissance, etc.). It is also assumed that there are "No Fly Zones" in the environment.

Given this information, the problem is to assign the UAVs to the tasks to optimally fulfill a specified objective. The objective that is used throughout this research is maximizing the expected time-discounted value of the mission.

1.2.2 Consensus Problem

Suppose there are n agents (i.e. UAVs) $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ with inconsistent information and let x_i be the information associated with agent i. The objective is for the agents to communicate this information amongst themselves to reach consensus, which means that all of the agents have the same information ($x_i = x_j,\ \forall i, j \in \{1, \ldots, n\}$).

The communication pattern at any time t can be described in terms of a directed graph $\mathbb{G}(t) = (\mathcal{A}, \mathcal{E}(t))$, where $(\mathcal{A}_i, \mathcal{A}_j) \in \mathcal{E}(t)$ if and only if there is a unidirectional information exchange link from $\mathcal{A}_i$ to $\mathcal{A}_j$ at time t.

If the information, x_i of agent $\mathcal{A}_i$, is updated in discrete time steps using the data communicated from the other agents, then the update law can be written as

$$x_i(t+1) = x_i(t) + \sum_{j=1}^{N} \alpha_{ij}(t) g_{ij}(t)(x_j(t) - x_i(t)) \tag{1.1}$$

where $\alpha_{ij}(t) \geq 0$ represents the relative effect of information of agent $\mathcal{A}_j$ on the information of agent $\mathcal{A}_i$. The parameter $\alpha_{ij}(t)$ can be interpreted as the relative confidence that agent $\mathcal{A}_i$ and $\mathcal{A}_j$ have that their information variables are correct [8]. Several methods such as *Fixed Coefficients, Vicsek Model, Gossip Algorithm* and *Kalman Filtering* have been proposed to pick values for $\alpha_{ij}(t)$, [20, 68, 6].

1.3 Outline and Summary of Contribution

The goal of this research is to address two important issues of UAV task assignment by developing new decentralized methods for the UAV task assignment problem and making the assignment algorithms robust to the uncertainty and noise in the environment. It involves work in theory, algorithm design, and simulations. The thesis consists of four main chapters. In the following sections, the contributions of each chapter are presented.

1.3.1 Kalman Consensus

The Kalman filtering idea can be used to design consensus algorithms. In the Kalman filtering approach, the coefficients, α_{ij} in (1.1), are chosen to account for the uncertainty each agent has in its information.

Ren *et al.* developed the Kalman filtering algorithm for the continuous and discrete consensus problem and presented simulations and analytical proofs to show that they converge [68]. However, these simulations make a strong assumption about the topology of the communication network in which the inflow and outflow of all the agents are equal. Although the algorithm converges for the general form of a strongly connected communication network, its convergence to the true estimate (the estimate if it was calculated using a centralized Kalman filter) is only guaranteed for this special type of network.

This thesis proposes a modification to the decentralized Kalman consensus algorithm to create unbiased estimates for the general case of communication networks. The thesis provides simple examples that highlight the deficiencies of previous approaches, and simulation results that show this method eliminates biases associated with the previous Kalman consensus algorithm. Theorems are presented to prove the convergence of the new algorithm to the centralized estimate for many different classes of communication networks.

Previous literature shows that to achieve the weighted average consensus with the existing consensus algorithms, communication networks have to be balanced [59, 62]. The algorithm developed in this chapter eliminates this limiting requirement of a balanced network and shows that the desired weighted average can be achieved for the very general form of dynamic communication networks.

1.3.2 Robust Decentralized Task Assignment

One commonly proposed decentralized approach to planning is to replicate the central assignment algorithm on each UAV [4, 25]. The success of this so called *implicit coordination* algorithm strongly depends on the assumption that all UAVs have the same information (situational awareness). However, this assumption is very limiting and usually cannot be satisfied due to uncertain, noisy, and dynamic environments. Chapter 3 presents simulations to show that reaching full information consensus for these types of algorithms is both necessary and potentially time consuming. The basic implicit coordination approach is then extended to achieve better performance with imperfect data synchronization.

In the implicit coordination framework, UAVs communicate with each other and apply the consensus algorithm until they reach full consensus. Having reached consensus, every UAV implements a task assignment algorithm and creates the optimal plan for the team and then implements its own plan. Since everything (information, algorithms used) is identical in all UAVs, the resulting team plans will be identical and therefore the implemented plan will also be optimal and conflict-free.

This thesis proposes a decentralized task assignment algorithm that is robust to the inconsistency of the information and ensures that the resulting decentralized plan is conflict-free. The resulting robust decentralized task assignment (RDTA) method assumes some degree of data synchronization, but adds a second planning step based on sharing the planning data. The approach is analogous to closing a synchronization loop on the planning process to reduce the sensitivity to exogenous disturbances.

Since the planning in RDTA is done based on consistent, pre-generated plans, it ensures that there are no conflicts in the final plans selected independently. Also, each UAV will execute a plan that it created; thus it is guaranteed to be feasible for that vehicle. Furthermore, communicating a small set of candidate plans helps overcome any residual disparity in the information at the end of the information update phase. This improves the possibility of finding a group solution that is close to the optimal, while avoiding the communication overload that would be associated with exchanging all possible candidates. There are two tuning knobs in the RDTA that makes it a flexible task assignment algorithm. These knobs can be set for different mission scenarios and different environments, so that the resulting plan meets the objectives of the mission. For instance, the algorithm can increase the performance and achieve an optimal solution if the required communication is provided, while for the cases where the communication resources are limited, the RDTA algorithm still performs well and produces feasible, conflict-free assignments.

1.3.3 Auction-Based Task Assignment

Although the RDTA algorithm performs well with most communication networks, it has a minimum requirement. For instance, in the second stage of the algorithm, the set of candidate plans is transmitted to every other UAV in the team. For networks that are not complete (i.e., there does not exist a link from each UAV to every other

UAV), the UAVs must be able to relay the information received from one neighbor to another. The requirement of having relaying capability does not impose a major limitation and UAVs usually have this capability. However, if this requirement is not satisfied, it can result in a significant performance degradation of the RDTA algorithm.

Although the implicit coordination algorithm does not require this type of network connectivity or relaying capability, it can make inefficient use of the communication network. It was mentioned that, for the implicit coordination algorithm to perform well, UAVs have to run a consensus algorithm to reach perfectly consistent information. However, reaching consistent information can be cumbersome if the information set is large, which is usually the case for a realistically sized problem. Note that this limitation was one of the motivations that led to the development of the RDTA algorithm.

An auction-based task assignment (ABTA) algorithm is developed in this thesis that eliminates both limitations discussed above. It performs the assignment in a completely decentralized manner, where each UAV is only allowed to communicate with its neighboring UAVs and there is no relaying. In contrast to a basic auction algorithm, where one agent is required to gather all the bids and assign the task to the highest bidding agent, the ABTA algorithm does everything locally. This feature enables the algorithm to create the task plan without any relaying of information between agents. At the same time, the algorithm is communication efficient in the sense that only the necessary information is communicated, and this enables the approach to be used in low bandwidth communication networks.

Simulation results show that although the solution of the ABTA algorithm is not optimal, it is very close to optimal and for most cases the sub-optimality is less than 2%. The results also show that the algorithm performs well with sparse communication networks and its advantages over the implicit coordination algorithm become more apparent for sparse communication networks.

1.3.4 Robust Planning

This thesis extends prior work that investigated the issues of planner robustness and adaptability [3, 18]. In particular, the individual advantages of robust planning [18] and Filter Embedded Task Assignment (FETA) [3] are combined to produce a new Robust Filter Embedded Task Assignment (RFETA) formulation that modifies the previous approaches to robustly plan missions while mitigating the effects of churning. Note that the previous algorithms were designed to operate under specific assumptions. The robust planning techniques, while accounting for cost uncertainty in the optimization, did not assume online collection of measurements during the entire mission, and hence did not include any notion of re-planning. Likewise, the FETA algorithm, while incorporating concepts of replanning due to noisy sensors, did not explicitly account for the cost uncertainty (i.e., target identity uncertainty). The new RFETA algorithm combines robust planning with online observations, and is therefore a more general algorithm that relies on fewer modeling assumptions. Extensive simulation results are presented to demonstrate the improvements provided by the new approach.

In Refs. [2, 3] we provided a new algorithm that accounts for changes in the SA during replanning. In this chapter, we extend this algorithm and show that the modified algorithm demonstrates the desired filtering behavior. Further analysis is provided to show these properties. The main contribution of this chapter is combining the robust planning [18] and adaptive approaches to develop a fully integrated solution to the UAV task planning problem, and discussing the interactions between the two techniques in a detailed simulation. The resulting Robust Filter Embedded Task Assignment (RFETA) is shown to provide an algorithm that is well suited for real-time calculation and yields superior performance to using robustness or FETA alone.

Chapter 2

Kalman Consensus Algorithm

2.1 Introduction

Coordinated planning for a group of agents has been given significant attention in recent research [4, 10, 21, 23, 25, 52, 72]. This includes work on various planning architectures, such as distributed [4, 25], hierarchic [21, 23], and centralized [10, 52, 72]. In a centralized planning scheme, all of the agents communicate with a central agent to report their information and new measurements. The central planner gathers this available information to produce coordinated plans for all agents, which are then redistributed to the team. Note that generating a coordinated plan using a centralized approach can be computationally intensive, but otherwise it is relatively straight forward because the central planner has access to all information. However, this approach is often not practical due to communication limits, robustness issues, and poor scalability [4, 25]. Thus attention has also focused on distributed planning approaches, but this process is complicated by the extent to which the agents must share their information to develop coordinated plans. This complexity can be a result of dynamic or risky environments or strong coupling between tasks, such as tight timing constraints. One proposed approach to coordinated distributed planning is to have the agents share their information to reach *consensus* and then plan independently [25].

Several different algorithms have been developed in the literature for agents to reach consensus [8, 20, 38, 59, 60, 61, 67, 68, 69] for a wide range of static and dynamic communication structures. In particular, a recent paper by Ren *et al.* [68] uses the well known Kalman filtering approach to develop the Kalman Consensus Algorithm (KCA) for both continuous and discrete updates and presents numerical examples and analytical proofs to show their convergence.

The objective of this chapter is to extend the algorithm developed in Ref. [68] to not only ensure its convergence for the general form of communication networks, but also ensure that the algorithm converges to the desired value. In the Kalman Consensus Algorithm the desired value is the value that is achieved if a centralized Kalman filter was applied to the initial information of the agents. We show, both by simulation and analytical proofs, that the new extended algorithm always converges to the desired value.

The main contribution of this chapter is developing a Kalman Consensus Algorithm that gives an unbiased estimate of the desired value for static and dynamic communication networks. The proof of convergence of the new *Unbiased Decentralized Kalman Consensus* (UDKC) algorithm to this unbiased estimate is then provided for both static and dynamic communication networks. Since the desired value in Kalman Consensus is essentially a weighted average of the initial information of the agents, the proposed algorithm can also be used to achieve a general weighted average for the very general form of communication networks. Previous research had shown that the weighted average can only be achieved for the special case of strongly connected balanced networks. Another contribution of this chapter is showing that these constraints on the network can be relaxed and the proposed algorithm still reaches the desired weighted average.

Section 2.2 provides some background on the consensus problem and Section 2.3 formulates the Kalman Consensus Algorithm and discusses the convergence properties. The new extension to the Kalman Consensus Algorithm is formulated in Section 2.4 and more examples are given to show its convergence to an unbiased estimate. Finally, the proof of convergence to an unbiased estimate for static and dynamic communication structure is given.

2.2 Consensus Problem

This section presents the consensus problem statement and discusses some common algorithms for this problem [8, 20, 38, 59, 60, 61, 67, 68, 69].

2.2.1 Problem Statement

Suppose there are n agents $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ with inconsistent information and let x_i be the information associated with agent i. The objective is for the agents to communicate this information amongst themselves to reach consensus, which means that all of the agents have the same information ($x_i = x_j,\ \forall i, j \in \{1, \ldots, n\}$).

To simplify the notation in this chapter, we assume that the information is a scalar value, but the results can be easily extended to the case of a vector of information.

The communication pattern at any time t can be described in terms of a directed graph $\mathbb{G}(t) = (\mathcal{A}, \mathcal{E}(t))$, where $(\mathcal{A}_i, \mathcal{A}_j) \in \mathcal{E}(t)$ if and only if there is a unidirectional information exchange link from $\mathcal{A}_i$ to $\mathcal{A}_j$ at time t. Here we assume that there is a link from each agent to itself, $(\mathcal{A}_i, \mathcal{A}_i) \in \mathcal{E}(t)$, $\forall\ i, t$. The adjacency matrix $G(t) = [g_{ij}(t)]$ of a graph $\mathbb{G}(t)$ is defined as

$$g_{ij}(t) = \begin{cases} 1 & \text{if} \quad (\mathcal{A}_j, \mathcal{A}_i) \in \mathcal{E}(t) \\ 0 & \text{if} \quad (\mathcal{A}_j, \mathcal{A}_i) \notin \mathcal{E}(t) \end{cases} \tag{2.1}$$

and a directed path from $\mathcal{A}_i$ to $\mathcal{A}_j$ is a sequence of ordered links (edges) in $\mathcal{E}$ of the form $(\mathcal{A}_i, \mathcal{A}_{i_1}), (\mathcal{A}_{i_1}, \mathcal{A}_{i_2}), \ldots, (\mathcal{A}_{i_r}, \mathcal{A}_j)$. A directed graph $\mathbb{G}$ is called strongly connected if there is a directed path from any node to all other nodes [31] and a balanced network is defined as a network where for any node $\mathcal{A}_i$, its outflow equals its inflow.

2.2.2 Consensus Algorithm

If the information, x_i of agent $\mathcal{A}_i$, is updated in discrete time steps using the data communicated from the other agents, then the update law can be written as

$$x_i(t+1) = x_i(t) + \sum_{j=1}^{N} \alpha_{ij}(t) g_{ij}(t)(x_j(t) - x_i(t)) \tag{2.2}$$

where $\alpha_{ij}(t) \geq 0$ represents the relative effect of information of agent $\mathcal{A}_j$ on the information of agent $\mathcal{A}_i$. The parameter $\alpha_{ij}(t)$ can be interpreted as the relative confidence that agent $\mathcal{A}_i$ and $\mathcal{A}_j$ have that their information variables are correct [8]. Equation (2.2) can also be written in matrix form as $\mathbf{x}(t+1) = A(t)\mathbf{x}(t)$, where $\mathbf{x}(t) = [x_1(t), \ldots, x_n(t)]^T$, and the $n \times n$ matrix $A(t) = [a_{ij}(t)]$ is given by

$$a_{ij}(t) \begin{cases} \geq 0 & \text{if} \quad g_{ij}(t) = 1 \\ = 0 & \text{if} \quad g_{ij}(t) = 0 \end{cases} \tag{2.3}$$

Several methods such as *Fixed Coefficients, Vicsek Model, Gossip Algorithm* and *Kalman Filtering* have been proposed to pick values for the matrix A [20], [68]. In the Kalman Filtering approach, the coefficients, a_{ij}, are chosen to account for the uncertainty each agent has in its information. Section 2.3.1 summarizes the Kalman filter formulation of consensus problem from Ref. [68]. Simulations are then presented to show that the performance of this algorithm strongly depends on the structure of the communication network. An extension to this algorithm is proposed in Section 2.4 that is shown to work for more general communication networks.

2.3 Kalman Consensus Formulation

This section provides a brief summary of Ref. [68], which uses Kalman Filtering concepts to formulate the consensus problem for a multi-agent system with static information.

2.3.1 Kalman Consensus

Suppose at time t, $x_i(t)$ represents the information (perception) of agent $\mathcal{A}_i$ about a parameter with the true value x^*. This constant true value is modeled as the state, $x^*(t)$, of a system with trivial dynamics and a zero-mean disturbance input $w \sim (0, Q)$,

$$x^*(t+1) = x^*(t) + w(t)$$

The measurements for agents $\mathcal{A}_i$ at time t are the information that it receives from other agents,

$$z_i(t) = \begin{bmatrix} g_{i1}(t)x_1(t) \\ \vdots \\ g_{in}(t)x_n(t) \end{bmatrix} \tag{2.4}$$

where $g_{ij}(t) = 1$ if there is a communication link at time t from agent $\mathcal{A}_j$ to $\mathcal{A}_i$, and 0 otherwise. Assuming that the agents' initial estimation errors, $(x_i(0) - x^*)$, are uncorrelated, $E[(x_i(0) - x^*)(x_j(0) - x^*)^T] = 0$, $i \neq j$ and by defining

$$P_i(0) = E[(x_i(0) - x^*)(x_i(0) - x^*)^T]$$

then the discrete-time Kalman Consensus Algorithm for agent i can be written as [68]

$$\begin{aligned} P_i(t+1) &= \left\{ [P_i(t) + Q(t)]^{-1} + \sum_{j=1, j\neq i}^{n} g_{ij}(t) [P_j(t)]^{-1} \right\}^{-1} \\ x_i(t+1) &= x_i(t) + P_i(t+1) \sum_{j=1, j\neq i}^{n} \left\{ g_{ij}(t) [P_j(t)]^{-1} [x_j(t) - x_i(t)] \right\} \end{aligned} \tag{2.5}$$

Since it is assumed that $g_{ii} = 1$, then to make the formulation similar to the one in Ref. [68], i is excluded from the summations $(j \neq i)$ in the above equations. Equations (2.5) are applied recursively until all the agents converge in their information or, equivalently, consensus is reached $(t = 1, \ldots, T_{consensus})$. Note that, although $P_i(0)$ represents the initial covariance of $x_i(0)$, the values $P_i(t)$; $t > 0$ need not have the same interpretation – they are just weights used in the algorithm that are modified using the covariance update procedure of the Kalman filter.

Ref. [68] shows that under certain conditions the proposed Kalman Consensus Algorithm converges and the converged value is based on the confidence of each agent about the information. The following sections analyze the performance of this algorithm for different network structures and modifications are proposed to improve the convergence properties.

2.3.2 Centralized Kalman Consensus

The centralized Kalman estimator for the consensus problem is formulated in this section to be used as a benchmark to evaluate different distributed algorithms. Since the centralized solution is achieved in one iteration ($T_{Consensus} = 1$) and the decentralized solution is solved over multiple iterations ($T_{Consensus} > 1$), some assumptions are necessary to enable a comparison between the two algorithms. In particular, since the process noise is added in each iteration and the centralized solution is done in one step, consistent comparisons can only be done if the process noise is zero ($w(t) = 0$; $\forall\ t$). These assumptions are made solely to enable a comparison of different algorithms with the benchmark (centralized), and they do not impose any limitations on the algorithm that will be developed in the next sections. Under these assumptions, the centralized solution using the Kalman filter is

$$\begin{aligned} \bar{P} &= \left\{ \sum_{i=1}^{n} [P_i(0)]^{-1} \right\}^{-1} \\ \bar{x} &= \bar{P} \sum_{i=1}^{n} \left\{ [P_i(0)]^{-1} x_i(0) \right\} \end{aligned} \tag{2.6}$$

2.3.3 Example

The *meet-for-dinner* example [68] is used in this chapter as a benchmark to compare the performance (accuracy) of different algorithms. In this problem, a group of friends decide to meet for dinner, but fail to specify a precise time to meet. On the afternoon of the dinner appointment, each individual realizes that he is uncertain about the time of dinner. A centralized solution to this problem is to have a conference call and decide on the time by some kind of averaging on their preferences. Since the conference call is not always possible, a decentralized solution is required. In the decentralized solution, individuals contact each other (call, leave messages) and iterate to converge to a time (reach consensus). Here the Kalman Consensus algorithm from Section 2.3.1 is used to solve this problem for $n = 10$ agents. Figure 2.1 shows the output of this algorithm for the two cases presented in Ref. [68], demonstrating that the results obtained are consistent. These simulations use a special case of a balanced communication network in which each agent communicates with exactly one other agent so that

$$\text{Inflow}(\mathcal{A}_i) = \text{Outflow}(\mathcal{A}_i) = 1, \quad \forall \;\; \mathcal{A}_i \in \mathcal{A} \tag{2.7}$$

where Inflow($\mathcal{A}_i$) is the number of links of the form $(\mathcal{A}_j, \mathcal{A}_i) \in \mathcal{E}$ and Outflow($\mathcal{A}_i$) is the number of links of the form $(\mathcal{A}_i, \mathcal{A}_j) \in \mathcal{E}$.

In the left plot of Figure 2.1, the initial states and the initial variances are uniformly assigned (Case 1). In the right plot, the variance of the agent with initial data $x_i(0) = 7$ (leader) is given an initial variance of $P_i(0) = 0.001$, which is significantly lower than the other agents and therefore has more weight on the final estimate (Case 2). To evaluate the performance of this algorithm, the results are compared to the true estimate, $\bar{x}$, calculated from the centralized algorithm in (2.6). The results in Table 2.1 clearly show that the solution to the decentralized algorithm in (2.5) is identical to the true centralized estimate.

As noted, these cases assume the special case of the communication networks in (2.7). To investigate the performance of the decentralized algorithm in more general cases, similar examples were used with slightly different communication networks. The graphs associated with these new architectures are still strongly connected, but the assumption in (2.7) is relaxed. This is accomplished using the original graphs of Cases 1 and 2 with four extra links added to the original graph. The results are presented in Table 2.1 (Cases 3, 4). For these cases, the solution of the decentralized

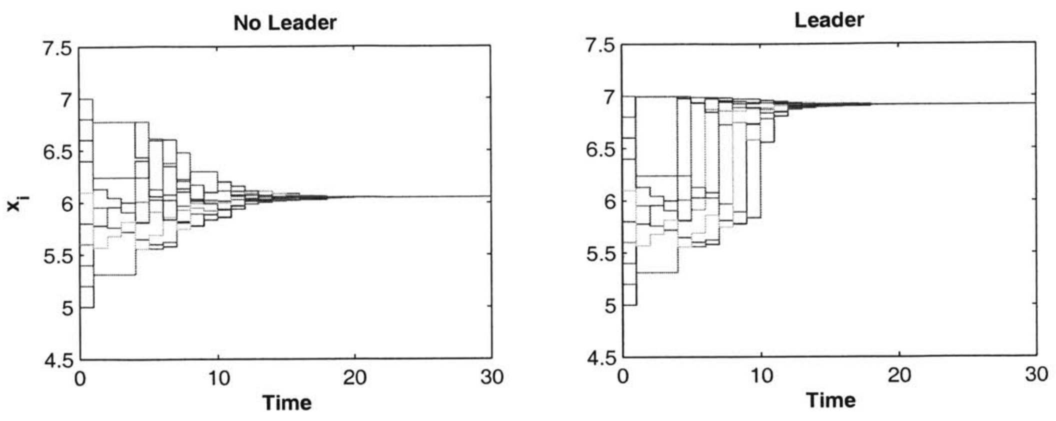


Figure 2.1: The result of Kalman consensus algorithm for cases 1 and 2, demonstrating consistency with the results in Ref. [68].

algorithm of (2.5) deviates from the true estimate, $\bar{x}$, obtained from the centralized solution. The Kalman Consensus Algorithm always converges to a value that respects the certainty of each agent about the information, but these results show that in cases for which the network does not satisfy the condition of (2.7), the consensus value can be biased and deviate from the centralized solution.

The next section extends this algorithm to eliminate this bias and to guarantee convergence to the true centralized estimate, $\bar{x}$, for the general case of communication networks.

2.4 Unbiased Decentralized Kalman Consensus

This section extends the Kalman Consensus formulation of (2.5) to achieve the desired unbiased solution, which is the solution to the centralized algorithm presented in (2.6). The new extended algorithm generates the true centralized estimate, $\bar{x}$, using a decentralized estimator for any form of communication networks.

The main idea is to scale the accuracy of the agents by their outflow, which gives the *Unbiased Decentralized Kalman Consensus* (UDKC) algorithm. For agent $\mathcal{A}_i$ at

Table 2.1: Comparing the results of different algorithms.

Algorithm	Case 1	Case 2	Case 3	Case 4
Centralized	6.0433	6.9142	6.0433	6.9142
Kalman Consensus	6.0433	6.9142	5.6598	6.2516
UDKC	6.0433	6.9142	6.0433	6.9142

time $t+1$, the solution is given by

$$\begin{aligned} P_i(t+1) &= \left\{ [P_i(t)+Q(t)]^{-1} + \sum_{j=1}^{n} \left(g_{ij}(t) \left[\mu_j(t) P_j(t) \right]^{-1} \right) \right\}^{-1} \\ x_i(t+1) &= x_i(t) + P_i(t+1) \sum_{j=1}^{n} \left\{ g_{ij}(t) \left[\mu_j(t) P_j(t) \right]^{-1} [x_j(t) - x_i(t)] \right\} \end{aligned} \tag{2.8}$$

where $\mu_j(t)$ is the scaling factor associated with agent $\mathcal{A}_j$ and,

$$\mu_j(t) = \sum_{k=1,\ k\neq j}^{n} g_{kj}(t) \tag{2.9}$$

To show the unbiased convergence of the UDKC algorithm, the four cases of the meet-for-dinner problem in Section 2.3.3 were re-solved using this new approach. The results for the four cases are presented in Table 2.1. As shown, in all four cases the UDKC algorithm converges to the true estimates (the results of the centralized algorithm). The following remarks provide further details on the UDKC algorithm.

i) Both the original KCA and new UDKC formulations presented here differ from the previously developed weighted average consensus algorithms [62] in the sense that these algorithms not only update the information in each iteration, but also update the weights (P's) that are used in the formulation. This additional update (Eqs. 2.5 and 2.8) enables the UDKC algorithm to converge to the desired weighted average for a very general class of communication networks, while the previous form of consensus algorithm (Eq. 2.2), where only the information itself gets updated at each iteration [62], was limited to a special kind of strongly connected balanced network.

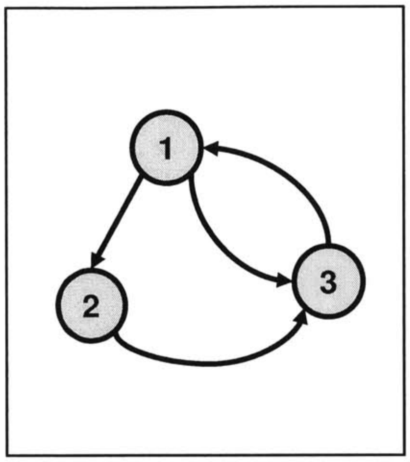


Figure 2.2: A simple imbalanced network with unequal outflows.

ii) Reference [62] introduces an alternative form of the consensus algorithm that has some apparent similarities to the UDKC formulation introduced in this chapter. The form of the consensus algorithm in [62] is as follows:

$$\dot{x}_i = \frac{1}{|N_i|} \sum_{j \in N_i} (x_j - x_i) \tag{2.10}$$

where $N_i = \{j \in \mathcal{A} : (i,j) \in \mathcal{E}\}$ is the list of neighbors of agent $\mathcal{A}_i$. Note that in the notation of [62], if $(i,j) \in \mathcal{E}$ then there is a link from $\mathcal{A}_i$ to $\mathcal{A}_j$ but the information flow is from $\mathcal{A}_j$ to $\mathcal{A}_i$. Therefore, although $|N_i|$ is defined as the outdegree of agent $\mathcal{A}_i$, it is essentially the inflow of agent $\mathcal{A}_i$ in our formulation. Thus the consensus formulation of Eq. 2.10 has a scaling factor that is equal to the inflow of the receiving agent, $\mathcal{A}_i$. Note however, that the scaling factor in the UDKC algorithm (the coefficient μ in Eq. 2.8) is the outflow of the sending agent, $\mathcal{A}_j$. This clarifies the key differences between UDKC and the method introduced in Ref. [62].

iii) The scaling introduced in UDKC (the coefficient μ in Eq. 2.8) does not change the topology of the network to make it a balanced network. The implicit effect of μ is essentially making the outflows of all agents equal to 1 and has no effect on the inflow of the agents. Thus the resulting network will not necessarily be a balanced network and therefore the results presented in Refs. [59, 62] for balanced networks can not be used to prove the convergence of the UDKC algorithm to the desired weighted average. Figure 2.2 shows a simple network that is neither balanced (Inflow(1) $= 1$, Outflow(1) $= 2$) nor are its outflows equal (Outflow(1) $= 2$, Outflow(2) $= 1$). The adjacency matrix for this network is:

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \tag{2.11}$$

and applying the scaling μ defined in Eq. 2.9 gives

$$\begin{bmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0 \\ 0.5 & 1 & 0 \end{bmatrix} \tag{2.12}$$

which has the same outflow for all the nodes, but is still imbalanced (Inflow(2) $=$ 0.5, Outflow(2) $= 1$).

To show why the outflow scaling results in convergence to the desired solution, a simple example is presented here. Based on the Kalman filter, the relative weights given to each estimate should be relative to the accuracy of the estimates, P_i's (see (2.6)). The formulation in (2.5) uses the same idea, but these weights are further scaled by the outflow of the agents. This means that if agent $\mathcal{A}_i$ and $\mathcal{A}_j$ have exactly the same accuracy, $P_i = P_j$, but in addition the outflow of agent $\mathcal{A}_i$ is greater than the outflow of agent $\mathcal{A}_j$, then using (2.5) causes the information of agent $\mathcal{A}_i$ to be treated as if it is more accurate than information of $\mathcal{A}_j$ (or the effective value of P_i is less than P_j), which creates a bias in the converged estimate. Obviously, for the special balanced networks considered in the simulations of Ref. [68], this bias does not occur since the outflows are all equal to one.

Figure 2.3 presents a simple example to illustrate the problem with the Kalman

Consensus Algorithm of (2.5). There are 3 agents with $[x_1(0)\, x_2(0)\, x_3(0)] = [\ 4 \quad 5 \quad 6\ ]$ and ($P_i(0) = 1,\ i \in \{1,2,3\}$). As shown in the figure, the outflows of agents 2 and 3 are both one, but it is two for agent 1. Since all agents have the same initial accuracy, the centralized solution is the average of the initial estimates, $\bar{x} = 5$. Figure 2.3 shows four steps of the Kalman Consensus Algorithm for this example. At time $t = 3$, all of the estimates are less than 5, and the final converged estimate is 4.89, which is different from the centralized estimate. Note also that the deviation of the final value from the correct estimate is towards the initial value of agent 1, which has the largest outflow. This bias is essentially the result of an imbalanced network in which information of agents with different outflows is accounted for in the estimation with different weights. In order to eliminate the bias, weights should be modified to cancel the effect of different outflows, which is essentially the modification that is introduced in (2.8).

The following sections present the proof of convergence of the UDKC algorithm to the true centralized estimate.

2.4.1 Information Form of UDKC

The information form of Kalman Filtering is used to prove that the UDKC algorithm converges to the true centralized estimate, $\bar{x}$, in (2.6). The information filter is an equivalent form of the Kalman filter that simplifies the measurement update, but complicates the propagation [56]. It is typically used in systems with a large measurement vector, such as sensor fusion problems [42, 43]. Since the propagation part of the Kalman filter is absent (or very simple) in the consensus problem, the information form of the filter also simplifies the formulation of that problem. The following briefly presents the information form of the Kalman consensus problem. To be consistent with the example in Section 2.3.3, it is assumed that the process noise is zero. To write the UDKC (2.8) in the information form, for agent $\mathcal{A}_i$ define

$$Y_i(t) \equiv P_i(t)^{-1} \text{ and } y_i(t) \equiv Y_i(t)x_i(t) \tag{2.13}$$

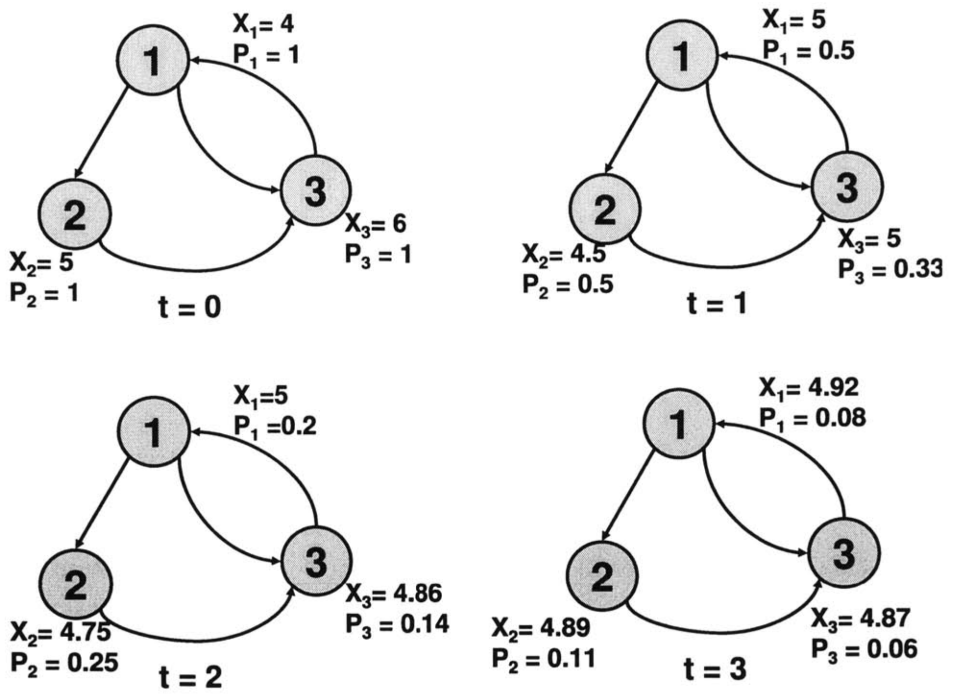


Figure 2.3: An example to show the bias of the Decentralized Kalman Consensus Algorithm, $x_i(t)$ and $P_i(t)$ are the estimate and its accuracy of agent $\mathcal{A}_i$ at time t.

then, (2.8) can be written as

$$Y_i(t+1) = \frac{1}{2}\left\{ Y_i(t) + \sum_{j=1, j\neq i}^{n} \frac{g_{ij}(t)}{\mu_j(t)} Y_j(t) \right\} \tag{2.14}$$

$$y_i(t+1) = \frac{1}{2}\left\{ y_i(t) + \sum_{j=1, j\neq i}^{n} \frac{g_{ij}(t)}{\mu_j(t)} y_j(t) \right\} \tag{2.15}$$

and after each iteration (time t), for agent $\mathcal{A}_i$

$$x_i(t) = Y_i(t)^{-1} y_i(t) \tag{2.16}$$

Note that the expressions in (2.15) are scaled by a factor of $1/2$, which has no effect on the estimation, but simplifies later proofs. These equations can be written in matrix form,

$$\mathbf{Y}(t+1) = \Psi(t)\mathbf{Y}(t) \tag{2.17}$$

$$\mathbf{y}(t+1) = \Psi(t)\mathbf{y}(t) \tag{2.18}$$

where $\mathbf{Y}(t) = [Y_1(t), \dots, Y_n(t)]^T$, $\mathbf{y}(t) = [y_1(t), \dots, y_n(t)]^T$ and $\Psi(t) = [\psi_{ij}(t)]$ with

$$\psi_{ij}(t) = \begin{cases} \frac{1}{2} & \text{if } \ j = i \\ \frac{g_{ij}(t)}{2\mu_j(t)} & \text{if } \ j \neq i \end{cases} \tag{2.19}$$

A comparison of the simple linear update in (2.17) and (2.18) with the nonlinear updates of the Kalman filter (2.8) shows the simplicity of this information form for the consensus problem. Note that since agents iterate on communicating and updating their information before using it, the inversions in (2.13) and (2.16) do not need to be performed every iteration. At the beginning of the consensus process, each agent $\mathcal{A}_i$ transforms its initial information, $x_i(0)$, and associated accuracy, $P_i(0)$, to $y_i(0)$ and $Y_i(0)$ using (2.13). In each following iteration, the transformed values ($y_i(t)$, $Y_i(t)$) are communicated to other agents and are used in the update process of (2.15). At the end of the consensus process the state $x_i(T_{consensus})$ can be extracted from $y_i(T_{consensus})$ and $Y_i(T_{consensus})$ using (2.16).

2.4.2 Proof of Unbiased Convergence

This section provides the results necessary to support the proof of convergence of the UDKC algorithm to an unbiased estimate in the absence of noise.

Definition 2.1 ([73]) *A nonnegative matrix $A = [a_{ij}] \in \mathbb{C}^{n \times n}$ is called row stochastic if $\sum_{j=1}^{n} a_{ij} = 1$, $1 \leq i \leq n$ and it is called column stochastic if $\sum_{i=1}^{n} a_{ij} = 1$, $1 \leq j \leq n$. Note that if A is a row stochastic matrix, A^T is a column stochastic matrix.*

Theorem 2.1 ([73]) *If we denote by $\mathbf{e} \in \mathbb{R}^n$ the vector with all components $+1$, a* *nonnegative matrix A is row stochastic if and only if $A\mathbf{e} = \mathbf{e}$.*

Lemma 2.1 *The matrix $\Psi(t) = [\psi_{ij}(t)]$ defined in (2.19) is column stochastic.*

Proof: *For any column j,*

$$\sum_{i=1}^{n} \psi_{ij}(t) = \frac{1}{2}\left(1 + \sum_{i=1, i\neq j}^{n} \frac{g_{ij}(t)}{\mu_j(t)}\right) = \frac{1}{2}\left(1 + \frac{1}{\mu_j(t)} \sum_{i=1, i\neq j}^{n} g_{ij}(t)\right) \tag{2.20}$$

Thus using (2.9)

$$\sum_{i=1}^{n} \psi_{ij}(t) = \frac{1}{2}\left(1 + \frac{1}{\mu_j(t)}\mu_j(t)\right) = 1 \tag{2.21}$$

so Ψ is column stochastic.

Lemma 2.2 *The directed graph associated with matrix $\Psi = [\psi_{ij}]$ defined in (2.19), is strongly connected.*

Proof: *By definition (2.19), $\psi_{ij} > 0$ if $g_{ij} > 0$ and $\psi_{ij} = 0$ if $g_{ij} = 0$ and therefore matrices $\Psi = [\psi_{ij}]$ and $G = [g_{ij}]$ are both adjacency matrices to the same graph, which was assumed to be strongly connected.*

Theorem 2.2 ([78]) *For any $A = [a_{ij}] \in \mathbb{C}^{n\times n}$, A is irreducible if and only if its directed graph $\mathbb{G}(A)$ is strongly connected.*

Theorem 2.3 (Perron-Frobenius Theorem, [78]) *Given any $A = [a_{ij}] \in \mathbb{R}^{n\times n}$, with $A \succeq 0$ and with A irreducible, then:*

i) A has a positive real eigenvalue equal to its spectral radius $\rho(A)$;

ii) to $\rho(A)$ there corresponds an eigenvector $\mathbf{v} = [v_1, v_2, \ldots, v_n]^T \succ \mathbf{0}$;

iii) $\rho(A)$ is a simple eigenvalue of A.

Theorem 2.4 (Geršgorin, [35]) *Let $A = [a_{ij}] \in \mathbb{C}^{n\times n}$, and let*

$$R_i(A) \equiv \sum_{j=1, j\neq i}^{n} |a_{ij}|, \qquad 1 \leq i \leq n \tag{2.22}$$

denote the "deleted absolute row sums" of A. Then all the eigenvalues of A are located in the union of n discs

$$\bigcup_{i=1}^{n} \{z \in \mathbb{C} : |z - a_{ii}| \leq R_i(A)\}$$

Definition 2.2 *A nonnegative matrix $A \in \mathbb{C}^{n \times n}$ is said to be "primitive" if it is irreducible and has only one eigenvalue of maximum modulus.*

Theorem 2.5 ([35]) *If $A \in \mathbb{C}^{n \times n}$ is nonnegative and primitive, then*

$$\lim_{m \to \infty} [\rho(A)^{-1} A]^m = L \succ 0$$

where $L = \mathbf{v}\mathbf{u}^T$, $A\mathbf{v} = \rho(A)\mathbf{v}$, $A^T\mathbf{u} = \rho(A)\mathbf{u}$, $\mathbf{v} \succ 0$, $\mathbf{u} \succ 0$, and $\mathbf{v}^T\mathbf{u} = 1$.

Lemma 2.3 *For the matrix $\Psi = [\psi_{ij}]$ defined in (2.19),*

$$\lim_{m \to \infty} \Psi^m = \mathbf{v}\mathbf{e}^T \succ 0$$

where $\mathbf{v}$ is a column vector and for the matrix C, $C \succ 0$ means that $c_{ij} > 0 \ \forall\ i, j$.

Proof: *By definition $\Psi \succeq 0$ ($\psi_{ij} \geq 0$), and the directed graph associated with it is strongly connected (*Lemma 2.2*), so from* Theorem 2.2, *Ψ is irreducible. Thus Ψ has a simple eigenvalue equal to $\rho(\Psi)$ (*Theorem 2.3*).*

*Furthermore, Ψ is column stochastic (*Lemma 2.1*) and by definition Ψ has an eigenvalue $\lambda_1 = 1$ (*Theorem 2.1*). Using the Geršgorin Theorem (*Theorem 2.4*), all of the eigenvalues of the row-stochastic matrix Ψ^T are located in the union of n disks*

$$\bigcup_{i=1}^{n} \{z \in \mathbb{C} : |z - \psi_{ii}| \leq R_i(\Psi^T)\}$$

Using (2.19), $\psi_{ii} = 0.5$, $\forall i$, and $R_i(\Psi^T) = 0.5$ (see (2.22)), and thus all the eigenvalues of Ψ^T and Ψ are located in the disc $\{z \in \mathbb{C} : |z - 0.5| \leq 0.5\}$. Consequently all the eigenvalues of Ψ satisfy $|\lambda_i| \leq 1$, $\forall i$, and hence $\rho(\Psi) \leq 1$.

Since $\lambda_1 = 1$, *therefore* $\lambda_1 = \rho(\Psi) = 1$. *As a result* Ψ *has only one eigenvalue of maximum modulus and therefore is primitive (see* Definition 2.2*). Finally, using* Theorem 2.5,

$$\lim_{m\to\infty} [\rho(\Psi)^{-1}\Psi]^m = L \succ 0$$

where $L = \mathbf{v}\mathbf{u}^T$, $\Psi\mathbf{v} = \rho(\Psi)\mathbf{v}$, $\Psi^T\mathbf{u} = \rho(\Psi)\mathbf{u}$, $\mathbf{v} \succ 0$, $\mathbf{u} \succ 0$, *and* $\mathbf{v}^T\mathbf{u} = 1$. *However, since* $\rho(\Psi) = 1$, *and using* Theorem 2.1, $\mathbf{u} = \mathbf{e}$, *then it follows that* $\lim_{m\to\infty} \Psi^m = \mathbf{v}\mathbf{e}^T \succ 0$.

With these results, we can now state the main result of the chapter.

Theorem 2.6 *For any strongly connected, time-invariant communication network,* $\mathbb{G}$, *and for any agent* $\mathcal{A}_i$ *and any initial estimate,* $x_i(0)$, *and variance,* $P_i(0)$, *the estimate,* $x_i(t)$, *resulting from the Modified Distributed Kalman Consensus Algorithm introduced in (2.8) and (2.15), converges to the true centralized estimate,* $\bar{x}$, *calculated using (2.6), or equivalently,*

$$\lim_{t\to\infty} x_i(t) \to \bar{x} \qquad \forall i \in \{1, \ldots, n\} \tag{2.23}$$

Proof: *The objective is to show that (2.23) is satisfied or equivalently,* $\lim_{t\to\infty} \mathbf{x}(t) \to \bar{x}\mathbf{e}$, *where* $\mathbf{x} = [\ x_1, \ldots, x_n\ ]^T$. *Let* $\mathbf{v}^\dagger$ *denote the element inverse of a vector,* $\mathbf{v}^\dagger = [v_1^{-1}, \ldots, v_n^{-1}]^T$. *Using (2.16) it follows that* $\lim_{t\to\infty} \mathbf{x}(t) = \lim_{t\to\infty} \mathbf{Y}^\dagger(t) \odot \mathbf{y}(t)$, *where the operator* $\odot$ *represents the element by element multiplication. With the assumed time-invariance of the communication network,* $\Psi(t) = \Psi$, *and using (2.17) and (2.18)*

$$\lim_{t\to\infty} \mathbf{x}(t) = \lim_{t\to\infty} \left(\Psi^t \mathbf{Y}(0)\right)^\dagger \odot \left(\Psi^t \mathbf{y}(0)\right)$$

Using Lemma 2.3,

$$\begin{aligned}\lim_{t\to\infty} \mathbf{x}(t) &= \left(\mathbf{v}\underbrace{\mathbf{e}^T\mathbf{Y}(0)}_{\text{scalar}}\right)^\dagger \odot \left(\mathbf{v}\underbrace{\mathbf{e}^T\mathbf{y}(0)}_{\text{scalar}}\right) \\ &= \left(\mathbf{e}^T\mathbf{Y}(0)\right)^{-1} \left(\mathbf{v}^\dagger \odot \mathbf{v}\right) \left(\mathbf{e}^T\mathbf{y}(0)\right)\end{aligned}$$

Since $\mathbf{v}^T\mathbf{e} \succ 0$ *(*Lemma 2.3*),* $\mathbf{v} \succ 0$, *therefore,* $\mathbf{v}^\dagger \odot \mathbf{v} = \mathbf{e}$ *and*

$$\begin{aligned}\lim_{t\to\infty} \mathbf{x}(t) &= \left(\mathbf{e}^T\mathbf{Y}(0)\right)^{-1}\left(\mathbf{e}^T\mathbf{y}(0)\right)\mathbf{e} \\ &= \left\{\sum_{i=1}^{n} Y_i(0)\right\}^{-1}\left\{\sum_{i=1}^{n} y_i(0)\right\}\mathbf{e}\end{aligned}$$

Using the relationship $Y_i(0) = P_i(0)^{-1}$, *it follows that*

$$\lim_{t\to\infty} \mathbf{x}(t) = \left\{\sum_{i=1}^{n} P_i(0)^{-1}\right\}^{-1}\left\{\sum_{i=1}^{n} P_i(0)^{-1}x_i(0)\right\}\mathbf{e}$$

and then from (2.6), $\lim_{t\to\infty}\mathbf{x}(t) = \bar{x}\mathbf{e}$. *Thus the UDKC algorithm introduced in (2.8) converges to the true centralized estimate,* $\bar{x}$, *when the strongly connected communication network is time-invariant.*

In what follows we prove that the same is true for a time-varying communication network.

Definition 2.3 (**[79]**) *A stochastic matrix A is called indecomposable and aperiodic (SIA) if*

$$L = \lim_{m\to\infty} A^m$$

exists and all the rows of L are the same. Define $\delta(A)$ *by*

$$\delta(A) = \max_{j}\max_{i,k}|a_{ij} - a_{kj}|$$

Note that if the rows of A are identical, $\delta(A) = 0$, *and vice versa.*

Definition 2.4 *Let* $A_1, \ldots, A_k \in \mathbb{C}^{n\times n}$. *By a* word *in* A_i*'s of the length t we mean the product of t* A_i*'s with repetition permitted.*

Theorem 2.7 (**[79]**) *Let* $A_1, \ldots, A_k$ *be square row-stochastic matrices of the same order such that any word in the* A_i*'s is SIA. For any* $\epsilon > 0$ *there exists an integer* $\nu(\epsilon)$ *such that any word B (in the A's) of length* $m \geq \nu(\epsilon)$ *satisfies* $\delta(B) < \epsilon$.

In other words, the result is that any sufficiently long word in the A_i's has all its rows the same or, $\lim_{m\to\infty} A_1A_2\dots A_m = \mathbf{e}\mathbf{v}^T$.

Lemma 2.4 *If matrices $A_1,\dots,A_N \in \mathbb{R}^{n\times n}$, $\forall i, A_i \succeq 0$ have strictly positive diagonal elements, then matrix $C = A_1A_2\dots A_N$ has the same properties ($C \succeq 0$ and all diagonal elements of C are strictly positive).*

Proof: *To establish this result, it will first be shown that if matrices A, $B \succeq 0$ have strictly positive diagonal elements then $D = AB$ has the same properties. Given that $D = AB$, then*

$$\begin{aligned} d_{ij} &= \sum_{k=1}^{n} \underbrace{a_{ik}b_{kj}}_{\geq 0} \geq 0 \\ d_{ii} &= \sum_{k=1}^{n} a_{ik}b_{ki} = \underbrace{a_{ii}b_{ii}}_{>0} + \sum_{k=1,k\neq i}^{n} \underbrace{a_{ik}b_{ki}}_{\geq 0} > 0 \end{aligned}$$

which provides the necessary result. Therefore by induction, $C = A_1,\dots,A_N \succeq 0$ and all diagonal elements of C are strictly positive.

Theorem 2.8 *Let $\mathbb{G}$ be any dynamic communication network, where at each time step, $\mathbb{G}(t)$ is strongly connected. Then for any agent $\mathcal{A}_i$ and any initial estimate, $x_i(0)$, and variance, $P_i(0)$, the estimate, $x_i(t)$, resulting from the Modified Distributed Kalman Consensus Algorithm, introduced in (2.8) and (2.15), converges to the true centralized estimate, $\bar{x}$, calculated using (2.6).*

Proof: *From* Lemma 2.3, *for any t, $\lim_{m\to\infty}(\Psi^T(t))^m = \mathbf{e}\mathbf{v}_t^T$, where the $\mathbf{v}_t$ is a column vector. Using* (2.19) *and* Lemma 2.1, *$\Psi^T(t)$ is row stochastic, so for any t, $\Psi^T(t)$ is SIA (see* Definition 2.3*). Then from* Theorem 2.7,

$$\lim_{t\to\infty} \Psi^T(1)\Psi^T(2)\dots\Psi^T(t) = \mathbf{e}\mathbf{v}^T$$

for some $\mathbf{v}$, or equivalently,

$$\lim_{t\to\infty} \Psi(t)\Psi(t-1)\dots\Psi(2)\Psi(1) = \mathbf{v}\mathbf{e}^T \qquad (2.24)$$

Thus if it can be shown that $\mathbf{v} \succ 0$*, then the proof of Theorem 2.8 would follow the same steps as the proof for the time-invariant case in Theorem 2.6. To demonstrate that* $\mathbf{v} \succ 0$*, we first show that the diagonal elements of*

$$L = \lim_{t \to \infty} \Psi^T(1)\Psi^T(2)\dots\Psi^T(t) \tag{2.25}$$

are positive ($L_{ii} > 0,\ \forall i$)*. Since, by its definition in (2.19),* $\Psi(t) \succeq 0$ *and all the diagonal elements of* $\Psi(t)$ *are strictly positive, then* $C = \Psi^T(1)\Psi^T(2)\dots\Psi^T(t)$ *and consequently* L *in (2.25) have positive elements,* $L_{ij} \geq 0,\ \forall i,j$*, and strictly positive diagonal elements,* $L_{ii} > 0,\ \forall i$*, (see* Lemma 2.4*).*

Also, since $L = \mathbf{e}\mathbf{v}^T$ *(see (2.24) and (2.25)), then all of the rows of* L *are equal* ($L_{ji} = L_{ii},\ \forall i,j$)*. Furthermore, since* $L_{ii} > 0, \forall i$ *then* $L_{ji} > 0, \forall i,j$*, which implies that* $L = \mathbf{e}\mathbf{v}^T \succ 0$ *and that* $\mathbf{v} \succ 0$*. The remainder of the proof then follows the same steps as the proof for the time-invariant case in Theorem 2.6.*

Convergence Proof for General Network Structure In this section the UDKC is extended to be applied to the general communication networks, which means the strong connectivity assumption is relaxed and more general assumptions are made.

Assumption 1. There exists a positive constant α such that:

(a) $a_{ii}(t) \geq \alpha, \forall i, t.$

(b) $a_{ij}(t) \in \{0\} \cup [\alpha, 1], \forall i, j, t.$

(c) $\sum_{j=1}^{n} a_{ij}(t) = 1, \forall i, t.$

Assumption 2. (connectivity) The graph $(N, \bigcup_{s \geq t} E(s))$ is strongly connected. This assumption says that the union of the graphs from anytime to infinity is strongly connected, which means that when all the future networks are overlapped, then there is a directed graph from any node to any other node.

Assumption 3. (bounded intercommunication interval) If i communicates to j an infinite number of times, then there is some B such that, for all t, $(i,j) \in E(t) \cup E(t+1) \cup \cdots \cup E(t+B-1)$.

Theorem 2.9 *Consider an infinite sequence of stochastic matrices* $A(0), A(1), \dots,$

that satisfies Assumptions 1, 2 and 3. There exists a nonnegative vector v such that,

$$\lim_{t\to\infty} A(t)A(t-1)A(t-2)\dots A(1)A(0) = \mathbf{e}v^T$$

Proof: *See the proof in Reference [20].*

Theorem 2.10 *Let $\mathbb{G}$ be any dynamic communication network that satisfies Assumptions 2 and 3. Then for any agent $\mathcal{A}_i$ and any initial estimate, $x_i(0)$, and variance, $P_i(0)$, the estimate, $x_i(t)$, resulting from the UDKC algorithm, introduced in (2.8) and (2.15), converges to the true centralized estimate, $\bar{x}$, calculated using (2.6).*

Proof: *By construction $\Psi^T(t)$ has the properties of Assumption 1,*

(a) $\psi_{ii}(t) \geq \alpha, \forall i, t.$

(b) $\psi_{ij}(t) \in \{0\} \cup [\alpha, 1], \forall i, j, t.$

(c) $\sum_{i=1}^{n} \psi_{ij}(t) = 1, \forall j, t.$

Therefore all the assumption of theorem 2.9 are satisfied and therefore:

$$\lim_{t\to\infty} \Psi^T(1)\Psi^T(2)\Psi^T(3)\dots\Psi^T(t) = \mathbf{e}v^T$$

and therefore

$$\lim_{t\to\infty} \Psi(t)\Psi(t-1)\Psi(t-2)\dots\Psi(1) = v\mathbf{e}^T$$

The rest of the proof follows the proof of theorem 2.6.

2.5 Conclusions

The performance of the Kalman Consensus Algorithm was investigated for a team of agents with static data. It was shown that, although this algorithm converges for the general case of strongly connected communication networks, it can result in a biased estimate when the outflow of the agents is not equal. An extension to this algorithm was then presented which was shown in simulations to converge to the true centralized estimate for general strongly connected networks. This algorithm was further proved to converge to an unbiased estimate for both static and dynamic communication networks.

Chapter 3

Robust Decentralized Task Assignment

3.1 Introduction

To ensure scalability and flexibility of high-level control systems, various decentralized architectures have been developed for the task assignment process [10, 23, 24, 41]. Within some decentralized frameworks (*e.g., implicit coordination*), each vehicle determines its own mission by simultaneously choosing tasks for all vehicles in the fleet using a centralized planning algorithm [25] and then executing its own plan. To ensure consistency, information is shared to update the situational awareness (SA) [8, 59, 67]. Note that the list of vehicles included in this calculation could be severely constrained to reduce the computation/communication required to plan for all other vehicles.

Hierarchic approaches typically assume the formation of sub-teams that use locally dense communication networks to share information (states, measurements, and plans). Communication between sub-teams would be limited, although it is assumed to be available if necessary to exchange resources. To maintain flexibility, the sub-teams are assumed to be "dynamic" and tasks can be assigned to other sub-teams by the scheduling algorithm. These two approaches reduce the reliance on a central planner system, thereby increasing the rate that the planning system can react to pop-up threats and/or targets of opportunity, increasing the robustness to failure, and ensuring that the control system degrades gracefully. However, it is essential that these decentralized control decisions be well coordinated to maintain good overall performance. A key problem is that achieving tight coordination typically requires that the vehicles exchange large quantities of information about the environment, their current states, and their future intentions. Communication on this scale will not always be possible and it also increases the visibility of the vehicles to threats.

Constraining the communication limits situational awareness, which raises two key issues: first, that decisions must be made based on incomplete information and second, that information may be inconsistent across the fleet, potentially leading to a less cooperative behavior. Thus, one of the primary challenges is that these high-level algorithms must be modified to make them much less reliant on having "perfect, global" situational awareness while still obtaining reasonable performance. For example, a UAV may be uncertain of the distant terrain but able to plan anyway, since another UAV has greater awareness of that region and will be responsible for tasks within it. While it is intuitive that such a scheme could perform very well with limited communication and global awareness, the exact nature of the resulting performance degradation is not well understood. This chapter will investigate this question and tackle the underlying problem of algorithmically identifying the relative significance of information.

Another challenging problem of the decentralized planning is dealing with uncertainty in the vehicles' SA, and it is made even harder when each vehicle has limited knowledge of the SA of the other vehicles in the team. This uncertainty can be reduced to some extent by communicating to share information. Important questions here are to determine which vehicles to communicate with, what data to exchange, and how to balance the effort between communicating input data (SA) or output data (control plans). These questions are driven by the conjecture, based on observations, that much of the information that could be exchanged does result in small changes in the control solution, but does not significantly impact the actual performance. The goal is to avoid this type of inefficiency and focus on only exchanging the data that will have the largest impact on the performance of the closed-loop system. Ref. [34]

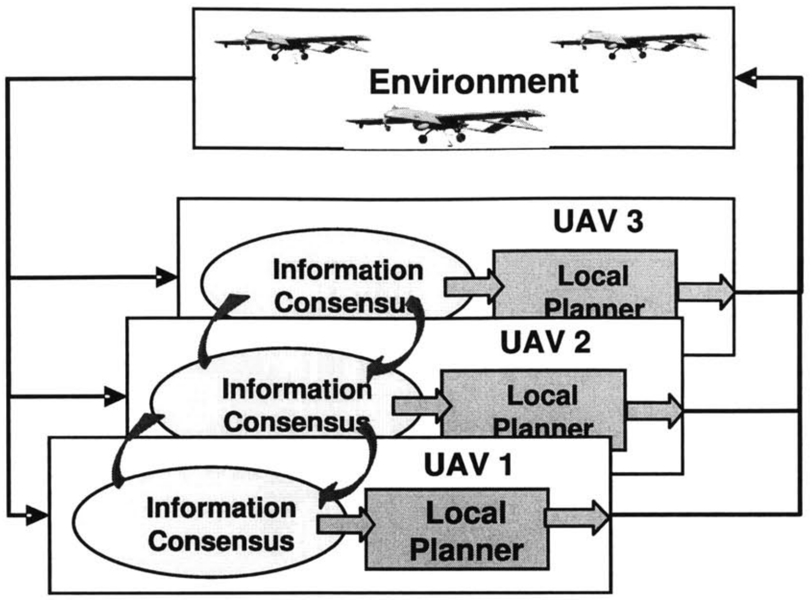


Figure 3.1: Implicit coordination approach using information consensus followed by independent planning.

investigates a similar reduction in the information flow using *channel filters* for *decentralized estimation*. Our problem is very similar, but based on the observation above, a more **control-centric** view of the information exchange must be developed to establish what information will have the largest impact on the closed-loop performance.

3.2 Implicit Coordination Algorithm

This section discusses the implicit coordination method and points out some of its shortcomings. A new methodology is further developed to overcome these shortcomings.

The idea of implicit coordination is to replicate the centralized assignment in each UAV [23]. In this method, each UAV plans for all the UAVs in its team based on its own information and the map of the environment. It then implements its own plan. The premise is that UAVs have the same information and use the same algorithms and objectives to plan. As a result, the plans are the same and similar to the case of

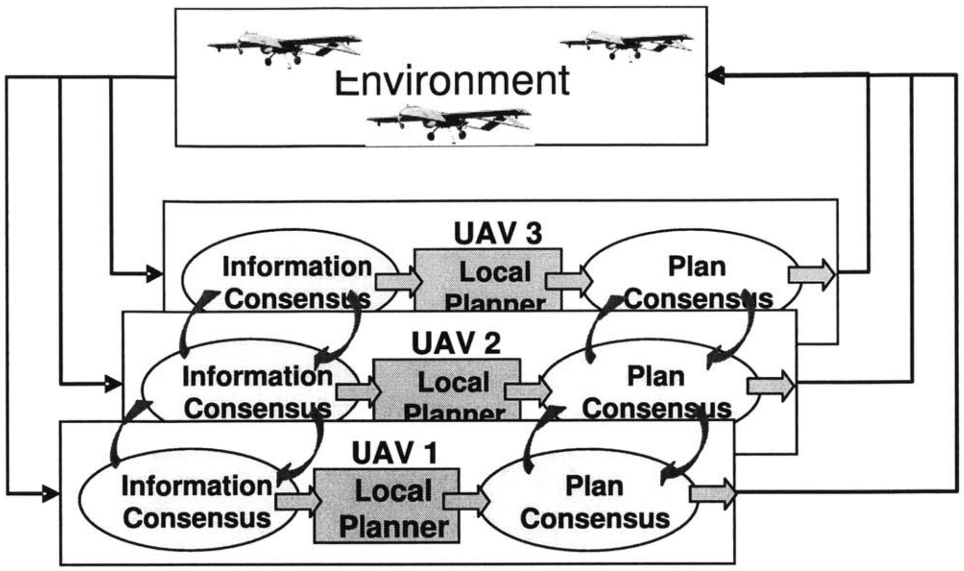


Figure 3.2: Robust Decentralized Task Assignment algorithm that adds an additional round of communication during the plan consensus phase.

the centralized planning. Hence, each UAV can argue that it has the optimal, feasible plan for itself and this plan is consistent with the other UAVs. With these assumptions, one can assume that there will be no conflicts between the plans executed. In reality, however, reaching consensus and having exact information and a consistent map of the environment is not always possible. The environment can change rapidly, and UAVs update their map and information set, which easily causes the mismatch in the information. UAVs must communicate in order to keep the information consistent, but relying on a perfect high bandwidth communication structure makes the implicit coordination method very fragile. Examples in Section 3.4.1 demonstrate this fragility. Even with no limit on the amount of data that could be communicated between UAVs, the system could still fail as a UAV loses its communication with the team. The lack of robustness in the implicit coordination comes from the assumption of consistent information. In order to resolve this shortcoming, an algorithm has to produce consistent plans without the need for perfect consistency of information. In the next section, the implicit coordination is modified to remove this constraint and produce a robust decentralized planning algorithm for UAVs with imperfect communication structure.

3.3 Robust Decentralized Task Assignment

In the implicit coordination method (Figure 3.1), each UAV assumes that once it generates the plan, it is consistent with the other UAVs and therefore it is executed. If the plans are not consistent, then there could be conflicts and the overall plan might be infeasible. Of course, further communication of the information can be performed to develop consensus across the UAV fleet. However, with the sensitivity of the planning process to the input data, this process can take a large number of iterations and still does not guarantee reaching a feasible plan. To avoid the conflicting cases, the UAVs need to communicate their plans and resolve any possible infeasibilities. This can be interpreted as adding a "feedback loop" to the planning phase (Figure 3.2). By a similar analogy, the implicit coordination is essentially an "open-loop" control system that can be strongly influenced by exogenous disturbances. As with standard systems, closing a feedback loop can help improve the overall performance and robustness.

The robust decentralized task assignment (RDTA) algorithm addresses this issue by dividing the planning into two phases. The first phase is similar to the implicit coordination method - each UAV communicates to other UAVs to reach a degree of consensus. In the second phase, each UAV solves the assignment problem for all of the UAVs, as is done in the centralized assignment. But instead of generating one single optimal plan for itself, it generates a set of good (including the optimal) plans. Each UAV then communicates its set of plans to other UAVs. After receiving the plans from other UAVs, each UAV has a set of plans for all of the UAVs in the fleet, which can be used to generate the best feasible plan by solving the task assignment again. The key difference here is that the set of information that forms the basis of the final planning is the communicated set of good plans. Therefore, all of the UAVs have the same set of information and hence if they execute the same task assignment algorithms (same criteria and objectives), they would all generate consistent plans for the fleet. The following describes each phase of the RDTA in more detail.

3.3.1 Algorithm Overview

The RDTA algorithm has two major phases: (1) Information Update and (2) Planning.

Phase 1: Information Update (Reaching Consensus)

In phase 1 of the algorithm, the UAVs communicate with each other to improve the consistency of the information. A simple consensus algorithm is used in this phase of the algorithm to reach the required degree of consistency. If $I_i(t)$ is the information of UAV$_i$ at time t, then the linear discrete form of consensus filter can be written as:

$$I_i(t+1) = I_i(t) + \sum_{j=1}^{N_v} \sigma_{ij} G_{ij}(t)(I_j(t) - I_i(t)) \tag{3.1}$$

where $G(t)$ represents the communication network and G_{ij} is 1 if there is a direct communication link from UAV$_i$ to UAV$_j$ and zero otherwise. σ_{ij}'s are positive constants that represent the relative confidence of UAV$_i$ to UAV$_j$ about their information.

Reaching consensus, however, is not always possible due to communication limits, noise in the communications, and possibly slow rates of convergence in a dynamic environment. Thus, it is likely that the second (planning) phase will have to be executed with a limited degree of consistency in the SA. This phase is when the UAVs calculate and implement their task plans. In the implicit coordination approach, this is done by replicating the centralized assignment algorithm on each UAV. Given full consensus on the SA, and using exactly the same algorithm on each UAV, this method would create similar and non-conflicting plans for each vehicle. However, any differences in the SA (for example, if the consensus algorithm has not yet converged), could lead to a plan with conflicts. RDTA has a modified second phase that eliminates these possible conflicts. Phase 2 of RDTA itself has two stages, as outlined in the following.

Phase 2.1: (Generating the Set of Best Plans):

The UAVs use their updated information to generate a set of ρ candidate plans. The petal algorithm [4, 10] is a good choice for the proposed algorithm because it performs an optimization based on pre-generated feasible plans and is modified to be used here.

In the petal algorithm, first, each UAV$_i$ creates a list of all un-ordered feasible task combinations for every UAV$_j$, ($P_{ij} = \{p_{ij}^1, \ldots, p_{ij}^{k_j}\}$ and $P_i = \bigcup_{j=1}^{N_v} P_{ij}$). Next, the length of the shortest path made up of straight line segments between the waypoints and around obstacles is calculated for all possible order-of-arrival permutations of each combination (these permutations are referred to as petals). The construction of these paths can be performed extremely rapidly using graph search techniques [10]. The time of visit for each waypoint w, t_w is estimated by dividing the length of the shortest path to that waypoint by the UAV's maximum speed. The time-discounted score is consequently calculated for each waypoint in each petal ($S_w(t) = \lambda^t\ S_w$, $\lambda < 1$ is the discount coefficient). The time-discounted score for each petal is the sum of the discounted score of its waypoints.

The algorithm produces a set of N_M petals, denoted by $\mathbf{P}_i$, and a vector of size N_M, denoted by $\mathbf{S}_i$, whose p^{th} elements, taken together, fully describe one permutation of waypoints for one UAV. The P_i^{pw} entry of the set $\mathbf{P}_i$ is 1 if waypoint w is visited by petal p and 0 if not. Element S_i^p of the vector $\mathbf{S}_i$ is the time-discounted score of the petal p. This procedure is described in detail in Ref. [10].

Once these sets are created, a mathematical method is developed for allocating the waypoints to each UAV based on these scores and other constraints. The base of the task allocation problem is formulated as a Multidimensional Multiple-Choice Knapsack Problem (MMKP) [57]. The "knapsack" in this case is the complete mission plan. The vector corresponding to each of the N_M petals makes up the multi-dimensional weight. The "multiple-choice" comes from choosing which petal to assign to each of the N_v different UAVs (sets). The objective is to assign one petal (element) to each vehicle (set) that is combined into the mission plan (knapsack), such that the score of the mission (knapsack) is maximized and the waypoints visited (weight) meet the constraints for each of the N_w dimensions. The problem is

$$\begin{aligned} \max_{x} \ J_i &= \sum_{p \in \mathcal{M}_i} S_i^p x_i^p \\ \text{subject to} \quad & \forall w \in \mathcal{W} : \sum_{p \in \mathcal{M}} P_i^{wp} x_i^p \leq 1 \\ & \forall v \in \mathcal{V} : \sum_{p \in \mathcal{M}_{iv}} x_i^p = 1 \end{aligned} \tag{3.2}$$

where $\mathcal{M}_i = \{1, \ldots, N_M\}$ and $\mathcal{M}_{iv} \subseteq \mathcal{M}$ are the indexes of petals in P_i and P_{ij} respectively and $\mathcal{W} = \{1, \ldots, N_w\}$ is the list of waypoints. The binary decision variable x_i^p equals 1 if petal p is selected, and 0 otherwise. The objective in this problem formulation maximizes the sum of the scores to perform each selected petal. The first constraint enforces that each waypoint w is visited at most once. The second constraint prevents more than one petal being assigned to each vehicle. Solving this MMKP selects a petal for each UAV$_j$, ($p_{ij}^{k_1^*}$), which is essentially a solution to the task allocation problem. Note that this process is repeated in each UAV$_i$ and therefore each UAV$_i$ generates plans for every single UAV in the team.

The selected petal for UAV$_i$ will be the first candidate plan for UAV$_i$ and is added to the set, P_i^*, $P_i^* = \{p_{ii}^{k_1^*}\}$. After updating the set of feasible petals for each UAV$_i$,

$$P_{ij} = \begin{cases} P_{ij} & \text{if } (i \neq j) \\ P_{ij} - \{p_{ij}^{k_1^*}\} & \text{if } (i = j) \end{cases} \qquad \forall j \tag{3.3}$$

The optimization in 3.2 and update in 3.3 are repeated ρ times to create a list of candidate plans for UAV$_i$, $P_i^* = \{p_{ii}^{k_1^*}, \ldots, p_{ii}^{k_\rho^*}\}$ and the scores associated with them, $S_i^* = \{S_i^{k_1^*}, \ldots, S_i^{k_q^*}\}$. Each UAV$_i$ then communicates this set of ρ candidate plans, P_i^* and the scores associated with these petals to all other UAVs.

Phase 2.2: (Generating the Final Feasible Plans):

Each UAV$_i$ uses the set of candidate plans from all the UAVs, P_j^*'s and implements the petal p_{ii}^* that results from the optimization

$$\max_{x_{ij}^k} J_i = \sum_{j=1}^{N_v} \sum_{k \in \mathcal{M}_j^*} x_{ij}^k S_j^k \tag{3.4}$$

$$\text{subject to} \quad \sum_{k \in \mathcal{M}_j^*} x_{ij}^k = 1; \forall j \tag{3.5}$$

$$\sum_{j=1}^{N_v} \sum_{k \in \mathcal{M}_j^*} x_{ij}^k p_j^k \leq \mathbf{1} \tag{3.6}$$

where $\mathcal{M}_j^* = \{k_1^*, \ldots, k_\rho^*\}$. Algorithm 1 presents the summary of the Robust Decentralize Task Assignment algorithm. Note that this algorithm with $\rho = 1$ is essentially the implicit coordination algorithm.

3.4 Analyzing the RDTA Algorithm

The RDTA algorithm has several important features that improve the performance when compared to the implicit coordination algorithm as the benchmark. First, phase 2.2 is done based on consistent, pre-generated, plans, which ensures that there are no conflicts in the final plans selected independently. Second, since each UAV will execute a plan from the candidate set that it created, it is guaranteed to be feasible for that vehicle. Furthermore, communicating a small set of candidate plans helps overcome any residual disparity in the information at the end of phase 1 (consensus). This improves the possibility of finding a group solution that is close to the optimal, while avoiding the communication overload that would be associated with exchanging all possible candidates. This section analyzes the important aspects of the RDTA algorithm and presents simulation results to highlight the effect of various design parameters on the performance.

Algorithm 1 – RDTA Algorithm

```
for i = 1 : Number of UAVs ($N_v$) do
  while (Not reached the desired degree of Consensus) do
    Send the current SA to neighbors,
    Receive other UAVs' SA,
    Update the SA using the consensus algorithm,
  end while
  Set $P_i = \{\}$,
  for j = 1 : Number of UAVs ($N_v$) do
    Enumerate all feasible plans for UAV$_j$, $P_{ij}$,
    Calculate the time-discounted score of each plan, $S_{ij}$,
    $P_i = P_i \bigcup P_{ij}$,
  end for
  Set $P_i^* = \{\}$,
  for k = 1 : $\rho$ do
    Solve the optimization in Eq. 3.2,
    Find the optimal plan for all the UAVs, $p_{ij}^*$,
    Add $p_{ii}^*$ to the list of candidate plans, $P_i^* = P_i^* \bigcup \{p_{ii}^*\}$,
    $P_i = P_i - \{p_{ii}^*\}$,
  end for
  Send the set of candidate plans, $P_i^*$ to all other UAVs,
  Receive other UAVs' candidate plans,
  Solve the optimization in Eq. 3.6,
  Find the optimal plan for all the UAVs,
  Implement the plan for UAV$_i$, $p_i^*$.
end for
```

3.4.1 Advantages of RDTA Over the Implicit Coordination

The first set of simulations were designed to demonstrate the shortcomings of the implicit coordination and advantages of RDTA over the implicit coordination. A simple example of 3 UAVs with 8 targets where each UAV is capable of visiting at most 2 targets is used. For simplicity, all targets are assigned the same score. It is also assumed that all the UAVs are capable of visiting all the targets. In the first run, the implicit coordination method is implemented. In this case, all the UAVs have consistent information and the result is the same as the centralized assignment (Figure 3.3).

In the second run, the same algorithm is used, but the data is perturbed so that the UAVs have inconsistent information to develop their plans. There are different attributes of targets that can be altered such as their type, score, and position. In this case only the positions of the targets are changed. A random number is added to the position of targets for each UAV. The random number is generated with a uniform distribution in the interval of $[-30\%, +30\%]$ of the value of the position. Each UAV then has its own version of the information, which is inconsistent with other UAVs.

Figure 3.4 demonstrates the result for a case when the UAVs have conflicting assignments. Note that conflict here is defined as an assignment in which two or more UAVs are assigned to the same target. The same problem is also solved using the RDTA algorithm introduced in Section 3.3. Here, $\rho_i = 2, \forall i \in \{1, 2, 3\}$. The result is presented in Figure 3.5, which shows that using the RDTA for this example and only communicating two petals per UAV can eliminate the conflicts that appeared in the implicit coordination solution.

3.4.2 Simulation Setup

A simple scenario of 5 UAVs and 10 targets is used as the baseline for all simulations presented in the following sections. The simulation results are typically generated from 100 Monte Carlo simulations, with the position of the targets, (x,y) created randomly (these random numbers are uniformly distributed in $[200\ \ 400] \times [200\ \ 400]$). Figure 3.6 shows a sample scenario with the optimal assignment. When comparing different algorithms, the exact same problems are solved in all cases using the same seed for the random number generators. To make the information inconsistent for different UAVs, a random number is added to the position of each waypoint. These random numbers are different for each target and each UAV. These random numbers are generated with a uniform distribution in the interval of $[-30\%, +30\%]$ of the value of the position. Each UAV then has its own version of the information, which is inconsistent with other UAVs. If $I_i = [I_i(1), \ldots, I_i(m)]^T$ is the information vector (here position of waypoints) associated with UAV$_i$, then the level of inconsistency in

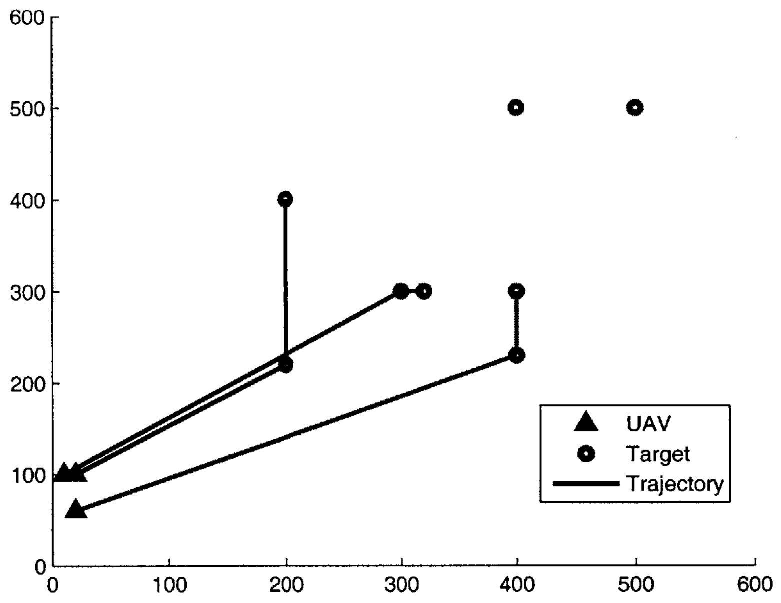


Figure 3.3: Optimal Plan resulting from consistent information.

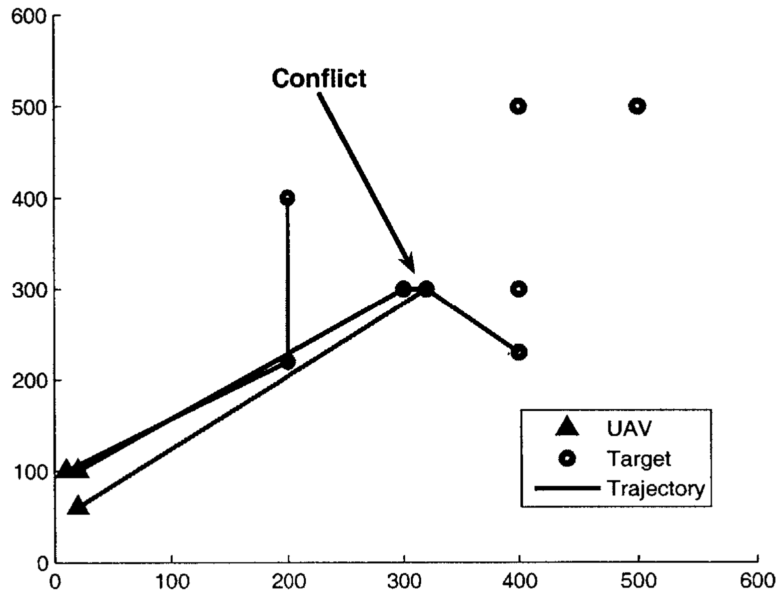


Figure 3.4: Plan with conflicts resulting from inconsistent information.

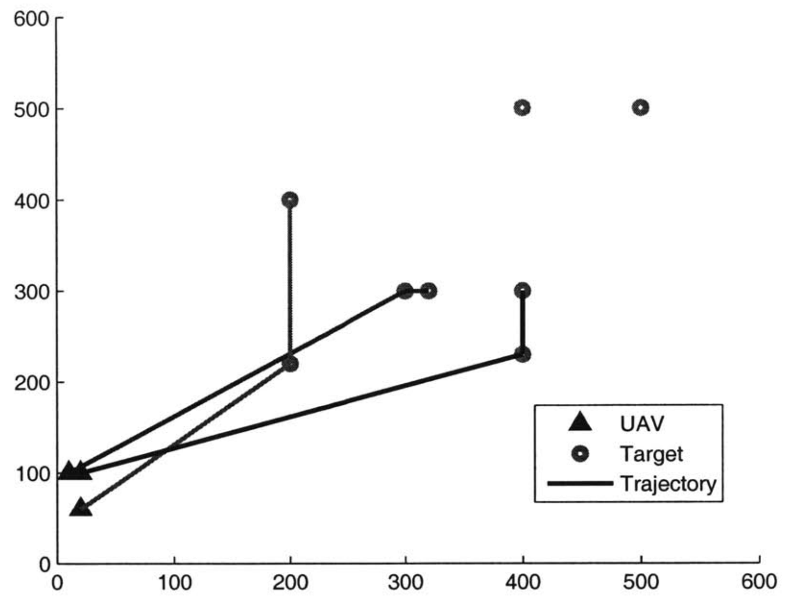


Figure 3.5: Plan with inconsistent information. No conflict result from RDTA.

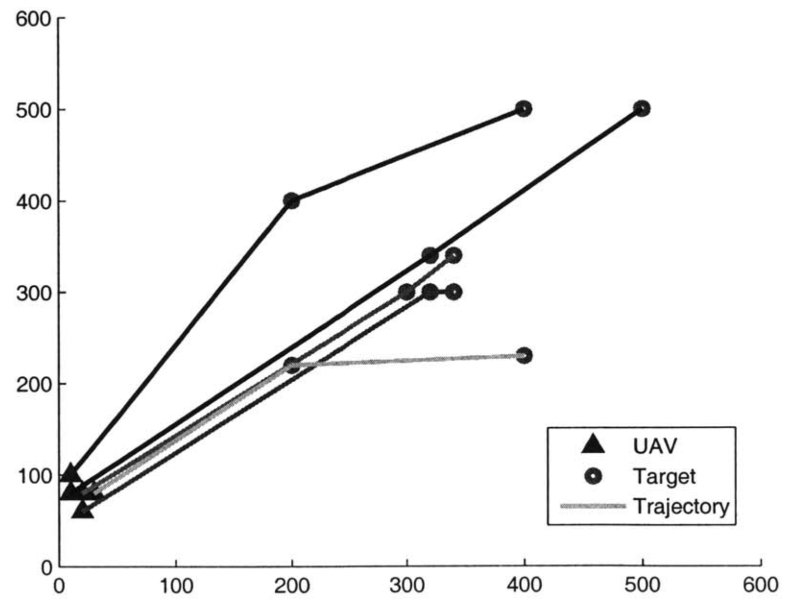


Figure 3.6: Optimal plan for the scenario of 5 UAVs and 10 targets.

the information is quantified using the inverse Sharpe ratio, ζ defined as,

$$\zeta = \sum_{k=1}^{m} \frac{\delta_I(k)}{\mu_I(k)} \tag{3.7}$$

where $\mu_I = [\mu_I(1), \ldots, \mu_I(m)]^T$ and $\delta_I = [\delta_I(1), \ldots, \delta_I(m)]^T$ are the mean and standard deviation of information vectors respectively. Communication between UAVs with a consensus algorithm decreases the inconsistency in the information and consequently decreases ζ. The communication noise, new measurements, and all the exogenous disturbances tend to increase ζ. A large ζ corresponds to a significant inconsistency in SA between the UAVs, which will impact the plans selected by each UAV. In the case of implicit coordination, where there is no feedback in the planning phase, this could result in conflicting plans for the team.

For most simulations, the petals are of size 2, which means that the UAV plans have less than or equal to two waypoints. In some cases this number is increased to 3. In a real-time implementation, these plans would be redesigned over time to ensure that all targets/tasks are completed. The communication networks for all simulations are generated randomly. These networks are all strongly connected networks, where there is a communication path from every UAV to any other UAV.

3.4.3 Effect of ρ on the Conflicts

To better show the advantages of robust decentralized assignment over the implicit coordination, Monte Carlo simulations described in Section 3.4.2 are used. Figure 3.7 shows the average number of conflicts versus the size of the communicated petal set, ρ. For the case $\rho = 1$, on average 2.1 of the UAVs have conflicts with other UAVs. The number of conflicts decreases as ρ increases and drops to zero at $\rho = 7$. To understand the result of these conflicts on the overall performance of the plan, Figure 3.8 shows the performance versus the size of the communicated petal set, ρ. Increasing ρ can have a large impact on the performance – it is almost doubled for $\rho = 7$ compared to $\rho = 1$, which is the implicit coordination algorithm.

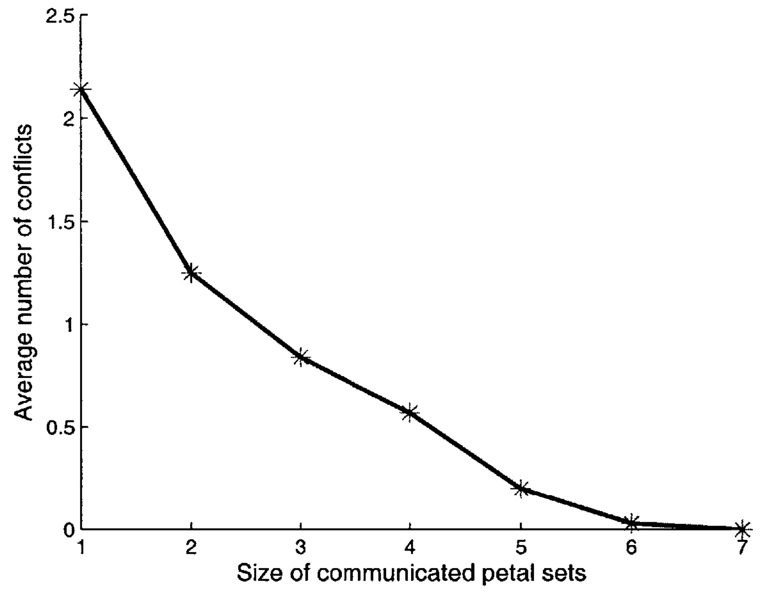


Figure 3.7: Effect of communication in the planning phase (closing the loop in the planning phase) on the reduction of conflicts.

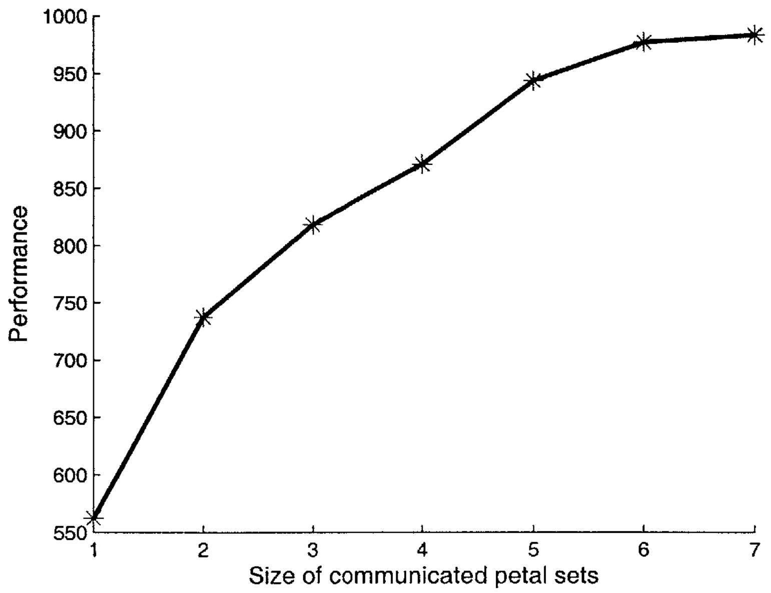


Figure 3.8: Effect of communication in the planning phase (closing the loop in the planning phase) on the performance.

3.4.4 Effect of Communication on the Two Phases of the Algorithm

The first analysis considers the effect of communication during the two phases of the algorithm on the performance. In the first phase, the consensus algorithm is used to improve the consistency of the information across the UAV team. This information is then used to produce a robust feasible plan for the fleet. To see the effect of communication in each phase, two parameters are varied in the simulations. In the first (consensus) phase, the convergence of the information is directly related to the number of iterations. The amount of communication is also a linear function of the number of iterations. Therefore, by changing the number of iterations, the amount of information communicated in the first stage can be directly controlled.

The second important parameter is the size of the candidate plan set that is communicated in phase 1 of the planning phase, ρ. The communication in this phase is also a linear function of this parameter and thus can be controlled as well. Figure 3.9 shows the result of the simulations introduced in Section 3.4.2. The result shown is the average of 100 Monte Carlo simulations in which the positions of the targets were chosen randomly. Also the communication networks for these simulations are randomly-generated, strongly connected networks. For each scenario, the number of iterations in consensus phase was changed from 0 to 7 and for each case the RDTA algorithm was applied with different values of $\rho = 1, \ldots, 5$. In this figure, the x-axis shows the number of iterations, the y-axis is the size of the candidate petal set (communicated petal set), ρ, and the z-axis is the performance of the algorithm (total score of the plan). Note that the performance values are normalized to be between 0 and 1, where performance of 0 means the total accumulated score by UAVs is zero and the performance of 1 is the performance of the optimal solution. The plot clearly shows that the performance of the algorithm increases as both parameters (number of iterations and the size of candidate petal set) increase.

To better show the relationship between performance and the communication in each phase, Figure 3.9 is transformed into Figure 3.10, in which the x- and y-axes are

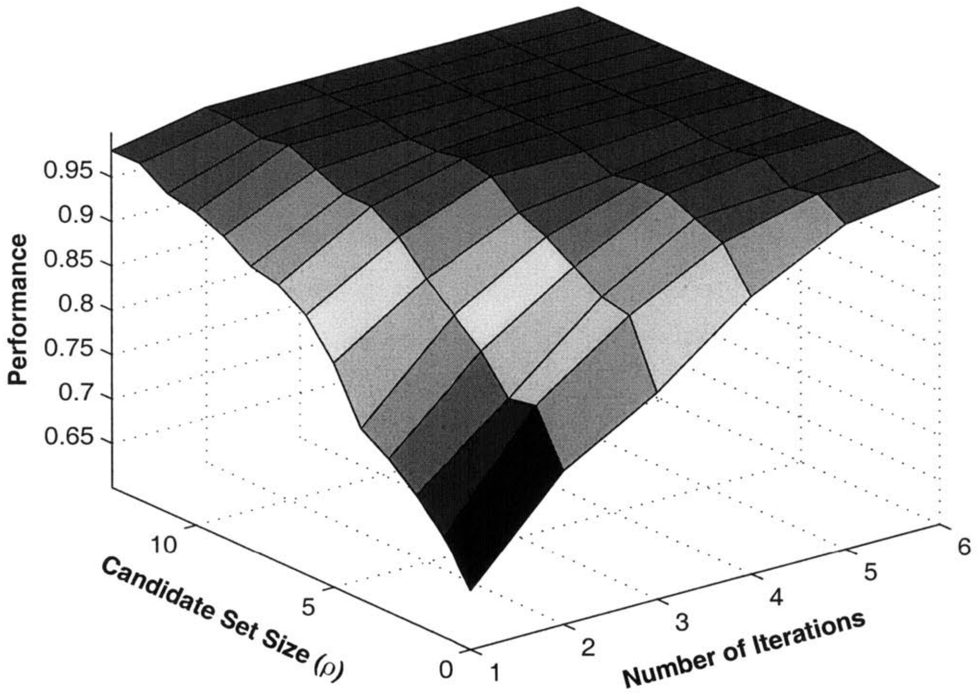


Figure 3.9: Effect of two important parameters (iterations for consensus and size of candidate petal set) on the performance.

the communication in the information (consensus) and planning phases, respectively. Communication was measured using the following rules. In the information phase, in each iteration, each UAV has to communicate its information about the position of all targets to other UAVs. Assuming that the position of each target has two dimensions, then two words of information must be communicated for each target. There are 10 targets in this example and therefore the total communication (the number of words that all the UAVs communicate in all iterations) is

$$C_c = 20 \cdot r \cdot l \tag{3.8}$$

where $20 = 2 \times 10$ is the number of words that each UAV communicates in each

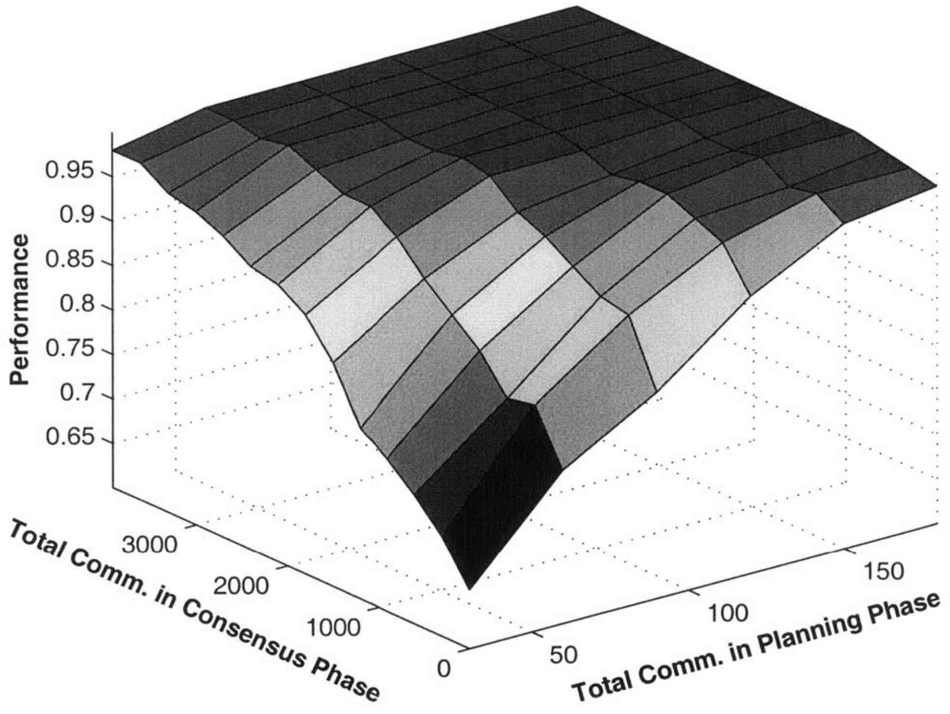


Figure 3.10: Effect on performance of communication in the two phases of algorithm (consensus and loop closure in the planning).

iteration, r is the number of iterations in the consensus phase and l is the total number of links in the network.

In the planning phase, the petals and the score of each petal must be communicated. Each petal is a binary vector which can be interpreted to an integer number and transmitted as a word. Hence for each petal, each UAV must communicate a total of two words and therefore, in the planning phase, the total communication is:

$$C_p = 2 \cdot \rho \cdot l \cdot q \tag{3.9}$$

where the 2 is the number of words per candidate plan that UAV$_i$ has to communicate, ρ is the size of the candidate plan sets, l is the number of links in the network, and q is the number of iterations required for all the UAVs to receive the candidate plan

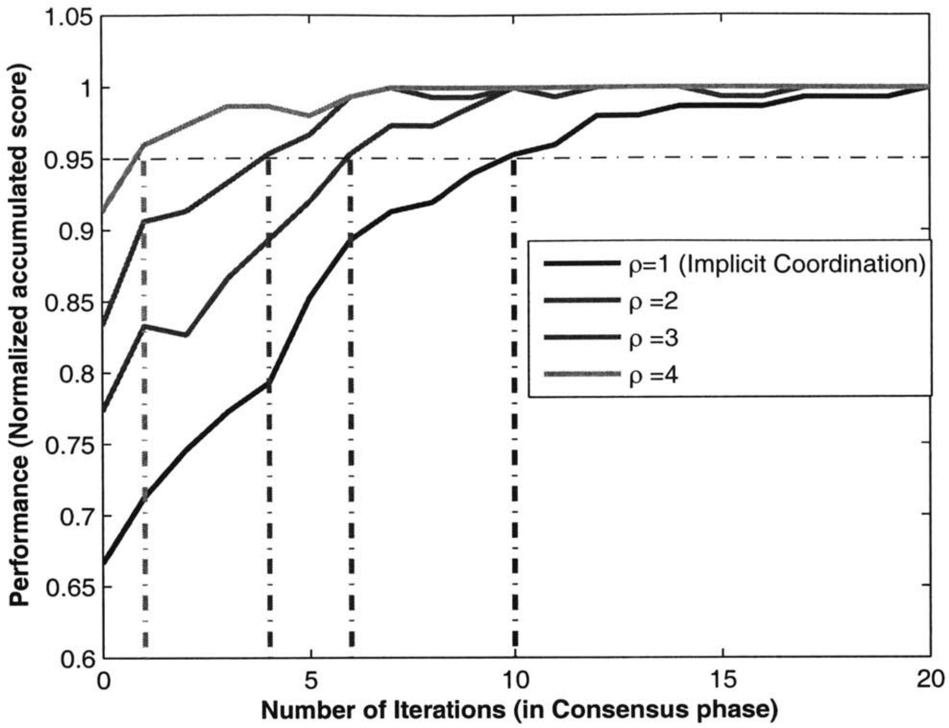


Figure 3.11: Performance versus the number of iterations for different values of ρ.

from any other UAV. Note that when there is no direct communication between two UAVs, these two UAVs communicate their candidate plan sets through other UAVs that act as relays. This is always possible since the networks are assumed to be strongly connected. Therefore, if the shortest path from UAV$_i$ to UAV$_j$ consists of u UAVs, then the total iterations needed for these UAVs to send their plans to each other will be $u + 1$, which is represented by parameter q in Eq. 3.9.

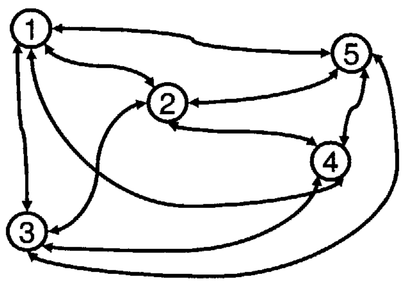
Figure 3.10 shows that increasing the communication in either axis (consensus or planning phase) improves the performance. However, the results also clearly show that communication in the planning phase is more efficient than in the information (consensus) phase in the sense that 200 words of communication in the planning phase have approximately the same effect on performance as 2000 words in consensus phase.

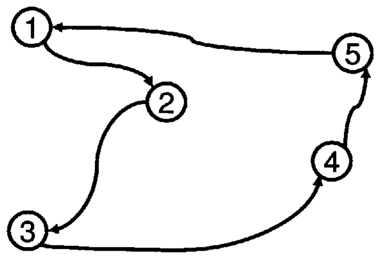
The plot also shows that to maximize the performance, some communication in both phases is needed.

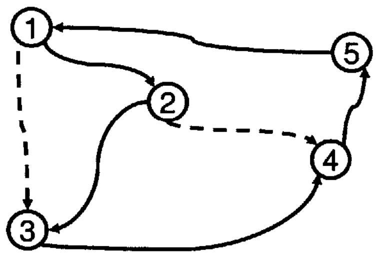
The performance curve versus the number of iterations is shown for four different values of ρ (size of candidate plan set) in Figure 3.11. This can be thought of as four slices of Figure 3.9. In this figure $\rho = 1$ corresponds to the implicit coordination algorithm. The horizontal dashed line corresponds to 95% of the optimal performance, and the vertical dashed lines show the number of iterations needed for each case (different ρ) to reach the 95% limit. Note that for the implicit coordination case ($\rho = 1$), on average 10 iterations in consensus phase are needed to reach this performance limit. The required number of iteration decreases to 6, 4 and 1 by increasing ρ to 2, 3 and 4 respectively. Note that only by one iteration in the consensus phase and using RDTA with $\rho = 4$, can the 95% limit be reached. This figure clearly shows that the sufficient consistency level is significantly lower for RDTA compared to the implicit coordination algorithm.

3.4.5 Effect of Communication Structure on the Performance

The second analysis extends the previous results to communication architectures with different network topologies. The effect of the sparsity of the communication network on the RDTA performance is analyzed using the same simulation setup as in Section 3.4.4. All the simulations considered here so far, assume that the communication network is strongly connected, which means there is a directed path from every UAV to every other UAV. A strongly connected graph with the minimum number of links is first created, and then additional links are added randomly to generate communication networks with 5, 10 and 15 links. Figure 3.12 presents different network topologies for a 5 node (UAVs) network. Fig. 3.12(a) shows a fully connected network in which each UAV directly communicates with every other UAV in the team. Fig. 3.12(b) shows a strongly connected network with the minimum number of links (here 5). Fig. 3.12(c) shows a randomly generated strongly connected network with 7 links and Fig. 3.12(d) shows a network that is not strongly connected (weakly connected).


(a) A fully connected Network


(b) A strongly connected network with the minimum number of links (5 links)


(c) A random strongly connected network with 7 links

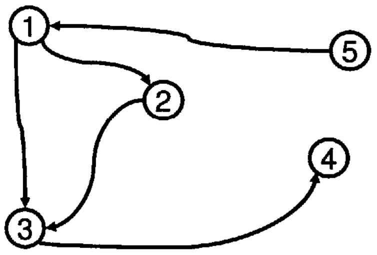

(d) A weakly connected network

Figure 3.12: Demonstration of the different network connection topologies.

The results of these simulations are presented in Figures 3.13–3.15. The total communication in each phase (x- and y-axes) is calculated using Eqs. 3.8, 3.9. Comparing the performance for these three network topologies, it is clear that much more communication is needed for a sparse network (Fig. 3.13) to reach the same level of performance. It is also shown that, to reach a certain level of performance, communication in the planning phase is more effective than communication in the information consensus phase, thereby making the advantages of RDTA (and in particular communication in the planning phase) very apparent for sparse communication networks. In the cases of limited communication bandwidth and/or constraints in communication time, for sparse networks, the inefficiency of communicating raw information significantly degrades the performance and communicating the processed data (candidate

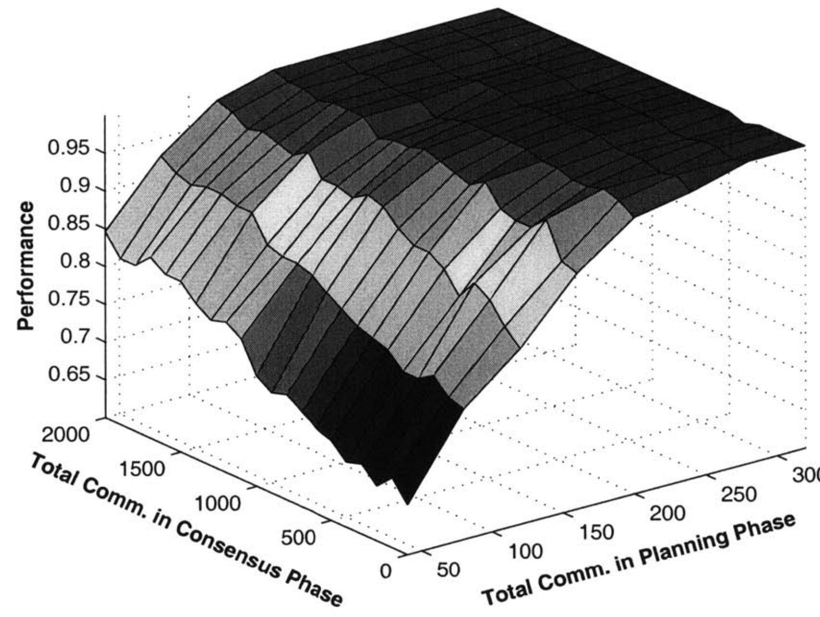


Figure 3.13: Strongly connected communication network with 5 unidirectional links.

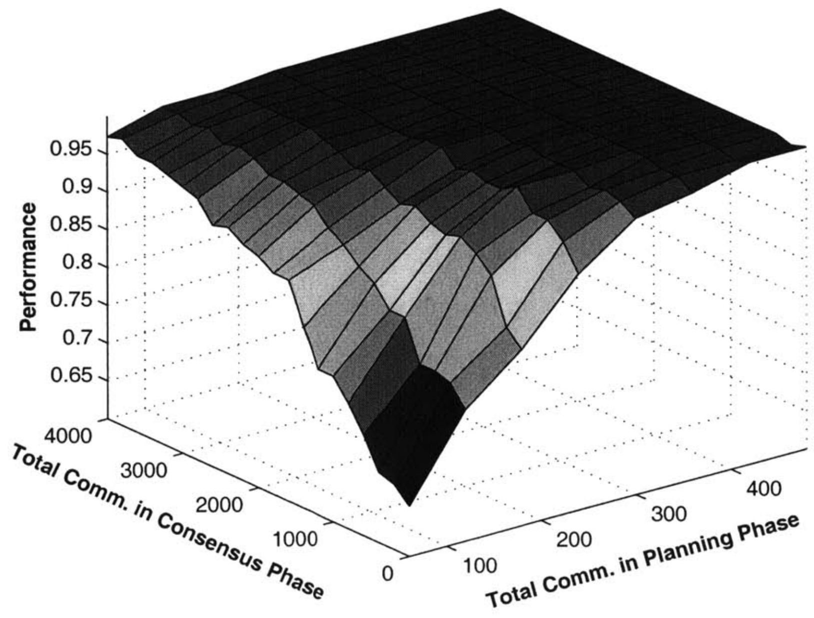


Figure 3.14: Strongly connected communication network with 10 unidirectional links.

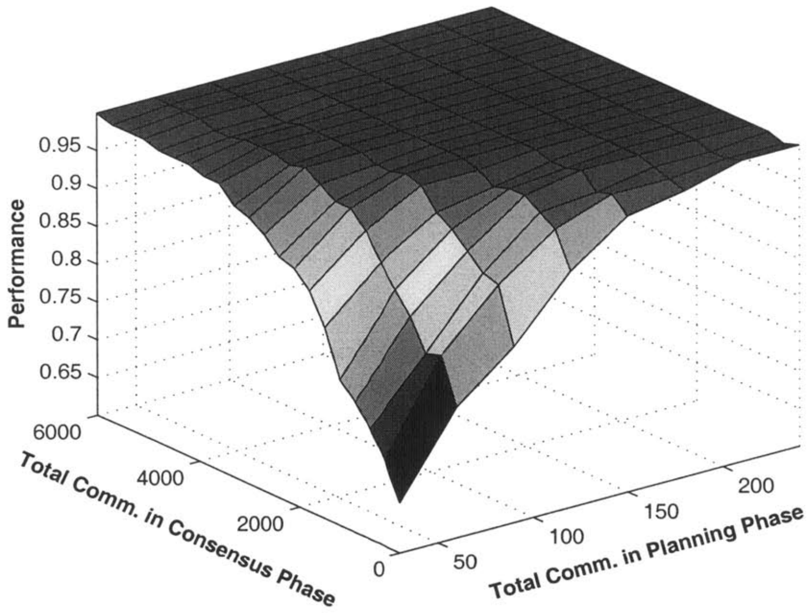


Figure 3.15: Strongly connected communication network with 15 unidirectional links.

plan set) in the planning phase is essential.

To further illustrate the effect of communication on improving consistency, Figure 3.16 presents the inverse Sharpe ratio, ζ, defined in (3.7) for these three communication topologies. The x-axis is the number of iterations in the consensus phase, or the number of times that the UAVs communicate with each other, and the y-axis is the inverse Sharpe ratio (a measure of inconsistency). The graphs clearly show that for sparse networks, the convergence rate for the information consensus is very low, and thus, with the same number of iterations (time spent in communication), the UAVs in a sparse network will have more inconsistencies in the information. This increased inconsistency in the data directly leads to more conflicts in the decentralized assignment, thereby decreasing the team's performance. In this case, adding the communication during the planning phase has a large impact on the performance by significantly reducing the effect of the residual inconsistencies in the information.

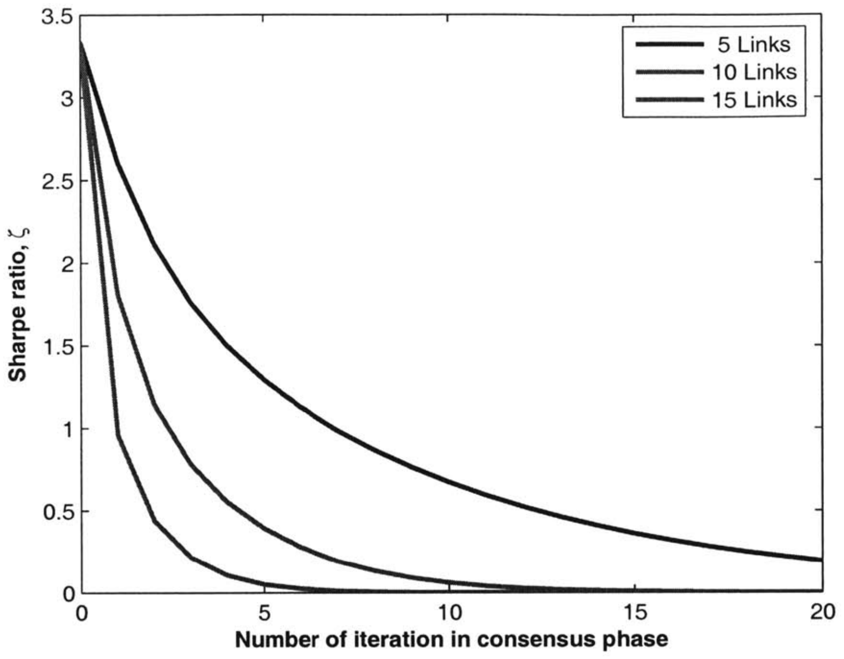


Figure 3.16: Comparing convergence rate of information in the consensus phase for 3 different network topologies.

3.4.6 Effect of Algorithm Choice in the First Stage of Planning Phase

This analysis addresses the issue of the impact of the algorithm used in the first stage of the planning. Section 3.3.1 discusses an approach that applies a centralized scheme to create a set of *"optimal, coordinated plans"* for the set of candidate plans. However, this level of sophistication (and its associated computational effort) may not be necessary to achieve good performance. The following investigates four alternatives for creating the candidate petals by varying the extent to which the plans account for future time and other vehicles in the team.

A) **Greedy in space and in time:** This method ignores all other UAVs in its task selection (uncoordinated). It is also greedy in time since it only assigns the tasks one at a time.

B) **Greedy in space (uncoordinated petal):** This method is similar to the petal algorithm, since it bundles the tasks and plans using these petals. However, this algorithm only generates petals for the planning UAV, selecting the petals with the highest score without any regard to the other UAVs (uncoordinated).

C) **Greedy in time, non-greedy in space (coordinated):** In this algorithm tasks are selected one at a time and added to the path (greedy in time), but the choices of all UAVs are considered in selecting these tasks. Therefore, each UAV creates its petal with consideration of what tasks the other UAVs might be doing (coordination).

D) **Petal (coordinated):** This algorithm is the one given in Section 3.3.1 from the original RDTA algorithm.

The simulation from Section 3.4.2 is used to compare these four algorithms with the goal of determining the trade-off in performance degradation versus computational improvement. Similar to the previous setup, the inconsistency is in the position of targets. Figures 3.17–3.19 compare the results of the four algorithms. To compare the effect of greediness in time, the algorithms D (petal) and C (greedy in time, non-greedy in space) are compared in Figure 3.17. The z-axis in this figure is the difference in the performance between algorithms D and C $(D-C)$. The performances of these two algorithms are very close, and greediness in time does not appear to have a significant effect on the overall performance.

In Figure 3.18, the petal algorithm, D, is compared to the greedy-in-space algorithm, B, $(D-B)$. The difference in the performance is always positive, which clearly shows the advantage of algorithm D over B and highlights the disadvantage of poor team coordination. By accounting for the preferences of the other UAVs in selecting the candidate petals, in algorithm D each UAV can increase the probability of finding a non-conflicting plan with good performance in the second stage of the planning phase. These results show that if the coordination is ignored in the first stage of planning, it cannot be recovered in the second stage, and although the final plan is non-conflicting, it might not assign some UAVs to any targets to avoid

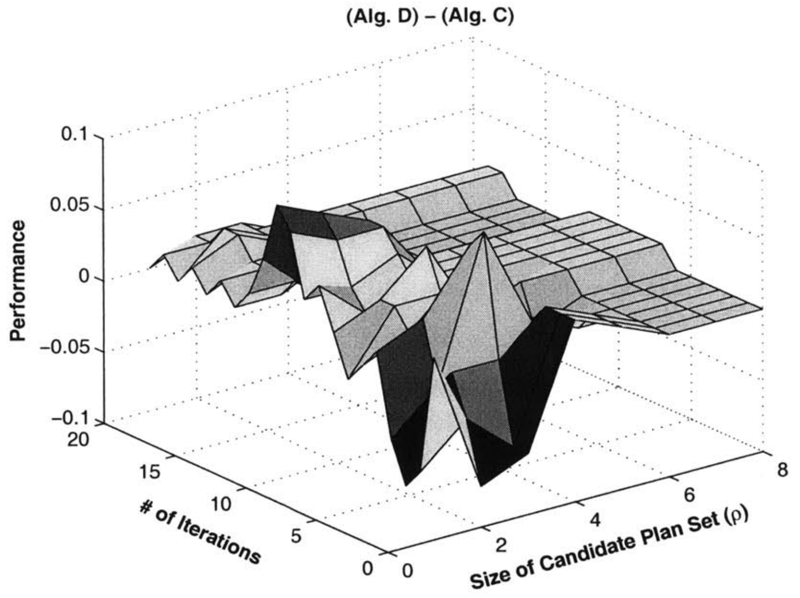


Figure 3.17: Compare algorithms D and C to show effect of greediness in time. z-axis is difference in performance $D - C$.

conflicts. Figure 3.19 compares algorithms C and A, $(C - A)$, and confirms that coordination with teammates is a bigger driver of performance than planning further into the future.

Table 3.1: Comparing the results of different algorithms

Algorithm	A	B	C	D
Avg. Performance (normalized)	0.36	0.42	0.99	1.0
Avg. Computation Time (normalized)	0.20	0.22	0.93	1.0

Table 3.1 summarizes the performance and computation time of these four algorithms. The numbers here are normalized by the maximum. The results clearly show the advantage of having coordination in the first stage of planning (algorithms C and D). The performances of these two algorithms are significantly better than the uncoordinated algorithms, but at the same time, the computation times for these two algorithms (C and D) are significantly higher than for algorithms A and B. The decision on which algorithm to use will be problem specific. In the cases where longer computation time of the coordinated algorithms is acceptable, C and D would be the

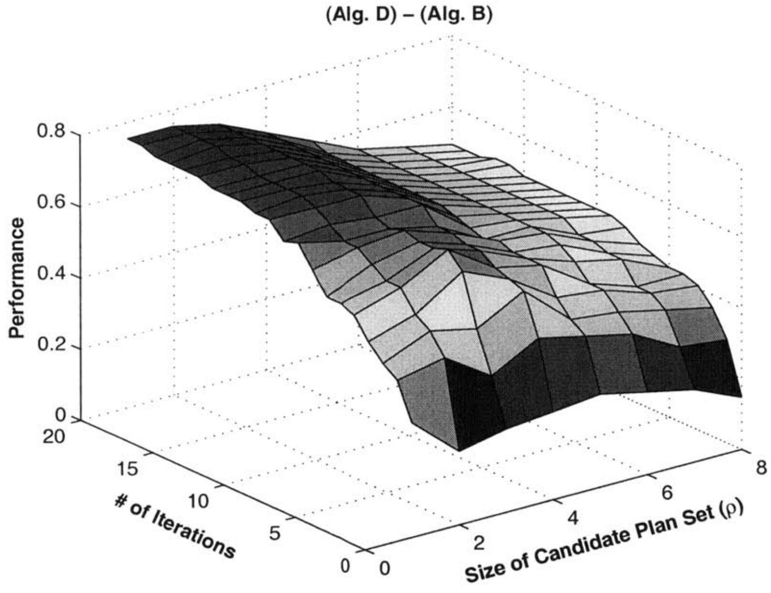


Figure 3.18: Compare algorithms D and B to show effect of coordination. z-axis is difference in performance $D - B$.

better choice. But in a case where the computation time is constrained, algorithms A and B would have to be investigated.

3.5 Performance Improvements

This section presents two extensions to the original RDTA algorithm that are introduced to improve the performance of the algorithm. These modifications are in the petal selection in the first stage of the planning phase. In stage 2 of the planning phase, the UAVs create the final non-conflicting plan based on the candidate plan set. The performance of the final plan strongly depends on the quality of these candidate plans. For instance, if the algorithm cannot create a non-conflicting plan in which all UAVs are assigned to targets, it will avoid conflicts by not assigning one or more of the UAVs to any targets. Although this plan is the optimal plan given the candidate plan set, the performance might be quite poor because it is underutilizing the UAV team.

In the original RDTA algorithm, the selected ρ petals in stage one of the planning

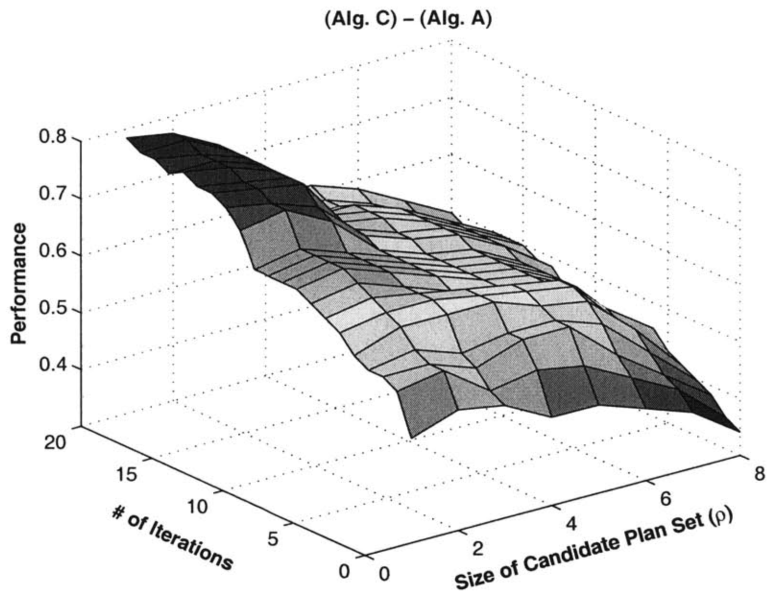


Figure 3.19: Compare algorithms C and A to show effect of coordination. z-axis is difference in performance $C - A$.

phase (phase 2.1) are the best petals for that specific UAV and therefore can be very similar in the sense that they have common targets/waypoints. In the case of inconsistent information between the UAVs, this similarity in the petals reduces the possibility of finding a feasible assignment where all UAVs can be assigned to their maximum number of targets, and therefore has the potential of decreasing the expected performance of the team. Thus the objective is to find an alternative method to choose these petals in the first stage of the planning phase in order to increase the possibility of having a feasible plan (plan with no conflicts) in the second stage. In particular, to reduce the potential of conflicts in the set of candidate plans, the communicated candidate plan sets must include a very diverse set of targets. Two approaches are presented and compared in the following. The first modification (A) is intuitively appealing, but is shown to actually decrease the performance. The second modification (B) improves the performance over the nominal algorithm.

Modification A

In this modification, when a petal $p_{ii}^{k^*}$ is selected and added to petal list P_i^* for UAV$_i$, then all of the remaining petals in P_{ii} that have common waypoints with the selected petal are penalized. Penalizing the score of petals that are similar to the petals that are already added to the candidate plan set (have common targets) improves the chance that the petals with a lower score that are not similar to the selected petals will be selected. This algorithm will avoid creating a candidate plan set that includes a set of petals with high scores, but very similar set of targets. For example, suppose that $p_{ii}^{k^*}$ contains waypoint w, then any petal in the P_{ii} that also contains w gets penalized,

$$\text{If } (w \in p_{ii}^{k^*} \ \& \ w \in p_{ii}^{k}) \ \Rightarrow \ s_{ii}^{k} = s_{ii}^{k} - \delta, \ \forall p_{ii}^{k} \in P_{ii}$$

where δ is the penalty factor. To create the next petal for the candidate petal set, the new sets of petals with adjusted scores (penalized) are used in the optimization. Since the petals that are similar to the ones already selected are penalized, the new selected petals tend to be different to those already selected, which leads to a more diverse final petal set.

Modification B

This approach uses the same penalty technique, but applies it to the petals associated with the planning UAV$_i$ and all other UAVs. For example, consider the optimization by UAV$_i$, which selects a set of petals for all UAVs, $\{p_{i1}^{k^*}, \ldots, p_{ii}^{k^*}, \ldots, p_{iN_v}^{k^*}\}$. In modification A, only petals in P_{ii} that were similar to p_{ii}^{k} were penalized, but in modification B, this penalty is applied to all petals for all UAVs: $\forall j \in 1, \ldots, N_v$

$$\text{If } (w \in p_{ij}^{k^*} \ \& \ w \in p_{ij}^{k}) \ \Rightarrow \ s_{ij}^{k} = s_{ij}^{k} - \delta, \ \forall p_{ij}^{k} \in P_{ij}$$

The same simulation as before is used to compare the impact of these modifications on the performance. In these simulations, however, the effect of consensus is not important and therefore is omitted. Figures 3.20 and 3.21 show the results for two sets of Monte Carlo simulations. These figures show the performance of each algorithm (original, Modifications A and B) for different sizes of the candidate plan set. Figure 3.20 is the result of the base case of 5 UAVs and 10 targets and petal size of 2. Figure 3.21 presents the same results for a scenario of 4 UAVs, 7 targets with a petal size of 3. The results clearly show that in modification B, the performance of the algorithm is improved by increasing the diversity and spanning a wider range of waypoints in the candidate plan sets. The advantage of this modification over the original RDTA is more obvious in Figure 3.21.

3.6 Hardware-In-the-Loop Simulations

The main objective of Hardware-in-the-loop simulation (HILsim) is to demonstrate the capability of the RDTA architecture in a high fidelity environment. Figure 3.22 shows the architecture that is used in these experiments. In this setup, the commercially available autopilot from CloudCap Technology receives waypoint commands from the on-board planning module (OPM) and sends the control inputs to a PC simulating vehicle dynamics. The autopilot measures the vehicle states, as generated by the HIL Simulation PC, uses them for flight control, and also routes them to the OPM as it would in flight. Finally, the states are downlinked to the ground station for the operator to monitor them.

Figure 3.23 is a picture of the HILsim setup. Hardware sufficient to emulate six vehicles resides at MIT. Figure 3.23 shows a simulation with a team of three UAVs. For this setup, a total of three avionics boxes and eight laptop computers makes up the simulation:

- On the left, a single laptop acts as the central ground station for all of the UAVs; it runs the Piccolo avionics monitoring software and the Mission Manager GUI.
- Immediately to the right of the ground station is the rack of Piccolo avionics systems. These avionics boxes are given simulated sensor data through CAN bus connections to the environment simulation computers (next item).

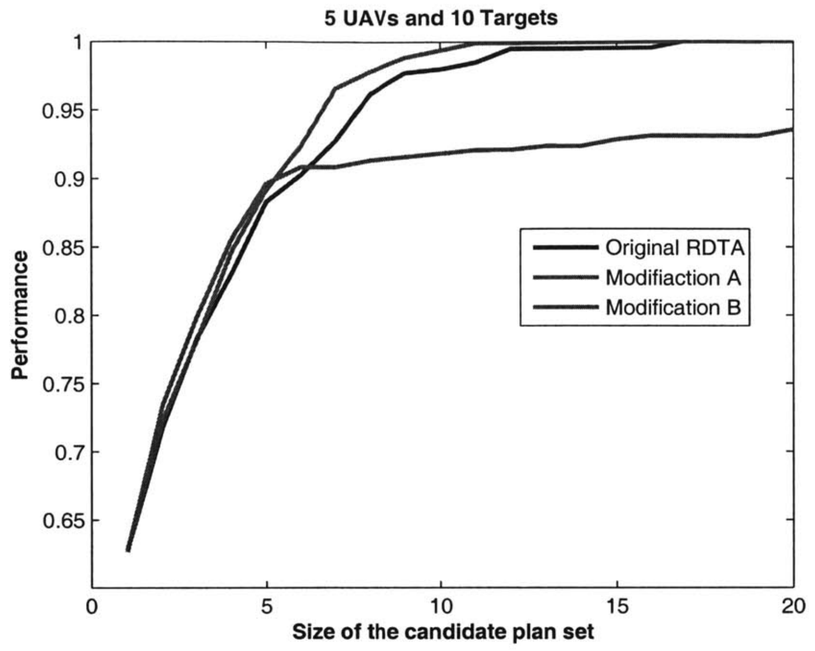


Figure 3.20: Comparing the performance of the original RDTA with its two modifications for a case of 5 UAVs and 10 targets.

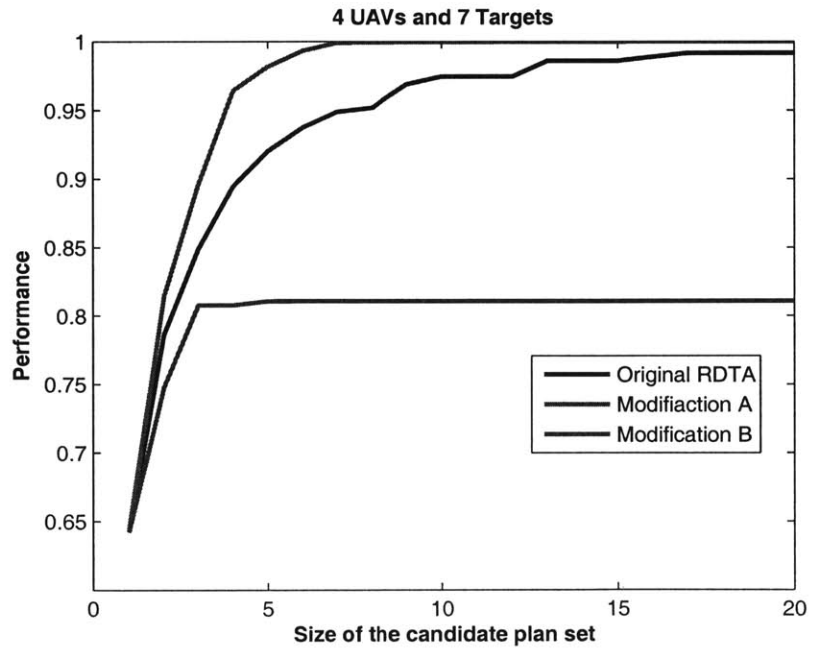


Figure 3.21: Comparing the performance of the original RDTA with its two modifications for a case of 4 UAVs and 7 targets.

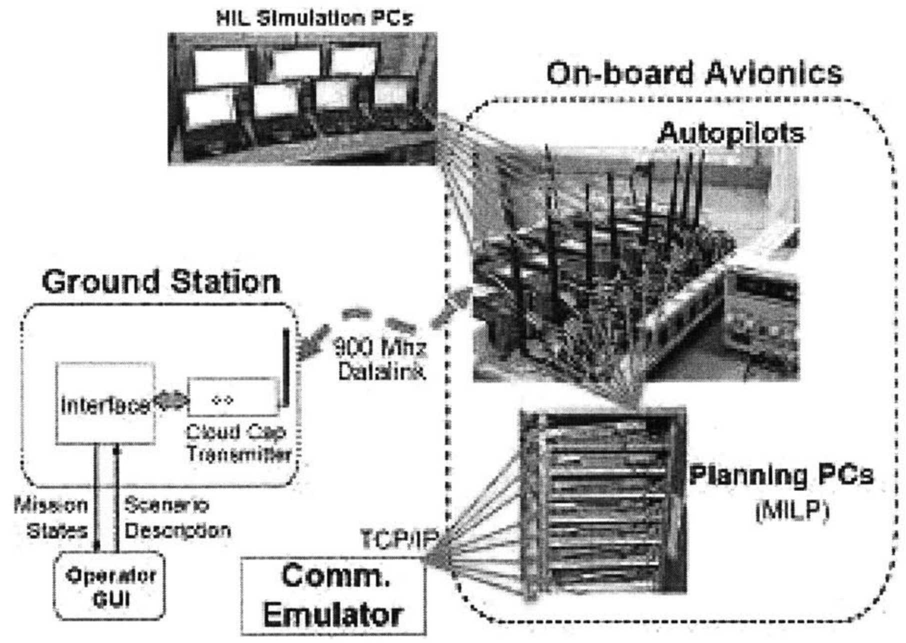


Figure 3.22: Hardware-in-the-loop simulation architecture with communication emulator.

- On the top shelf of the bench, three small laptops run the environment simulations and emulate the sensors on board the Piccolo avionics.
- On the bottom shelf, three larger laptops run the on-board software, as well as the CPLEX installations needed to solve the planning optimization. These computers communicate with their respective Piccolo avionics through a serial port (shown on Figure 3.24). The only difference between this setup (Figure 3.23) and the one in Figure 3.24 is that the wireless connection has been replaced by a hard line.
- The fourth laptop on the bench acts as the network emulator server. The three OPMs are interconnected via a TCP/IP router, and all messages between the OPMs are mediated by a wireless network emulator.

We emphasize that this is a true distributed task assignment environment in that:

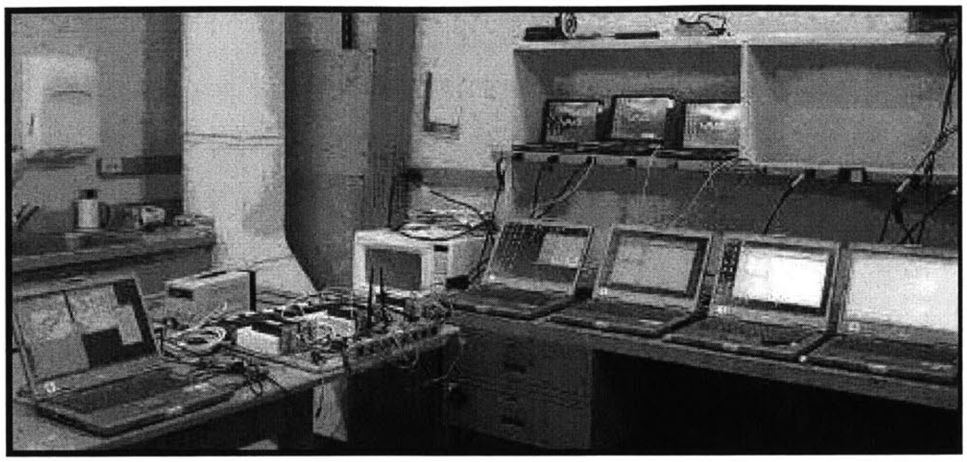


Figure 3.23: Hardware-in-the-loop simulation setup.

- The RDTA algorithms are being executed on separate OPM / laptops.
- Each OPM is getting its information from a separate Piccolo computer; and
- All communication between the OPMs is mediated by the wireless network emulator.

The RDTA process has been implemented within the dynamic network, and various tests have been performed. Figure 3.25 shows a simulation with 3 UAVs and 4 targets, where each UAV can only visit up to 2 targets at each assignment. Assignments are made every minute and the simulation shown was run for 400 seconds. The UAVs start at the center of their respective search areas. Each UAV has perfect knowledge of its own position and communicates it to the other UAVs periodically. On the other hand, the targets' positions are not known precisely and are estimated by distributed Kalman Consensus Algorithm. On each UAV, each target estimate at time $t-1$ is propagated, updated with the available measurements of the target's positions, communicated to the other UAVs, and averaged with the recently received estimates from the other UAVs with coefficients that depend on the estimates' covariance, to produce the target's positions estimate at time t. The measurements can

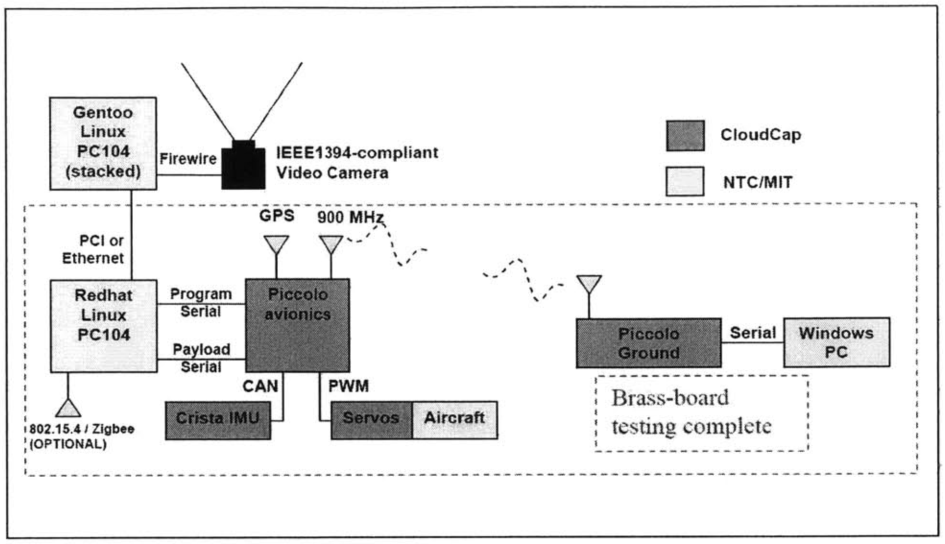
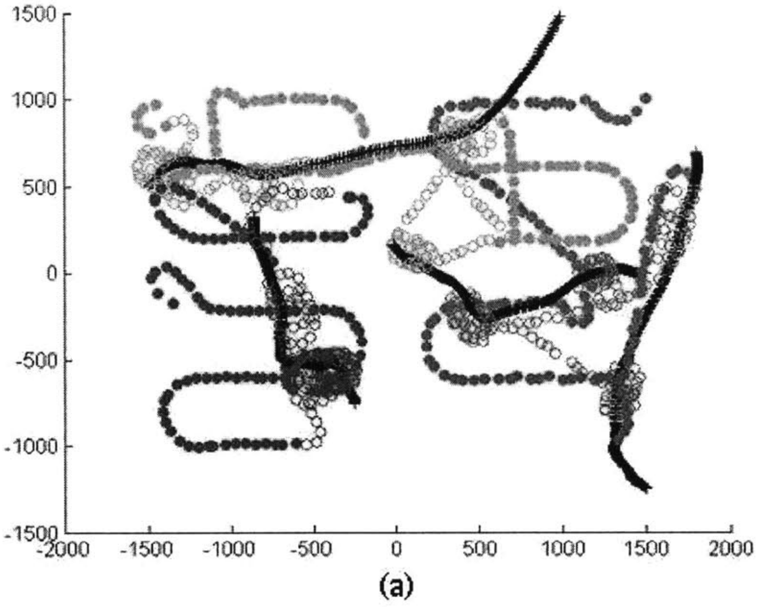


Figure 3.24: Architectures for the on-board planning module (OPM).

only take place if the UAV is within a certain range of the target (here 400 m). For the simulation shown in Figure 3.25, each UAV is typically able to detect between 2 and 4 targets at all times. To avoid overloading the communication network, the consensus phase in which the UAVs share their estimates only takes place during the last 30 seconds before a new assignment is made. The appropriate (typically nearest) UAV visits a target track when it has not been visited for a long period of time. If the UAV has been assigned 2 targets, it will track the first target for a specified period of time before it starts flying towards the second one. If all targets have been visited recently, the UAVs return to their search zones. In Figure 3.25 (b), the search mission is executed in parallel to the tracking tasks, and by the end of the simulation, 50–100% of the search has been performed, while the target tracks are updated on a regular basis.



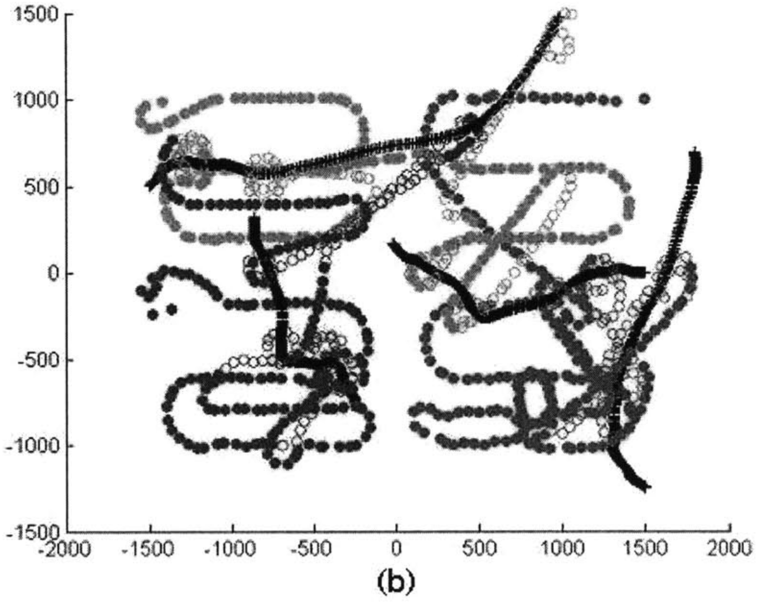


Figure 3.25: HILsim of 3 vehicles servicing 4 search patterns, while 4 undiscovered targets randomly traverse the search space. A different color is assigned to each UAV, and the circles change from closed to open when the UAV is tasked with tracking. (a) and (b) show runs of different durations, to show how search and track proceed over time.

3.7 Conclusions

The success of an implicit coordination approach in which the central assignment algorithm is replicated on each UAV strongly depends on the assumption that all UAVs have the same situational awareness. The examples showed that this consensus is necessary, but potentially time consuming. This chapter presented an extension of the basic implicit coordination approach that assumes some degree of data synchronization, but adds a second planning step based on shared planning data. The resulting Robust Decentralized Task Assignment method uses these shared candidate plans to overcome any residual disparity in the information at the end of the (possibly truncated) consensus stage. The simulations demonstrated the advantages of this new method in generating feasible plans that reduced the conflicts in the assignments and improved the performance compared to implicit coordination.

Further results demonstrated the effect of communication on the performance of assignment in different stages of the planning. The performance of the RDTA algorithm for different communication network topologies was also analyzed and it was shown that the communication during the planning phase introduced in this new technique is crucial to achieve high performance. This is especially true for sparse communication networks where the slow convergence of the information consensus results in decentralized activity planning based on inconsistent data. To analyze the sensitivity of the overall performance to the candidate plan set, four selection algorithms are presented. A comparison of the performance for these algorithms clearly shows the importance of accounting for the potential actions of other UAVs in the selection process. A modification of the original candidate plan selection algorithm was also presented to further improve the overall performance by increasing the robustness to inconsistencies in the information across the team.

Chapter 4

Decentralized Auction-Based Task Assignment

4.1 Introduction

Centralized algorithms are usually easy to implement and the optimal answer for many problems can be produced if enough computation power is available. Many centralized algorithms have been proposed over the years to solve the task assignment problem [2, 10, 22, 27, 52, 72]. A centralized algorithm usually requires the planner to gather the information from all the agents and to plan based on the aggregated information. This requires a communication structure that allows continuous transmission of a massive amount of data [4, 5]. Each agent also needs to be in continuous contact with the central planner to send new information and receive a new assignment (a full handshaking structure). This usually requires a wide bandwidth communication structure that might not always be possible.

Besides the communication bandwidth limits that may prohibit the use of a centralized planning algorithm, the response time can also be an issue. In many planning problems, one of the objectives or constraints is the planning time or the response time to new discoveries. The limited bandwidth of the communication network along with the large amount of data that needs to be sent to the central planner results in long planning times and consequently long response times.

Another limiting factor of the centralized algorithms is its computation requirement, which might be impractical. Since the optimization problem of the interest usually grows (faster than linear) with the number of agents, the centralized algorithms do not scale well. Given these limitations, many researchers have investigated decentralized and/or distributed planning [21, 23, 25, 26, 29, 33, 48, 53, 54, 63], and as a result many different algorithms have emerged.

Chapter 3 presented the Robust Decentralized Task Assignment (RDTA) as a decentralized method to solve the UAV task assignment problem. The *implicit coordination* algorithm was also presented to be used as a benchmark for evaluating RDTA. The results showed that RDTA results in robust, conflict-free assignments with much less communication compared to the implicit coordination algorithm. This was achieved by effectively closing the communication loop around the planning algorithm and communicating processed data instead of raw information. This enables the algorithm to achieve good performance and robustness without the perfectly consistent situational awareness, which is a requirement for the implicit coordination algorithm.

Although the RDTA algorithm performs well with most of the communication networks, it has a minimum requirement [4, 5]. In the second stage of the algorithm, the set of candidate plans is transmitted to every other UAV in the team. When the networks are not complete (a direct link from each UAV to every other UAV does not exist), UAVs have to be able to relay the information from one neighboring UAV to other neighboring UAVs. This requirement does not impose a major limitation and is usually satisfied in UAV networks, but if not satisfied, it can result in significant performance degradation of the RDTA algorithm.

The implicit coordination algorithm does not require this type of network connectivity or relaying capability; however, it is very communication inefficient. In the implicit coordination algorithm, UAVs run a consensus algorithm that is completely decentralized. This will result in a consistent information set among all the UAVs. This consistent information set will result in a consistent plan when all UAVs run the same deterministic centralized task assignment algorithm onboard. Assuming that all these consistency assumptions are met, the implicit coordination algorithm performs well. However, reaching consistent information can be cumbersome if the information set is large, which is usually the case for realistic size problems. Note that this limitation was one of the motivations that led to the development of the RDTA algorithm.

The objective of this chapter is therefore to develop a task assignment algorithm that eliminates both limitations discussed above. It should perform the assignment in a complete decentralized manner, where each agent is only allowed to communicate with its neighboring agent (the agents with direct communication link) and there is no relaying of information. At the same time it should be communication efficient, where only necessary information is communicated.

Another class of decentralized algorithms for the task assignment problem is the auction algorithms [14, 36, 40, 49, 65]. The basic idea in the auction assignment algorithms is for the agents to bid on each task, and the agent with the highest bid gets assigned to that specific task. Over the years, many different methodologies have emerged from this basic idea to improve performance and convergence rate of the auction algorithm. In a classic auction algorithm, one agent acts as the auctioneer and evaluates the bids of different agents and assigns each task to the appropriate agents. In some cases, the auctioneer is removed from the algorithm and one of the bidders acts as the auctioneer. In general, the bids are collected and a decision is made by looking at all the bids. This specification of the auction algorithm requires the agents to be able to send information either to a specific agent who acts as the auctioneer or to all other agents. This usually requires a complete communication network (a link from any agent to every other agent) or relaying capability in the agents.

The algorithm developed in this chapter combines the basic auction assignment idea with the consensus algorithm idea and creates an algorithm to solve the task assignment problem. The resulting Auction-Based Task Assignment (ABTA) algorithm eliminates the issues of the basic auction algorithms that were discussed above.

This chapter first states the task assignment problem and then proposes a new decentralized auction-based algorithm to solve this problem. The completeness of this algorithm is then proved and extensive simulation results are presented to compare the performance and communication of the proposed algorithm with the benchmark. Further modifications are presented to improve the performance of the algorithm. Note that although this work originally was developed for the UAV task assignment problem, the resulting algorithm can be implemented for any task assignment problem. Therefore, in the rest of the chapter, the more general word *agent* is used instead of UAV.

4.2 Problem Statement

The problem is a simple task assignment problem where there are N_u agents and N_t tasks, and there is a value associated with assigning agent i to tasks j represented by c_{ij}. The objective is to assign at most one task to each agent in order to maximize the overall assigned values. The centralized problem can be formulated as a simple linear integer programming problem:

$$\begin{aligned}
&\max\ J = \sum_{i=1}^{N_u}\sum_{j=1}^{N_t} c_{ij}x_{ij} \\
&\text{subject to} \quad \forall i = \{1,\dots,N_u\} : \sum_{j=1}^{N_t} x_{ij} \le 1 \\
&\qquad\qquad\quad\ \forall j = \{1,\dots,N_t\} : \sum_{i=1}^{N_u} x_{ij} \le 1 \\
&\qquad\qquad\quad\ \forall j = \{1,\dots,N_t\}, \forall i = \{1,\dots,N_u\} : x_{ij} \in \{0,1\}
\end{aligned} \tag{4.1}$$

Ref. [13] shows that although the variables in this problem are binary, a simple linear programming approach can result in a solution that satisfies the binary constraints. Therefore, there is no need for using complicated integer programming approaches such as Branch and Bound, etc. This makes the problem easy and fast to solve for even a large number of tasks and agents when it is done in a centralized fashion.

The objective here, however, is to solve this problem without a centralized planner and by distributing the planning over the agents. Suppose that agents are connected through a communication network $\mathbb{G}(t)$ with associated adjacency matrix $G(t)$, where $G_{ik}(t) = 1$ if there is a communication link from agent i to agent k (i is a neighbor of k), and $G_{ik}(t) = 0$ otherwise. The idea is for the agents to communicate with their neighbors to create complete conflict-free assignment with the defined objective. It means that either all the agents or all the tasks (whichever is smaller) are assigned and the constraints in Eq. 4.1 are satisfied (no conflicts).

4.3 The New Decentralized Approach

The basic idea is for each agent to act in a greedy way and choose the best task for itself. It then communicates with its neighbors to see if it is the best agent to be assigned to that specific task. If by communicating with the neighbors it finds out that there is another agent that can achieve a better value by being assigned to this specific task, it discards the task and assigns the next best task to itself. This is very similar to the auction algorithm idea that is used in the literature. In the Auction algorithm, all the agents bid on each task and the one with the highest bid gets assigned to the task.

The algorithm that is proposed here combines the ideas in the auction algorithms and the consensus algorithm. In words, each agent selects the best task for itself with all the information that it has from other agents. It then exchanges information with its neighbors and adjusts its assignment based on what it receives. The assumption is that even though the communication network is not fully connected at every time step, the union of the communication networks over a certain period of time is strongly connected, and therefore the information from every agent will eventually reach every other agent.

4.3.1 Algorithm Development

In the following, the new decentralized Auction-Based Task Assignment (ABTA) algorithm is formulated. We introduce two variables, x, y as follows: $x_{ij}(t) = 1$ if task j is assigned to agent i at time t, and is equal to zero otherwise. The variable $y_{ij}(t)$ is agent i's knowledge of what is achieved by executing task j. This value is a local estimate of the object price in the auction algorithms. The difference in this case is that this price could be different for each agent, whereas in the auction algorithms the agents all have the same price for any given object at any given iteration. In a sense, y_{ij} is the local price for object j.

Each iteration has two stages. In stage one of the algorithm, each agent i looks at x_{ij}'s and if $x_{ij} = 0 : \forall j$ (no task is assigned to i) then it chooses a task j (if there is any task available) and assigns it to itself ($x_{ij} = 1$, $y_{ij} = c_{ij}$). For stage one at iteration t, the algorithm is written as follows:

```
Stage 1:
1: $\forall i \in \{1, \dots, N_u\}$
2: **if** $\sum_j x_{ij}(t-1) = 0$ **then**
3:     $D_{ij} = [c_{ij} > y_{ij}(t-1)], \forall j \in \{1, \dots, N_t\}$
4:     $J = \arg\max_j \; D_{ij}\,(c_{ij} - y_{ij}(t-1))$
5:     $x_{iJ}(t) = 1$
6:     $y_{iJ}(t) = c_{iJ}$
7: **end if**
```

Description: Each agent $i \in \{1, \dots, N_u\}$ checks to see if it is assigned to any task or not (line 2). If it is assigned then it goes to stage 2 of the algorithm as described below. If not, it compares its own value for each task j, c_{ij} to the current local price (value) of task j, $y_{ij}(t)$ and creates a vector D, with elements $D_{ij} = 1$ if $c_{ij} > y_{ij}$ and 0 otherwise (line 3). In line 4, the agent finds the task j with $D_{ij} = 1$ and also improves the overall objective the most (i.e. where $c_{ij} - y_{ij}$ is maximum). It then assigns this task to itself (line 5) and updates the price for this task (line 6).

Stage two of the algorithm is the conflict resolution. In stage two, at iteration t

for agent i and task j:

$$\begin{aligned} y_{ij}(t) &= \max_k G_{ik}(t) y_{kj}(t) && (4.2) \\ K_{ij} &= \arg\max_k G_{ik}(t) y_{kj}(t) && (4.3) \\ x_{ij}(t) &= \begin{cases} 0 & \text{if } K_{ij} \neq i \\ x_{ij}(t) & \text{otherwise} \end{cases} && (4.4) \end{aligned}$$

Description: Each agent i receives the prices y_{kj} for every task j from all its neighbors k, $(G_{ik}(t) = 1)$. It then updates y_{ij} with the maximum of these values $(y_{ij},\ y_{kj} : G_{ik} = 1)$(Eq. 4.2). If the maximum price was its own bid, then nothing changes, but if the maximum price belonged to a neighbor, then the agent sets $x_{ij} = 0$, which means that agent i is outbid for this task (Eqs. 4.3, 4.4). Note that stages one and two are iterated until all the tasks or agents are assigned.

The first stage of the algorithm is essentially the bidding process of the auction algorithm. Each agent looks at the previous prices by other agents (y_{ij}) and its own task values (c_{ij}) and decides which task to bid on. The primary objective for the agent is to achieve the highest value by visiting its one task, which means bidding on the task with the highest value c_{ij}, but it also needs to consider previous bids of other agents (or the price of the task) y_{ij}. Therefore, it only looks at the tasks with $c_{ij} > y_{ij}$ and finds the maximum between them and bids on that task. The strategy is to iterate between stages 1 and 2 to create the desired solution.

The following section proves the convergence of the ABTA algorithm to a complete conflict-free assignment. Simulation results are then presented to analyze different aspects of the algorithm and to compare its performance with the benchmark centralized algorithms.

4.3.2 Proof of Convergence

This section gives a proof that the ABTA algorithm converges to a complete conflict-free assignment. This means that if $N_u \leq N_t$ then all the agents are assigned to one and only one task, and each task is assigned to at most one agent. If $N_u \geq N_t$, then each task is assigned to one and only one agent, and each agent is assigned at most to one task. In other words, the constraints of the optimization in Eq. 4.1 are satisfied.

For simplicity, assume for now that the communication network is static and strongly connected, which will be relaxed later. With this assumption, there is a bound $\mathbf{B} \leq N_u$ where after at most $\mathbf{B}$ iterations information from any agent i will have reached any other agent k. Note that the propagation of the information is achieved using an information consensus algorithm and does not require relaying information of one neighbor's information to another neighbor. It is also assumed that the agents' values for each task are not identical $c_{ij} \neq c_{kj}, \forall i \neq k$. This is a requirement for the ABTA algorithm to converge and can be ensured by adding a very small random number to every c_{ij}.

Let $I = \{1, \ldots, N_u\}$ and $J = \{1, \ldots, N_t\}$. At the beginning,

$$\exists j^*, i^* \quad s.t. \quad \forall i \neq i^*, j \neq j^*, \quad c_{i^*j^*} > c_{ij} \tag{4.5}$$

Based on stage one of the algorithm,

$$y_{i^*j^*} = c_{i^*j^*} \ , \ x_{i^*j^*} = 1.$$

Then, after at most $\mathbf{B}$ iterations,

$$\begin{aligned} y_{ij^*} &= c_{i^*j^*} \quad \forall i \in I \\ x_{i^*j^*} &= 1 \\ x_{ij^*} &= 0 \quad \forall i \in I - \{i^*\} \end{aligned} \tag{4.6}$$

Looking at stages 1 and 2 of the algorithm it can be seen that neither x_{ij^*} nor y_{ij^*} will be changed in future iterations; therefore, the assignment of task j^* will not be changed and it can be removed from the task list, $J = J - \{j^*\}$. Also, since agent i^* is assigned and its assignment does not change, it can be removed from the list of agents $I = I - \{i^*\}$. With the updated I and J, there are two possibilities:

1. $\exists j^*, i^* \quad s.t. \quad \forall i \in I,\ i \neq i^*,\ \forall j \in J,\ j \neq j^*; \quad c_{i^*j^*} > c_{ij} \ \ \& \ \ y_{i^*j^*} = c_{i^*j^*}$
2. $\exists j^*, i^* \quad s.t. \quad \forall i \in I,\ i \neq i^*,\ \forall j \in J,\ j \neq j^*; \quad c_{i^*j^*} > c_{ij} \ \ \& \ \ y_{i^*j^*} < c_{i^*j^*}$

In case 1, $x_{i^*j^*} = 1$. For case 2, in stage one of the algorithm again,

$$x_{i^*j^*} = 1, \quad y_{i^*j^*} = c_{i^*j^*} \tag{4.7}$$

Now for both cases 1 and 2, the same argument as before can be made and

$$\begin{aligned} y_{ij^*} &= c_{i^*j^*} \\ x_{i^*j^*} &= 1 \\ x_{ij^*} &= 0 \quad i \neq i^* \end{aligned} \tag{4.8}$$

After repeating this process for the lesser of N_u and N_t times, either all the agents or all the tasks are assigned.

So far, it is shown that after a finite number of iterations, the algorithm terminates with a complete assignment, where either every agent is assigned to a task or every task is assigned to an agent, depending on the relationship between N_u and N_t. Now we show that at termination, each task is assigned to at most one agent and each agent is assigned to at most one task.

Recall the assumption that the c_{ij}'s are different. Thus, if there are two agents i and k for which $x_{ij} = x_{kj} = 1$ and $c_{ij} > c_{kj}$, then after at most $\mathbf{B}$ iterations, y_{kj} converges to c_{ij}, for which $x_{kj} = 0$. Thus, there can not be any task j that is assigned to two agents or $x_{ij} = x_{kj} = 1$, which means that each task is assigned to at most one agent. For the second problem, since the assignment of agents to tasks is only done in stage one and in this stage each agent is only assigned to one task, the result follows by construction.

This proof was given for static and strongly connected networks, but it can easily be extended to a more general class of networks. For this proof to stand for dynamic networks, the following assumption needs to be made about the communication network.

Assumption. (bounded intercommunication interval) If i communicates to j an infinite number of times, then there is some B such that, for all t, $(i,j) \in E(t) \cup E(t+1) \cup \cdots \cup E(t+B-1)$.

With this assumption, if $\mathbf{B}$ in the proof is replaced with a new bound $\mathbf{B} \leq B \cdot N_u$, the rest of the proof follows and therefore the algorithm is complete for any communication network that satisfied the above assumption.

4.4 Simulation Results

The following analyses are presented to demonstrate different properties of the ABTA algorithm and compare them with the benchmark.

4.4.1 Performance Analysis

Monte Carlo simulations are used to analyze the performance of the ABTA algorithm. In these simulations, the communication network is randomly generated at each time step. The task values c_{ij} are also drawn randomly from the uniformly distributed set $[1, 100]$. The ABTA algorithm and the optimal centralized algorithm are then used to solve these randomly generated problems. The comparison is made for different numbers of tasks and different numbers of agents.

Figure 4.1 shows the result of these simulations. The x- and y-axes are the number of agents, N_u, and the number of tasks, N_t, respectively. The z-axis is the percentage deviation of the objective value from the optimal solution. Each point in this graph is the result of 625 Monte Carlo simulations where both the task values c_{ij}'s and networks are created randomly. The results show that the maximum deviation from the optimal case is around 7%, which shows that the ABTA algorithm performs very well. Note that the deviation is the highest when the number of tasks and agents is equal, $N_u = N_t$. Figure 4.2 gives the diagonal slice of Figure 4.1 ($N_u = N_t$). This corresponds to the worst performance of the ABTA algorithm for different numbers of agents and tasks. This plot clearly shows the trend discussed above, wherein this sub-optimality reaches its maximum at around $N_u = N_t = 20$ and drops/flattens to

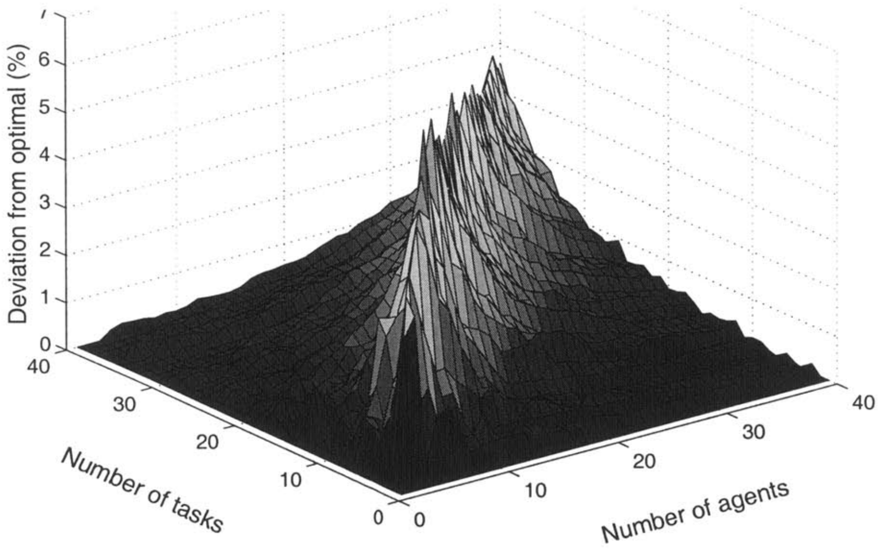


Figure 4.1: Performance of the ABTA algorithm compared to the optimal solution for different numbers of agents and tasks.

approximately 2% sub-optimality for larger values.

4.4.2 Communication Analysis

One alternative to the ABTA algorithm is to run consensus on the information of each agent (its task values) and then run a centralized algorithm on each agent to achieve the optimal solution. This is essentially the so called *implicit coordination* algorithm [25]. One of the potential advantages of the ABTA algorithm to this IC is the amount of communication and bandwidth required. Here simulations are used to support this claim.

To make a fair comparison, two quantities are tracked. The first quantity is the amount of information that is sent out from each agent at each iteration. Assume that the communication is in the form of broadcast where each agent sends out its information once and all its neighbors receive it. This assumption does not have any effect on the comparison and is made simply to make the comparison more straightforward. In the case that the communication is one to one (peer to peer), then the communication has to be multiplied by each agent's number of neighbors.

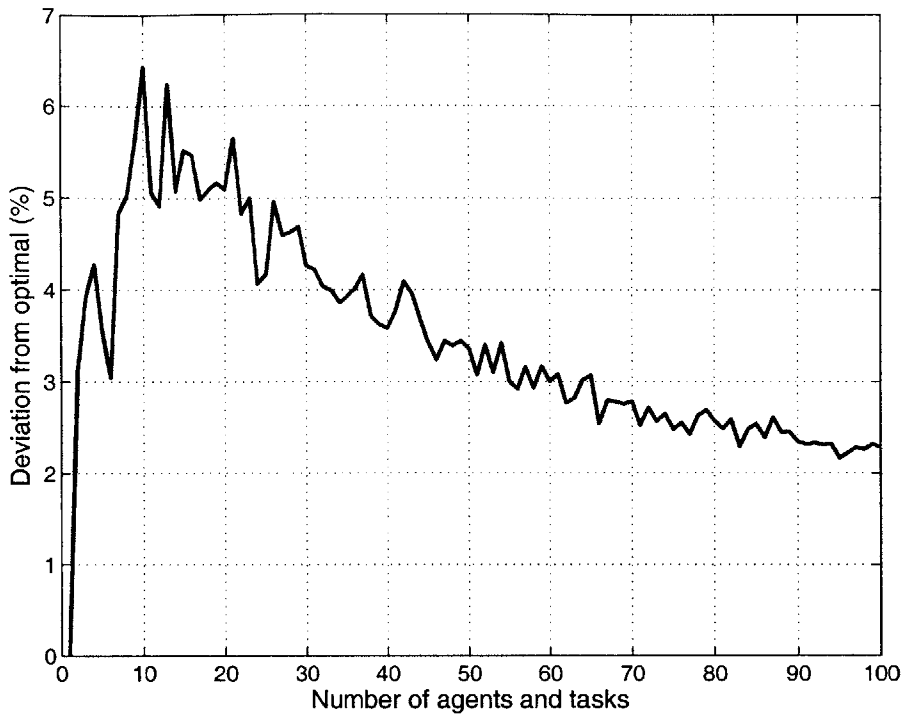


Figure 4.2: Trend of suboptimality (deviation from optimality) for different numbers of tasks and agents ($N_t = N_u$).

Since in these analyses the ratio of communication for two algorithms is used, these multiplications cancel each other and therefore have no effect on the analysis. In the implicit coordination algorithm, agents communicate to reach consensus on the information (here the value of each task for each agent, c_{ij}). The total number of parameters to reach consensus on (all agents have the same value for each parameter) is therefore $N_u \times N_t$, which is essentially the amount of data transmitted from each agent in each iteration. For the ABTA algorithm, however, at each iteration, the data that is sent from each agent is y_{ij} for that specific agent, which is a total of N_t parameters.

Figure 4.3 compares the quantity introduced above (the information out of each agent at each iteration) for the two algorithms. The z-axis here is the ratio of this quantity for implicit coordination (IC) to the one of the ABTA algorithm,

$$\frac{Comm.\ in\ IC}{Comm.\ in\ ABTA} = \frac{N_u \times N_t}{N_t} = N_u \geq 1$$

Note that this ratio is always greater than or equal to one, which shows the advantage of the ABTA algorithm to the implicit coordination algorithm. This advantage becomes more apparent and important as the number of agents increases.

Another comparison can be made for each agent's total amount of communication to complete the assignment. This is essentially the previously compared quantity times the number of iterations required. The number of iterations for the implicit coordination is the number of iterations to reach consensus on the c_{ij}'s and for the ABTA algorithm is the number of iterations to create a conflict-free assignment. To make this comparison, the Monte Carlo simulation of Section 4.4.1 is repeated and the results are shown in Figure 4.4. Similar to Figure 4.3, here the z-axis is the ratio of total communication (total number of bytes) for the implicit coordination to the ABTA algorithm.

Figure 4.4 shows that this ratio is always greater than or equal to one, which means the amount of communication required in the implicit coordination is always greater that the one in the ABTA algorithm. It also shows that this ratio increases with the number of agents.

To see the trend of the total communication ratio shown in Figure 4.4, the simulations were run for larger fleet sizes and the results are shown in Figure 4.5. In this figure, only the cases where $N_u = N_t$ are considered. Figure 4.5 shows that the ratio of the total communication increases linearly with the number of agents, which supports the previous claims. It should also be noted that the total communication presented in these figures can be interpreted as the time to create the assignment or the *assignment time.* Since the computation involved in both algorithms is negligible, the assignment time is essentially the time spent on communicating the information.

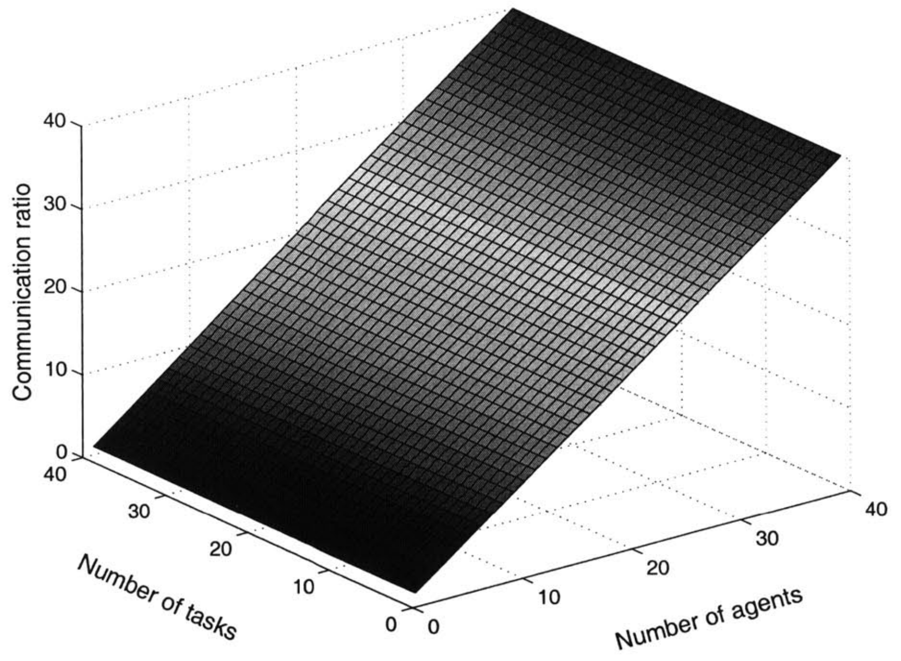


Figure 4.3: Ratio of communication of each agent at each iteration (Implicit coordination to the ABTA algorithm).

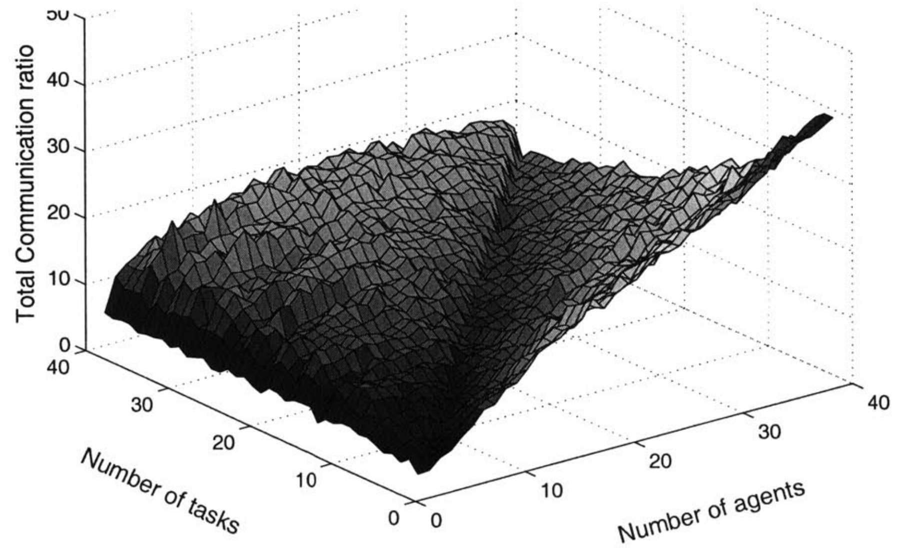


Figure 4.4: Ratio of total communication of each agent (Implicit coordination to the ABTA algorithm).

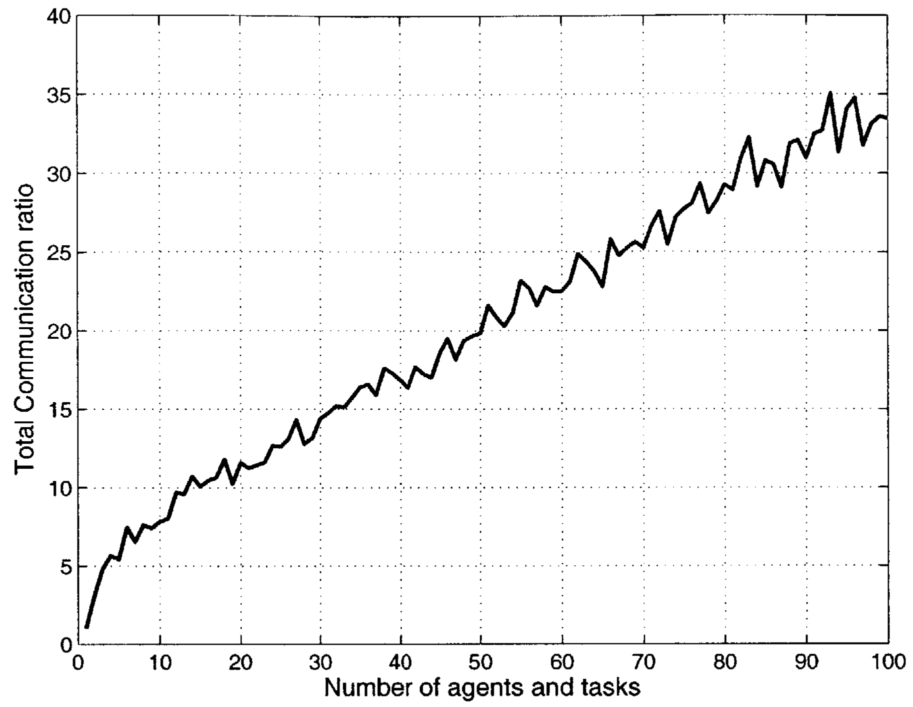


Figure 4.5: Trend of total communication ratio for different numbers of tasks and agents.

4.4.3 Effect of Network Sparsity

In this example the effect of the communication network and the degree of its connectivity on the performance and communication of the ABTA algorithm is analyzed. A measure of sparsity (or connectivity) of a network of N_u agents is the number of non-zero elements of the adjacency matrix of the communication network, G. This is equal to the number of links in the communication network. Here almost a similar notion is used to quantify the degree of sparsity of the network. In the simulations, the networks are generated randomly and thus the number of links varies from one time step to the next. Therefore, in order to quantify the sparsity, instead of using the number of links, the average number of links over all the iterations (time steps) is used. In the Monte Carlo simulations, in order to create a random network at iteration t, the following command is used in MATLAB,

$$G(t) = (\texttt{rand}(N_u) \geq \texttt{Threshold})$$

In this case, the value of $0 \leq \texttt{Threshold} < 1$ represents the average ratio of the number of zeros to the number of ones in the adjacency matrix G and is used in our analysis to quantify the sparsity of the communication network. The higher the `Threshold` the higher the ratio of zeros to ones and therefore the more sparse is the communication network.

In the set of simulations for this example, the number of tasks is set to $N_t = 15$ and the number of agents is changed from $N_u = 1$ to $N_u = 30$. The degree of sparsity is also changed from $\texttt{Threshold} = 0$ (fully connected) to $\texttt{Threshold} = 0.9$ (very sparse).

Figures 4.6 and 4.7 show the performance and communication results respectively. Similar to the previous figures, Figure 4.6 shows the percentage of performance deviation from optimal for the different cases. Here, the x-axis is the number of agents and the y-axis is the measure of sparsity defined above (`Threshold`). The figure shows that the performance of the algorithm does not change significantly with the sparsity. This is because the sparse communication network requires more iterations for the information to propagate, which does not affect the performance of the ABTA algorithm.

The amount of communication, however, changes significantly with the sparsity of the communication network. For more sparse communication networks, the number of iterations in the implicit coordination to reach consensus as well as the number of iterations in the ABTA algorithm to converge to a conflict-free complete assignment increases. Figure 4.7 shows the total communication ratio of the implicit coordination to the ABTA algorithm. This figure does not show an obvious trend, but it shows that this quantity is always much larger than one, which shows the communication efficiency of the ABTA algorithm for different degrees of sparsity. Looking at slices of the figure by keeping the number of agents fixed and varying the sparsity, it can

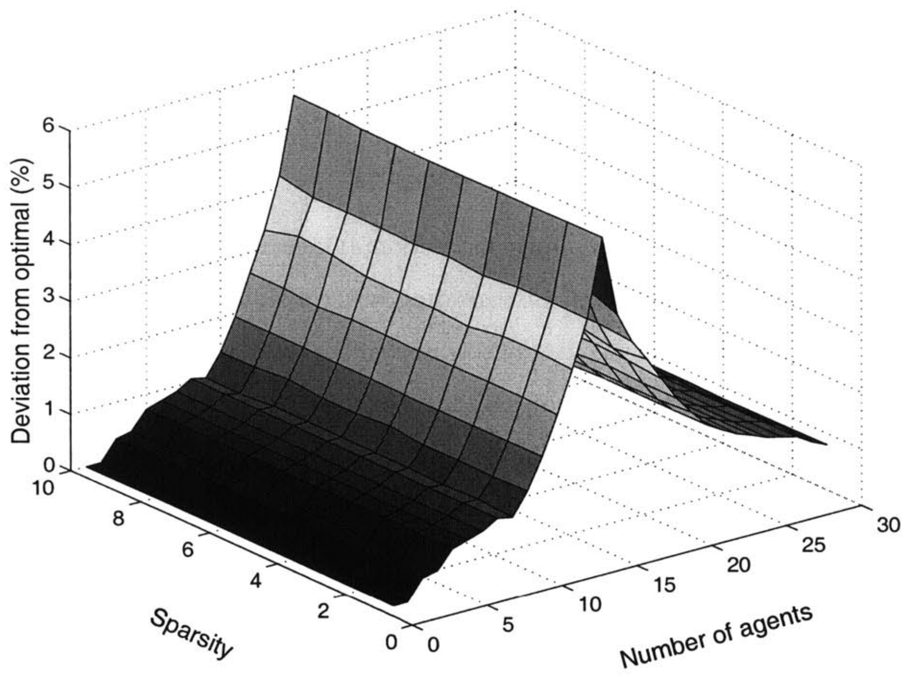


Figure 4.6: Performance of the ABTA algorithm for different communication networks (different levels of sparsity).

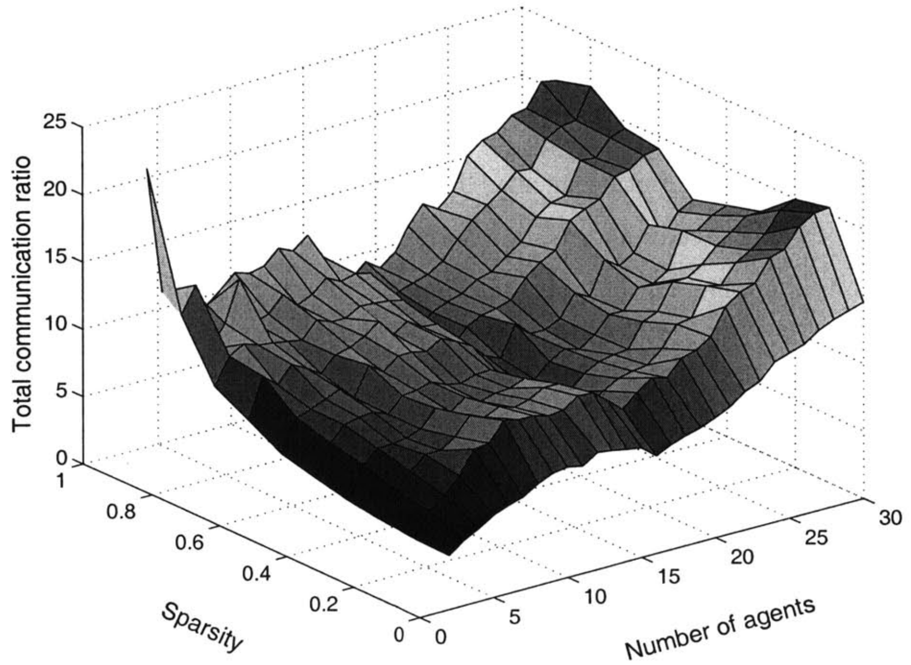


Figure 4.7: Total communication ratio for different communication networks (different levels of sparsity).

be seen that the communication ratio (advantage of ABTA over IC) increases as the communication network becomes more sparse.

4.4.4 Effect of Prior Information

In the ABTA algorithm developed in Section 4.3, each agent selects a task for itself based only on its own values, disregarding other agents. It then communicates its choice with neighbors to ensure there is no conflict, and at the end reaches a conflict-free assignment. In the case that agents have no notion of other agents' information, this is essentially the best that can be achieved. But in cases where agents have some knowledge of other agents' information, using this information might help improve the performance and convergence rate of the algorithm.

The following presents an approach to take advantage of an agent's knowledge of other agents' information. Suppose that each agent i knows exactly what value it achieves by getting assigned to each task, $c_{ij} : \forall j$, but its knowledge about the values of the tasks for other agents, $c_{kj} : k \neq i$ is inexact or noisy. The objective is to see if using this noisy information improves the performance and communication of the algorithm.

The algorithm is modified so that each agent selects its task by incorporating the expected action of other agents. To do so, in stage one of the ABTA algorithm, instead of being greedy and using a simple maximization, a centralized cooperative algorithm is applied to the noisy information to select the best task for agent i. A set of Monte Carlo simulations with the setup described in Section 4.4.1 is used here to show the effect of this modification. In these simulations, the number of tasks is set to $N_t = 15$ and the number of agents is changed from $N_u = 1$ to $N_u = 30$. Agent i has a noisy (inaccurate) knowledge of agent k's value $c_{kj}, k \neq i$, which is defined to be c^i_{kj}. For each agent i these values are created by adding random noise to the true value of the parameters:

$$c^i_{kj} = c_{kj} \cdot (1 - \lambda r)$$

where r is a random number drawn from a uniform distribution in $[-1, 1]$ and λ is

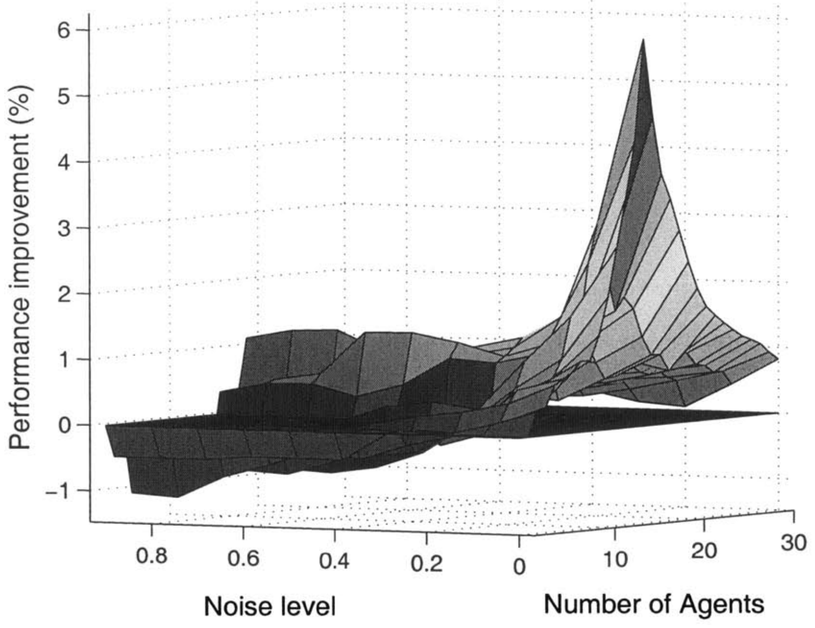


Figure 4.8: Effect of using prior information on the performance of the assignment. The flat surface at $z = 0$ is plotted to clearly show the positive and negative values.

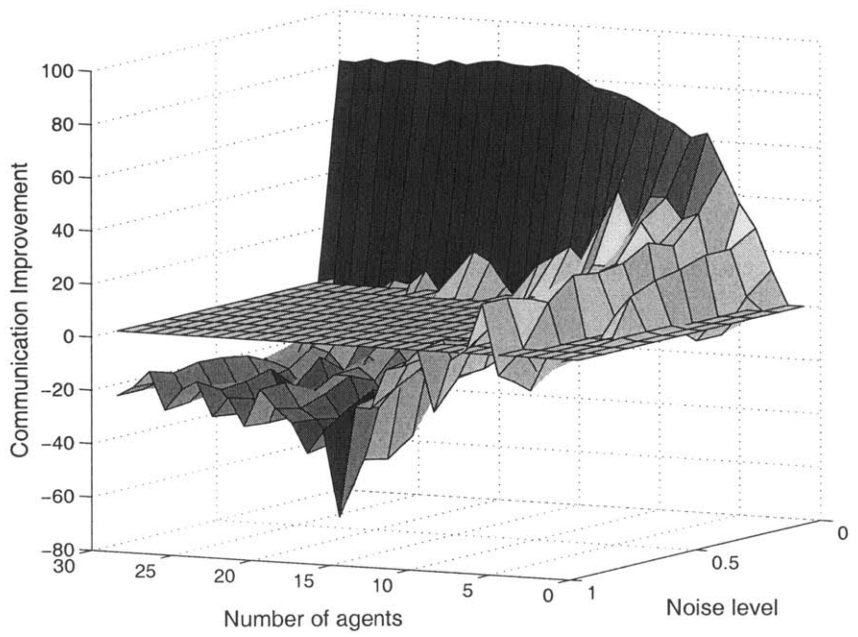


Figure 4.9: Effect of using prior information on the communication of the assignment.

a constant that sets the noise level. For instance, $\lambda = 0.2$ adds a noise with a value in the range of $[-20, 20]\%$ to the true value. In the simulations for this example, the value of λ was varied from $\lambda = 0$ (no noise) to $\lambda = 1$ (up to 100% noise), and the results are shown in Figures 4.8 and 4.9.

Figure 4.8 shows the percentage improvement of the performance of the modified cooperative algorithm compared to the original ABTA algorithm presented in Section 4.3. As expected, for smaller values of noise the modified cooperative algorithm helps to improve the performance while for the larger values i.e. $\lambda = 1$, this modification makes the performance worse. Note that the noise level of $\lambda = 0$ is essentially the optimal solution. This figure shows that if agents have relatively good information about other agents, then using the noisy information can substantially improve the performance. Figure 4.9 compares the total communication of the modified algorithm to the original ABTA algorithm. Similar to the performance graph, by using the modified algorithm, the communication is improved for smaller noise levels and is worse when the noise level is high.

4.5 Shadow Prices

To further improve the performance of the ABTA algorithm and possibly the convergence rate, an intuitive idea is incorporated into the algorithm through a new set of variables that are called shadow prices, z_{ij}. The idea is that when assigning agent i to task j, the assignment is not done selfishly and the bid of any other agent k that already has a bid on task j is considered. since the objective is to maximize the total value achieved by the team and not by any individual agent. Assuming that z_{ij} represents the score that agent k, currently assigned to j, will lose if agent i takes task j, then in order to make the decision to choose a task, agent i has to consider both y and z.

The modified algorithm using these new variables is presented here. After the initialization, at stage one,

```
$\forall i \in \{1, \dots, N_u\}$
if $\sum_j x_{ij}(t-1) = 0$ then
    $D_{ij} = c_{ij} > y_{ij}(t-1), \forall j \in \{1, \dots, N_t\}$
    $J = \arg\ \max_j\ \ D_{ij}\ (y_{ij}(t-1) - z_{ij}(t-1))$
    $MAX_i = \max_j\ \ D_{ij}\ (y_{ij}(t-1) - z_{ij}(t-1))$
    if $MAX_i > 0$ then
        $y_{iJ}(t) = c_{iJ}$
        $x_{iJ}(t) = 1$
        $D_{iJ} = 0;$
        $MAX2_i = \max_j\ \ D_{ij}\ (y_{ij}(t-1) - z_{ij}(t-1))$
        $z_{iJ} = c_{iJ} - MAX2_i$
    end if
end if
```

And the modified stage two of the algorithm will be:

$$\begin{aligned} y_{ij}(t) &= \max_k G_{ik}(t) y_{kj}(t) \\ K_{ij} &= \arg\max_k G_{ik}(t) y_{kj}(t) \\ z_{ij}(t) &= z_{kj}(t) \\ x_{ij}(t) &= \begin{cases} 0 & \text{if } K_{ij} \neq i \\ x_{ij}(t) & \text{otherwise} \end{cases} \end{aligned} \tag{4.9}$$

z_{ij}'s are called shadow prices since they resemble the shadow prices in linear programming. Looking at the optimization of Eq. 4.1, a set of constraints of the form $x_{ij} \leq 1$ is implicit in this formulation. z_{ij} is the change in the value achieved by an agent k if one of these constraints is changed from $x_{ij} \leq 1$ to $x_{ij} \leq 0$, which is essentially the shadow price of this constraint.

The Monte Carlo simulations described in Section 4.4.1 are used to compare the algorithm with shadow prices to the original ABTA algorithm. Figure 4.10 shows the result. Here the z-axis is the percentage improvement obtained using the modified algorithm over the original ABTA algorithm of Section 4.3.1. Note that the improvement is positive for most of the cases, which shows that the modification increases the performance of the algorithm. Although the improvements are small $< 1\%$, note that the maximum deviation from the optimal solution was around 7% and therefore

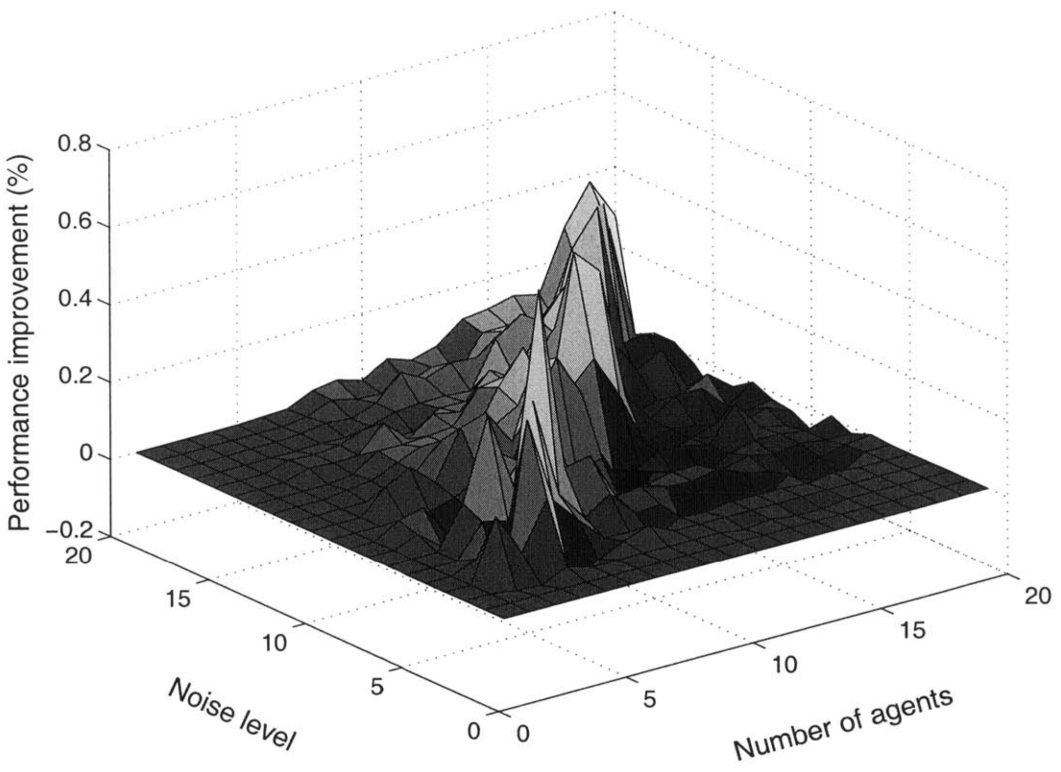


Figure 4.10: Effect of the modified algorithm (with shadow prices) on the performance of the algorithm.

these improvements are not negligible. Also note that the maximum improvement appears in the areas with maximum deviation from the optimal solution ($N_t = N_u$).

4.6 Conclusions

In this chapter some of the communication issues of the existing decentralized task assignment algorithms were raised. We discussed that, although RDTA produces conflict-free assignment with limited communication requirements, it requires that agents have the capability of relaying information from one neighbor to the other neighbors. The auction algorithms have similar limitations, where either all the agents have to be able to communicate to a centralized auctioneer or they have to be able to relay information. Implicit coordination does not require the relaying capability, but it requires a high bandwidth and was shown to be communication inefficient. A new Auction Based Task Assignment (ABTA) algorithm was developed in this chapter to overcome these limitations. The algorithm uses the ideas of both consensus and auction algorithms and creates conflict-free assignments with limited communication and without the requirement of having relaying capabilities. The algorithm was proved to converge to a conflict-free complete assignment. Simulation results were also presented to further show the communication advantages of the ABTA algorithm over the existing algorithms.

Chapter 5

Robust Filter-Embedded Task Assignment

5.1 Introduction

Unmanned Aerial Vehicles (UAVs) of the near future will require a significant high-level planning capability in order to successfully execute their missions without significant human interaction. The missions envisaged are complex, often requiring multiple heterogeneous vehicles to successfully cooperate in order to achieve the global mission objectives [22, 44, 77]. For example, in a typical cooperative search, acquisition, and track (CSAT) mission, the planning controller must keep track of all known and potential targets in a large area and use the available information to assign the team of UAVs to perform the search and track tasks as efficiently and timely as possible (see [3, 8, 11, 18, 39, 71, 77, 81] and the numerous references therein). The team of vehicles would typically be equipped with sensor payloads (e.g., video or IR cameras) that provide information that can be used to improve the overall situational awareness (SA), such as classifying and locating true targets and rejecting false ones. Unfortunately, imperfections in the sensor data and/or adversarial strategies could lead to inaccurate data (wrong target location or velocity) and false conclusions (wrong target type) that corrupt the SA. Though plagued with such uncertainties in their SA, the vehicles nonetheless must use the information at their disposal to autonomously develop a set of actions, whether through a centralized or decentralized planner [5, 48, 66].

The UAV coordination problem includes several important sub-problems: determining the team composition, performing the task assignment, and UAV trajectory optimization, which are all computationally intensive optimization problems. The task assignment problem, which is a decision making process, is typically solved using integer (or mixed-integer linear) programming techniques. The particular emphasis of this chapter is on ensuring that this key part of the planning process is robust to the uncertainty in the optimization data. As discussed, the uncertainty in this data can come from many sources, and it is well known that it can have a significant impact on the performance of an optimization-based planner. Mitigating the effect of the uncertainty in this type of optimization problem has recently been addressed by numerous researchers [12, 15, 16, 17, 46, 47, 58, 64]. For example, Ref. [12] discusses the issue of robust *feasibility* of linear programs, and Ref. [17] considers the problem of finding robust solutions to linear programs under more general norms, which extends the research of Ref. [64]. Ref. [58] introduces general robust optimization techniques that embed higher order statistics from the uncertainty model. Refs. [15, 16] develop techniques that hedge against worst-case performance loss and maintain robust feasibility in the presence of uncertainty in integer optimizations. Solutions are presented with both ellipsoidal and polyhedral uncertainty sets, and the authors show that the robust equivalent of some uncertain integer programs can be found by solving a finite number of new deterministic integer programs. Refs. [46, 47] introduce Conditional Value at Risk (CVaR) as a scenario-based resource allocation problems solved by generating a finite (but possibly large) number of realizations, and finding the assignment that minimizes the conditional expectation of performance loss.

While these papers present substantial contributions to the field of robust optimization, they do not fully address the problems associated with the online decision-making problem of interest in this thesis. In particular, most of these algorithms take a significant computation time or require generating numerous realizations of the uncertain data, which can lead to difficulties for implementation onboard the

UAVs. The goal of this chapter is to develop a real-time decision-making algorithm that both hedges against performance loss in the optimization and adapts to new information about the environment. The typical response to a change in the SA is to reassign the vehicles based on the most recent information. A potential problem, however, is that errors in the sensor measurements can lead to rapid changes in the situational awareness. Then, constant replanning by the vehicles based on this latest SA can, in the worst case, result in paths that oscillate between targets without ever reaching any of them, and in general can result in much longer mission times than necessary [28, 76]. Thus the replanning process must avoid this churning, or limit cycle, behavior to obtain good overall mission performance.

Several researchers have tackled specific aspects of combined robust planning with approaches that mitigate churning. For example, the previous work on robustness addresses the issue of sensitivity of decision-making algorithms to uncertainty in the optimization parameters, but does not consider churning. Refs. [28, 76] discuss the issue of churning, but do not explicitly account for performance robustness. Ref. [76] investigates the impact of replanning, with the objective function being a weighted sum of the mission goal and the difference between the previous plan and the current one. The problem of task reassignment due to changes in the optimization has also been addressed by Ref. [40] in their use of incremental algorithms for combinatorial auctions. Their goal is to limit the number of changes from one iteration to the next in order to ensure that the human operator can follow the changes. They propose that the perturbed optimization problem should also include a term in the objective function that penalizes changes from the nominal solution. Both Refs. [40, 76] use the plan generated prior to the current one as a reference, but they do not directly consider the impact of noise in the problem, nor do they develop techniques to mitigate its effect on the replanning.

The work in Refs. [18, 19] extended a simple robustness algorithm in the literature, producing an algorithm that is as effective as more complex techniques but significantly easier to implement, and thus is well suited for real-time implementation in the UAV problem. Section 5.2 presents a summary of this work, which provides the basis for the algorithm developed in this chapter. In Refs. [2, 3], we provided a new algorithm that accounts for changes in the SA during replanning. Our approach to mitigate churning is unique in that the coefficient weights that penalize changes in the assignment are tuned online based on the previous plan changes. In this chapter, we extend this algorithm and show that the modified algorithm demonstrates the desired filtering behavior. Further analysis is provided to show these properties. The main contribution of this chapter is combining these robust and adaptive approaches to develop a fully integrated solution to the UAV task planning problem, and discussing the interactions between the two techniques in a detailed simulation. The resulting Robust Filter Embedded Task Assignment (RFETA) algorithm is shown to be suited for real-time calculation and yields superior performance to using robustness or the Filter-Embedded Task Assignment (FETA) algorithm alone.

The chapter is organized as follows. Section 5.2 addresses robust planning for the task assignment problem and compares the performance of various algorithms. The filter-embedded task assignment algorithm (FETA) is presented in Section 5.3. The algorithm combining these two formulations, RFETA, is presented in Section 5.4, followed by numerical simulations that demonstrate the benefit of the proposed formulation in Section 5.5.

5.2 Planning Under Uncertainty

Planning under uncertainty has received a lot of recent attention, as it enables autonomous agents, such as UAVs, to make decisions while accounting for errors in their situational awareness. By including knowledge of the uncertainty in the plan optimization, these robust decision making processes are more successful on average than techniques that do not account for uncertainty. While there are many algorithms that address this important problem, there are some limitations to their use in online decision-making. For example, performance of these algorithms may require a very large number of data realizations [46, 47], or finding the solution to numerous deterministic problems [15], which may lead to computational delays in finding the robust solution that may be unacceptable for online implementation. Hence, algorithms that can embed uncertainty and plan robustly using simpler optimizations are required for real-time UAV missions. The UAV assignment problem is formally introduced in this section, and several approaches for planning under uncertainty are compared in Section 5.2.1. A computationally tractable approach is then presented in Section 5.2.2.

5.2.1 General Approaches to the UAV Assignment Problem

The UAV-target assignment problem is the allocation of vehicles to targets based on information obtained both from prior knowledge of the value and/or distance of the targets, as well as online observations acquired via possibly heterogeneous sensors. The UAVs generate plans based on this knowledge of the world (situational awareness, or SA). As their SA updates, the UAVs replan to account for the changes in the environment. In this approach, most techniques in the literature assume that at each planning step, the UAVs use the nominal value of parameters to create their plans. A simple assignment problem can be written as

$$\max_{x_k \in \mathcal{X}_k} \; \bar{c}_k^T x_k \tag{5.1}$$

where $\bar{c}_k$ is a vector of nominal scores for the N targets at time k,

$$\bar{c}_k = [\bar{c}_k(1), \bar{c}_k(2), \ldots, \bar{c}_k(N)]^T \tag{5.2}$$

and x_k is the vector of binary decision variables corresponding to the plan at time k,

$$x_k = [x_k(1), x_k(2), \ldots, x_k(N)]^T, \qquad x_k(i) \in \{0, 1\} \tag{5.3}$$

The decision variable $x_k(i) = 1$ if target i is selected in the assignment at time k, and $x_k(i) = 0$ otherwise. The set $\mathcal{X}_k$ denotes the feasible space for x_k. This space could represent general constraints such as limits on the total number of vehicles assigned to the mission. For example, if only $M < N$ vehicles could be assigned to the mission, such a constraint would require $\sum_{i=1}^{N} x_k(i) < M$.

Planning with the expected scores is reasonable if this is the only information available to the UAVs. However, it should be possible to improve the planning process if additional statistics (such as the variance, σ_k^2, of the scores) about the uncertainty in the target score information are available to the decision-making algorithm. The following section discusses the approach introduced in Ref. [18] for including this additional knowledge in a variation of the Minimum Variance Assignment algorithm developed in Ref. [74].

5.2.2 Computationally Tractable Approach to Robust Planning

The situational awareness obtained from sensor data is typically filtered using non-linear target tracking filters (e.g., particle filters or extended Kalman filters) and multi-hypothesis classifiers [7]. The results are then available in the form of mean and variance data that represent the best estimate of the target state (e.g., type, location). For example, Ref. [18] uses classical Kalman filtering equations to update the measurements of a *static* world with additional observations. The measurement model of each agent at each time step is the observation equation

$$z_{k+1|k} = H s_k + \nu_k \tag{5.4}$$

where H is a measurement matrix of the states s, which could include target location and type/score, and ν_k is assumed to be a white Gaussian noise with zero mean and covariance R_k. The observation equation is used to update the expected value of the state at time $k+1$, given all the information at time $k+1$ ($\bar{s}_{k+1|k+1}$) and uncertainty ($P_{k+1|k+1}$) using the following filtering equations

$$\begin{aligned} \bar{s}_{k+1|k+1} &= \bar{s}_{k|k} + L_{k+1}(z_{k+1|k} - \hat{z}_{k+1|k}) \\ P^{-1}_{k+1|k+1} &= P^{-1}_{k|k} + H^T R_k^{-1} H \end{aligned} \tag{5.5}$$

where $\hat{z}_{k+1|k} = E[z_{k+1|k}] = H\bar{s_k}$ and $L_{k+1} = P_{k|k}H^T(HP_{k|k}H^T + R_k)^{-1}$ is the Kalman filter gain. These state estimates can then be used to predict the scores associated with each task in the planning algorithms. In this chapter, the states used in the filters are actually the target scores themselves.

Fundamentally, these approaches all fall into the class of techniques known as robust optimization, where the requirement is to maximize the objective function while taking into account the uncertainty in the optimization parameters. This form of optimization can be interpreted as the decision-maker attempting to maximize the reward (by maximizing over the decision variables x), while the data coefficients c (which, in one interpretation, assumes that nature selects them) are assumed to belong to a set $\mathcal{C}$, from which they can take on their worst-case values. This problem can be mathematically written as

$$\min_{c\in\mathcal{C}} \max_{x\in\mathcal{X}} c_k^T x_k \tag{5.6}$$

We will refer to variations of this optimization throughout the chapter as *robust optimization*, since the goal is to maximize the objective in the presence of the uncertainty.

There are various approaches for including this uncertain information in the planning algorithms, many of which are variations of the Minimum Variance Assignment. One example is Soyster's method [74], which solves the following optimization

$$J_{S,k} = \max_{x_k\in\mathcal{X}_k} (\bar{c}_k - \sigma_k)^T x_k \tag{5.7}$$

This formulation assigns vehicles to the targets that exhibit the highest 1-σ worst-case score. In general, this approach may result in an extremely conservative policy, since it is unlikely that each target will indeed achieve its worst case score. Furthermore, unless otherwise known, it is extremely unlikely that all targets will achieve their worst case score simultaneously. Therefore, a modification to the cost function is introduced, allowing the operator to accept or reject the uncertainty, by introducing a parameter (λ) that can vary the degree of uncertainty introduced in the problem [18, 19]. The modified Soyster formulation then takes the form

$$J_{R,k} = \max_{x_k \in \mathcal{X}_k} (\bar{c}_k - \lambda \sigma_k)^T x_k \tag{5.8}$$

The scalar $\lambda \geq 0$ is a tuning parameter that reflects risk aversion (or acceptance). Note the selection of λ is critical in the success of this approach, and many ad-hoc heuristics could be developed for particular distributions, to ensure that an appropriate level of uncertainty is incorporated in the optimization. In the simplest case of Gaussian distributed target scores, a selection of $\lambda = 3$ would guarantee that the 99% percentile of the worst-case is accounted for in the optimization; similar percentiles can be derived for other target score distributions. Some insight into the selection of λ is provided later in this section.

This approach in Eq. 5.8 recovers the main attribute of the robust missions by explicitly taking into account the higher moment information contained in the variance of the score, σ^2 [19]. Note, however, that the optimization of the problem in Eq. 5.8 has essentially the same computational burden as the nominal planning approach, so it is much simpler than the optimizations in Refs. [15, 46, 47]. Two examples are given in Ref. [19] to present comparisons with more sophisticated robust optimization algorithms that show that, while simpler, the modified Soyster approach can be used to attain similar levels of performance.

The first example demonstrates numerical results of the proposed robust optimization for the case of an assignment with uncertain data, and compares them to the nominal formulation (where the target scores are replaced with the expected target scores). The simulations confirm the expectation that the robust optimization results in a lower *but more certain* mission score. The modified Soyster algorithm is also compared to a slightly more sophisticated robust algorithm known as the Conditional Value at Risk (CVaR). CVaR's performance and determination of the robust assignment relies heavily on the total number of data realizations. That is, the user provides the optimization algorithm with numerous instances of the uncertain data, and the CVaR algorithm finds the optimal robust assignment given the data. The example shows that the Modified Soyster algorithm achieves identical performance of the CVaR algorithm.

Sensitivity to λ: A set of Monte Carlo simulations is used here to investigate the effect of λ on the robust algorithm in Eq. 5.8. A scenario with 30 UAVs and 100 targets with nominal values drawn randomly from a uniform distribution in the range $[10, 20]$ is used. Each target value is assumed to be uncertain, which is modeled as a normal probability distribution with zero mean and a random standard deviation, which itself is drawn from a uniform distribution in the range $\Sigma = [0, 3]$. Then 5000 realizations of this scenario are considered, and for each realization, the actual target values were taken to be the nominal value added to a random normal uncertainty scaled by a particular realization of the standard deviation drawn from the set Σ. Each problem is then solved using Eq. 5.8 with different values of λ. This process is repeated for 100 different scenarios.

Figure 5.1 shows the mean and standard deviation of the objective values for these scenarios. Note that in this figure $\lambda = 0$ recovers the nominal solution. The results show that increasing the value of λ reduces both the average and standard deviation of the objective value, which are consistent with the previous observations that the robust optimization results in a lower but more certain mission score. The plot also shows that the decrease in the standard deviation as λ is increased from $0 \rightarrow 1$ is more significant than the increase from $1 \rightarrow 3$. Analysis such as this, based on the uncertainty model for a particular problem, can be used to investigate the trade-off between the desired performance and confidence levels and then to choose the λ parameter.

5.3 Filter-Embedded Task Assignment

The problem described in the previous section has tacitly assumed that the optimization is generally performed only when substantial changes in the environment have been observed (*e.g.*, as a result of UAV loss or target re-classification). In reality,

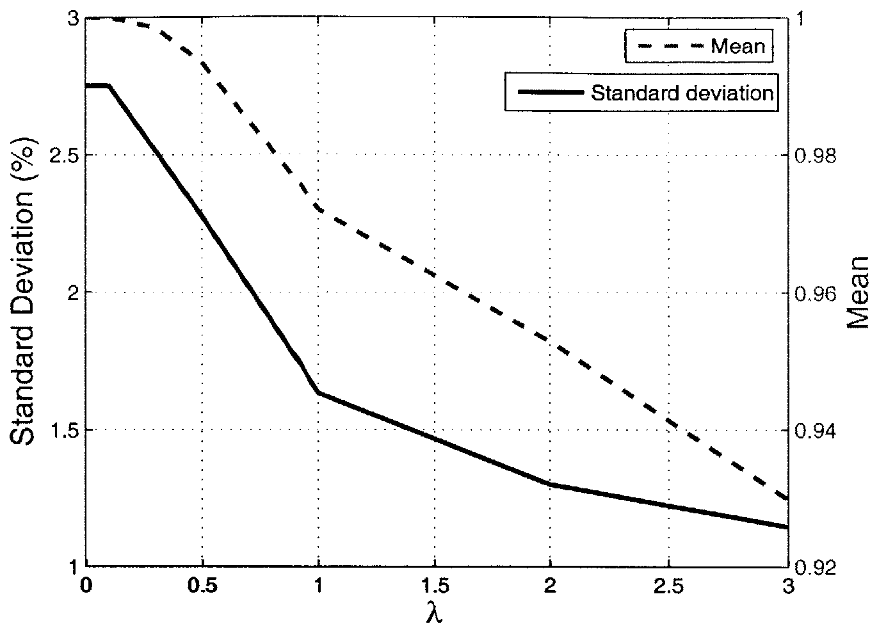


Figure 5.1: Sensitivity of the robust algorithm to parameter λ. Here $\lambda = 0$ is the nominal case.

these information updates are continuously occurring throughout the mission due to changes in the vehicle's SA. The typical response to a change in the SA is to reassign the vehicles based on the most recent information. The problem of task reassignment due to changes in the optimization has been addressed by Kastner et al. [40] in their use of incremental algorithms for combinatorial auctions. Their goal is to limit the number of changes from one iteration to the next in order to ensure that the human operator can follow the changes. They propose that the perturbed optimization problem should also include a term in the objective function that penalizes changes from the nominal solution.

The work of Tierno and Khalak [76] also investigates the impact of replanning, with the objective function being a weighted sum of the mission goal and the difference between the previous plan and the current one. Both of these formulations rely on the plan generated prior to the current one as a reference, but they do not directly consider the impact of noise in the problem, nor do they develop techniques to mitigate its effect on the replanning.

The objective of this section is to develop a modified formulation of the task assignment problem that mitigates the effect of noise in the SA on the solution. The approach taken here is to perform the reassignments at the rate the information is updated, which enables the planner to react immediately to any significant changes that occur in the environment. Furthermore, rather than simply limiting the rate of change of the plan, this new approach embeds a more sophisticated filtering operation in the task assignment algorithm.

We demonstrate that this modified formulation can be interpreted as a noise rejection algorithm that reduces the effect of the high frequency noise on the planner. A key feature of this filter-embedded task assignment algorithm is that the coefficients of the filter can be tuned online using the past information. Simulations are then presented to demonstrate the effectiveness of this algorithm.

5.3.1 Problem Statement

Consider the target assignment problem expressed in Section 5.2.1. The targets have nominal value $\bar{c}_k$, and the objective, Eq. 5.1, is to select the optimal set of targets to visit subject to a set of constraints. From a practical standpoint, these target values are uncertain and are likely to change throughout the course of the mission; real-time task assignment algorithms must respond appropriately to these changes in information.

The most straightforward technique is to immediately react to this new information by reassigning the targets. In a deterministic sense, replanning proves to be beneficial since the parameters in the optimization are perfectly known; in a stochastic sense replanning may not be beneficial. For example, since the observations are corrupted by sensor noise, the key issue is that replanning *immediately* to this new information results in a task assignment control process with short dwell times (analogous to having a high bandwidth controller) that could simply track the sensor noise. From the perspective of a human operator, continuous reassignment of the vehicles in the fleet may lead to increased human errors, especially if this effect is due primarily to the sensing noise. Furthermore, since the optimization is continuously responding to new information, and in the worst case responding to noise, it is likely that the assignment will change in every time step, ultimately resulting in a churning effect in the assignment, as observed in Ref. [27].

A simple example of churning is shown in Figure 5.2, where one vehicle is assigned to visit the target with the highest value. The original assignment of the vehicle (starting on the left) is to visit the bottom right target. At the next time step, due to simulated sensing noise, the assignment for the vehicle is switched to the top right target. The vehicle changes direction towards that target, and then the assignment switches once again. The switching throughout the course of the mission is an exaggerated behavior of the churning phenomenon. In fact, it can be seen that as the mission progresses, the vehicle is still alternating between the targets to visit, and never converges to a fixed assignment. While this is a simplified example of churning, it captures the notion that sensing noise alone could cause a vehicle to switch assignments throughout a mission, reducing the overall performance.

Likely missions may involve multiple vehicles, each with unique information and independent sensors and noise sources. It might be quite difficult to identify and correct the churning behavior in a large fleet of UAVs. The subsequent sections present methods of modifying the general task assignment problem to automatically avoid this phenomenon. The following sections develop a filter for the assignment problem that mitigates the effect of noise in the information vector (optimization parameters) and can be tuned to capture different noise frequencies.

5.3.2 Assignment With Filtering: Formulation

In this section the concept of binary filtering is defined and then incorporated into the assignment problem in order to mitigate the effect of environmental noise or uncertainties. Consider the simple task assignment problem introduced in Eq. 5.1,

$$\max_{x_k \in \mathcal{X}_k} \; c_k^T x_k \tag{5.9}$$

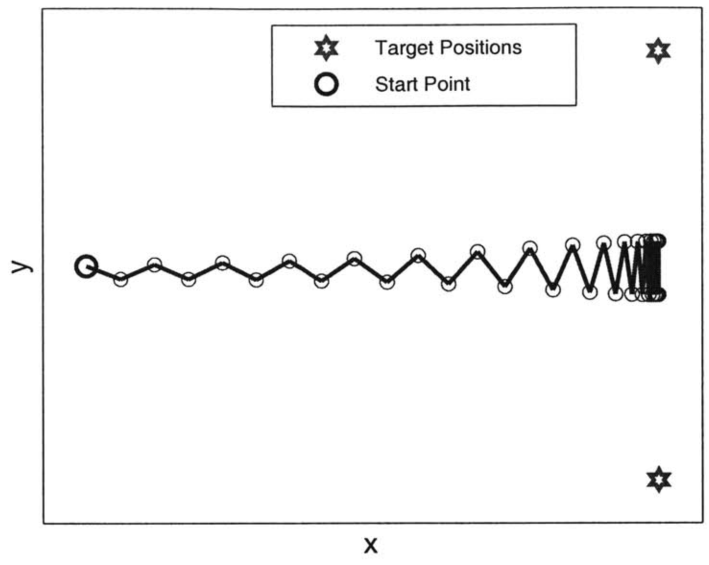


Figure 5.2: Trajectory of a UAV under a worst-case churning effect in a simple assignment problem.

The solution to this problem at any time step k is a binary vector x_k of size N, where N is the number of targets and $x_k(i)$ is 1 if target i is selected and 0 otherwise. With each replanning at each time step k, the values of the elements of this vector change, which results in changes in the assignment.

A binary filter of size r is defined as a system whose binary output changes at most with the rate of once every r time steps. Figures 5.3 shows the input and output to two binary scalar filters with lengths $r = 3$, $r = 5$. As illustrated in Figure 5.3-top, the input is a signal with the maximum rate of change (change at each time step). The output is a binary signal with the maximum rate of one change every 3 steps (Figure 5.3-middle) and every 5 steps (Figure 5.3-bottom). These figures show the filter for a single binary value (i.e. only $x_k(i)$), but the same idea can be extended to a binary vector (i.e. x_k).

Now with the simple assignment problem of Eq. 5.9, the idea is to replan for the same system in each time step and suppress the effect of parameter noise on the output of the system (generated plan). If the above problem (Eq. 5.9) is solved in each iteration, the optimization parameters will be directly impacted by the noise

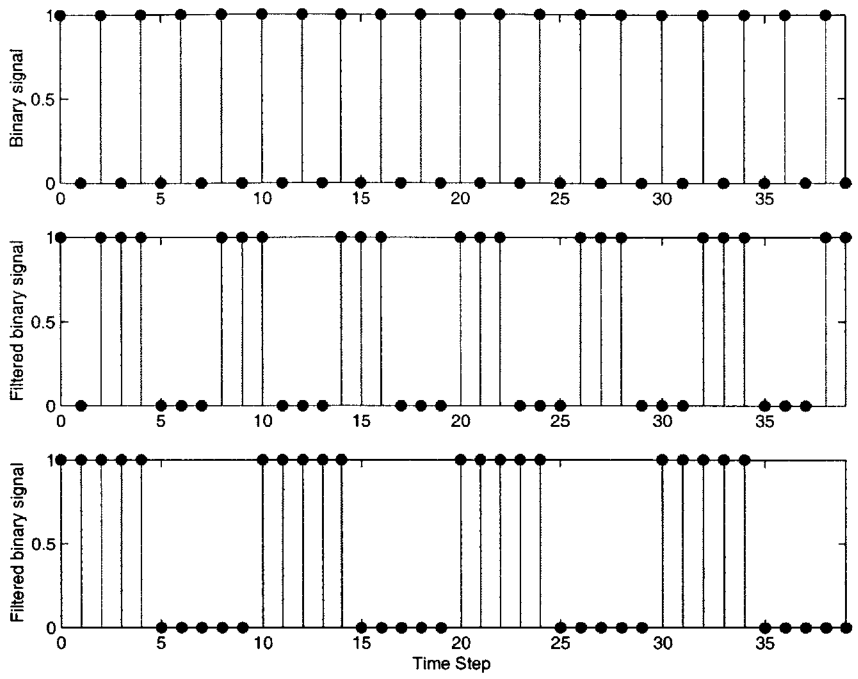


Figure 5.3: (top): Input to the binary filter; (middle): Filtered signal for the case $r = 3$; (bottom): Filtered signal for the case $r = 5$.

and the assignment can be completely different at each step. Variations in the plan that are due to environmental noise or parameter uncertainties, as well as rapid or large changes in the plan that are either impossible or inefficient to track, are usually unwanted. To avoid these types of variations in the plan over time, a binary filter is integrated into the algorithm to limit the rate of changes in the assignment problem and suppress the effect of noise.

An assignment with filtering can be written in the following form,

$$y_k = f(c_k, x_k, x_{k-1}, \dots, x_{k-q}, y_{k-1}, y_{k-2}, \dots, y_{k-r}) \tag{5.10}$$

where c_k, x_k, and y_k are the input to the system, the current nominal (unfiltered) solution, and the filtered solution respectively. Further, $x_{k-1}, \dots, x_{k-q}$ are the previous unfiltered plans and $y_{k-1}, \dots, y_{k-r}$ are the previous filtered plans (previous outputs of the system).

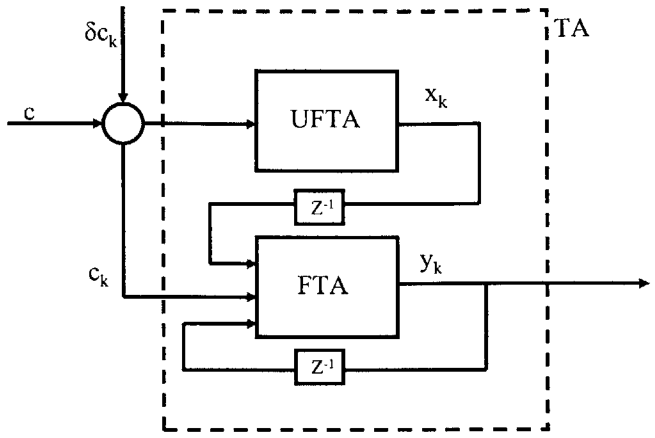


Figure 5.4: Block diagram representation of assignment with filtering.

It should be noted that c_k here is the noisy input to the system:

$$c_k = \tilde{c}_k + \delta c_k \tag{5.11}$$

where δc_k is the additive noise to the actual value of the parameters, $\tilde{c}_k$. Examples of this type of noise are the effect of decoy and sensor noise.

At each time step, both filtered and unfiltered plans are generated. Figure 5.4 gives a block diagram representation of assignment with filtering. Here, FTA and $UFTA$ represent the filtered and unfiltered task assignment algorithms, respectively, TA represents the overall task assignment algorithm, and Z^{-1} represents a bank of unit delays. $UFTA$ is essentially the algorithm that solves the nominal problem in Eq. 5.9. In this section, we develop an algorithm for the FTA box.

A comprehensive form of this filter can be implemented to obtain very general filtering properties. The general form of the FETA algorithm is

$$\begin{aligned}
\max_{y_k} \quad & c_k^T y_k - \sum_{l=1}^{r} (\beta_k^l)^T (y_k \oplus y_{k-l}) && (5.12)\\
\text{s.t.} \quad & y_k \in \mathcal{X}_k \\
& \beta_k^l = \sum_{j=l}^{q} b_j^l \delta x_j^l && (5.13)\\
& \delta x_j^l = x_{k-j} \oplus x_{k-j+l} && (5.14)
\end{aligned}$$

where x_j is the unfiltered plan at time step j, which is calculated in the $UFTA$ box in Figure 5.4. These values are then used in Eqs. 5.13, 5.14 to calculate β, which is then used in the optimization represented by Eqs. 5.12, 5.13. These calculations construct the FTA box in Figure 5.4.

The second term in the objective function (Eq. 5.12) is the penalty associated with changes that are made in the current plan compared to the previous plans. In contrast to existing algorithms [40], the penalty coefficients β in this approach are not constant. In particular, these coefficients are calculated dynamically using previous plans to selectively reject the changes associated with the high frequency noise. Note that the high frequency noise here can be either real noise, such as sensor noise, or any changes in the environment which either UAVs cannot respond to or it is not efficient for them to respond to. The value r in the second part of the objective function is the bandwidth of the binary filter incorporated into this formulation, which is shown explicitly in Section 5.3.3. The coefficient vector β_k^l specifies the penalty on the change of each element in the assignment vector. Each element of these coefficient vectors is a weighted summation of previous changes in the nominal plan.

Since the (unfiltered) plan will track environmental noise, measuring changes in the unfiltered plan provides a good metric to identify and suppress such disturbances. This is implemented in Eqs. 5.13 and 5.14. Here, q is the length of the window of old information that is included in calculating the coefficients. The coefficients b_j^l determine the impact of changes in previous plans on the current plan. It is usually the case that more recent changes better represent the current state of the environment and therefore they should be be given more weight when calculating the penalty coefficients. A good candidate for b_j^l that satisfies this criterion is

$$b_j^l = \frac{b^l}{2^j} \tag{5.15}$$

where b^l is a constant that can be set based on the problem. Eq. 5.15 generates smaller weights for larger j, and thus attenuates the effect of the changes that happened in the far past (larger j), compared to the more recent changes in the plan.

It should be noted here that the calculation of β (Eqs. 5.13, 5.14) is done outside and enters the optimization as input. This helps to keep the complexity of the optimization problem at the same level as the nominal formulation.

5.3.3 Assignment With Filtering: Analysis

This section analyzes the filtering properties of the formulation in Eq. 5.12 and the effect of the design parameters on the shape and the bandwidth of this filter. The discussion in Section 5.3.2 argued that r implicitly defines the bandwidth of the filter in the assignment formulation. The analysis presented here analyzes that relationship in more detail.

Consider a problem with one UAV and two targets. The UAV can go to only one target, and the objective is to maximize the score which is the value of the selected target. The true value of the targets is chosen to be 10 but was perturbed with noise in the range of $[-1, 1]$. At each time step the UAV updates its perception of target values and replans based on the new values. To analyze the filtering properties of the formulation, the frequency of the noise is changed from 1 to 0.125 and the response to each noise frequency is recorded. Note again that frequency of 1 means changes that happen every time step and is the highest possible frequency, and the frequency of 0.125 represents changes that happen at most every 8 time steps. The r was set at different values in the algorithm, and the behavior of the filter is recorded for each.

The results are shown in Figure 5.5. The noisy input is applied to both filtered and unfiltered assignment problems and the number of changes (the level of churning) is calculated for both cases. The values in the y-axis show the ratio of the number of changes in the filtered case to the one of the unfiltered case. Note that the values here are the average of 100 Monte Carlo simulations and each simulation runs for 100 time steps. The value of 1 at a particular frequency means that all of the changes with that frequency content would pass through the filter, while a value of 0 means that all of the changes would be suppressed. As expected, the results clearly show the low-pass filtering effect of the algorithm. The effect of the parameter r for tuning the effective bandwidth of the filter is also apparent. In particular, as r increases from 1

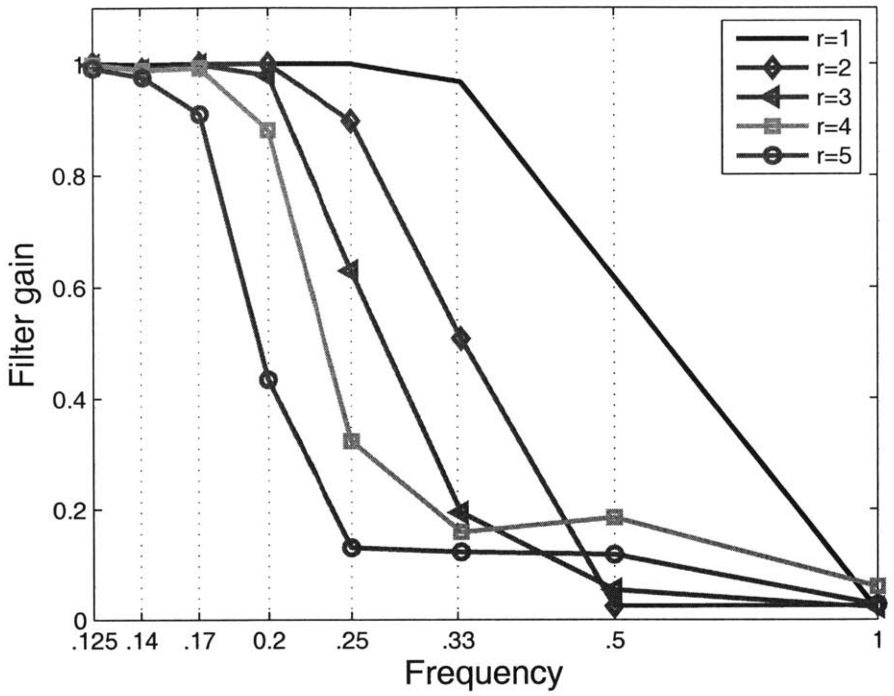


Figure 5.5: Frequency response of the filter for different values of r (bandwidth).

to 5, the effective bandwidth reduces from approximately 1 to $1/5 = 0.2$.

Another important parameter in the formulation of Eq. 5.12 is β, which is the coefficient on the penalty terms and is a weighted sum of previous changes in the nominal plans. To investigate the impact of β or b_j^l's on the shape of the filter, rewrite Eq. 5.13 as

$$\beta_k^l = \sum_{j=l}^{q} b_j^l \delta x_j^l = b_0 \sum_{j=l}^{q} \hat{b}_j^l \delta x_j^l \tag{5.16}$$

Figure 5.6 shows the results with $r = 3$, the $\hat{b}_j^l$ fixed, while the value of b_0 is changed from 0.5 to 1.1. It should be noted that there is no pure analytical way for selecting the $\hat{b}_j^l$ values and they are chosen by iterating over the resulting filter shape for a given noise level. This figure clearly shows that increasing the value of b_0 causes the filter to increase the level of attenuation at the desired cut-off frequency ($1/r = 0.33$) and leads to some further attenuation at lower frequencies.

In summary, Figures 5.5 and 5.6 indicate that r is a more effective means of

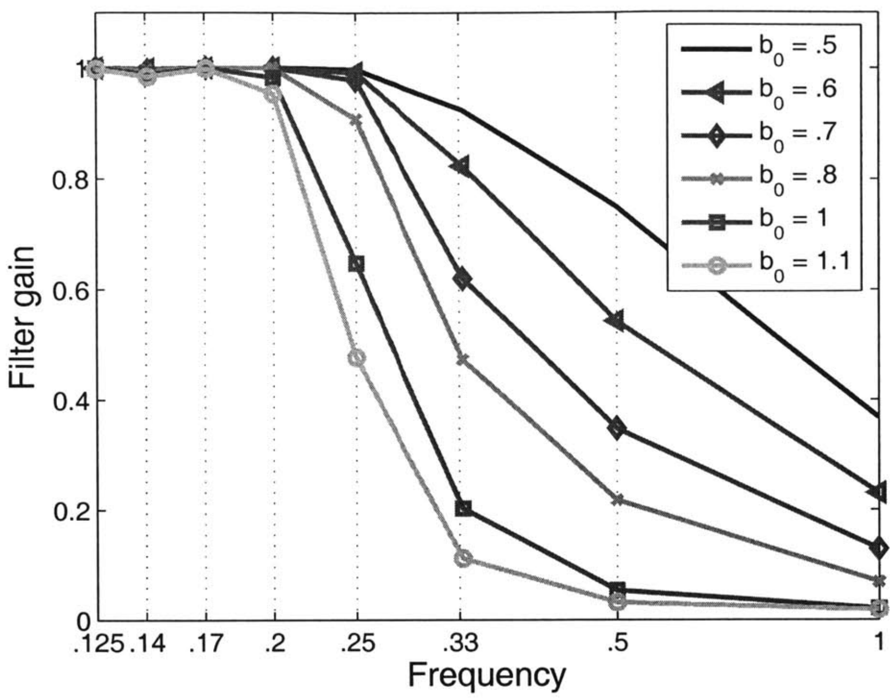


Figure 5.6: Frequency response of the filter for different penalty coefficients, b_0.

changing the cut-off frequency than changing b_0, and that for a given r, b_0 can be used to tune the level of attenuation achieved at the specified cut-off frequency.

To further show the behavior of the filtering formulation (Eqs. 5.12 - 5.13), it is simplified here to the following optimization problem, which rejects noises with frequency 1 (this is the highest frequency, meaning that the signal can change in each time step):

$$\max \quad c_k^T y_k - (\beta_k^1)^T (y_k \oplus y_{k-1}) \tag{5.17}$$

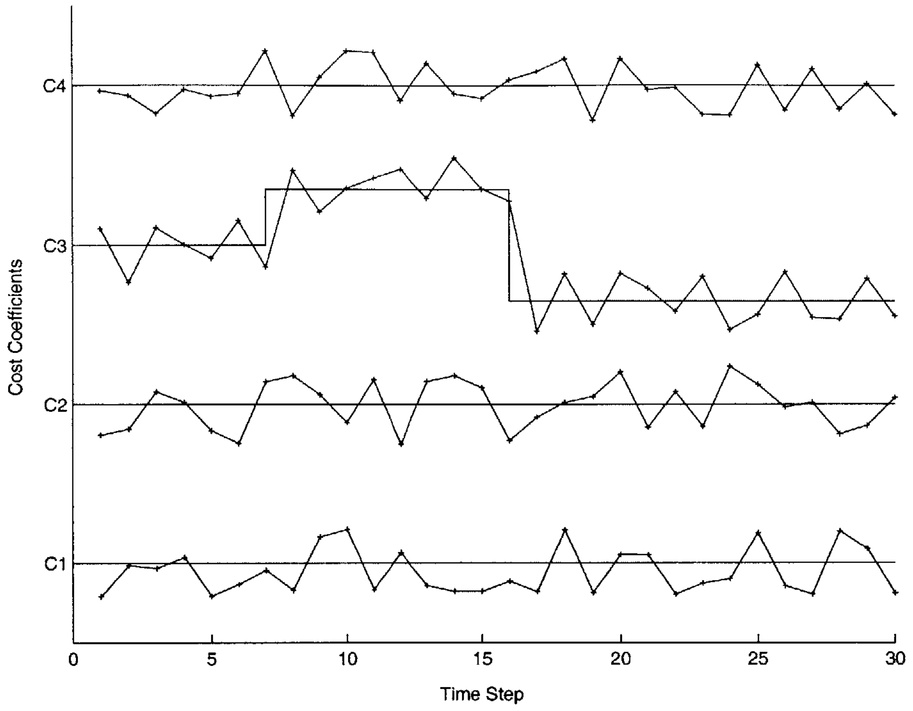
$$\begin{aligned} \text{s.t.} \quad & y_k \in \mathcal{Y}_k \\ & \beta_k^1 = \sum_{j=1}^{q} b_j^1 \delta x_j^1 \\ & \delta x_j^1 = x_{k-j} \oplus x_{k-j+1} \end{aligned} \tag{5.18}$$

This formulation is used here for a simple, but general, assignment problem and compares the results of the unfiltered and filtered formulations. Similar to previous examples, the objective of this problem is to choose M of N existing targets ($M < N$) in order to maximize the total value of the selected mission. Each target has a value associated with it that is affected by noise

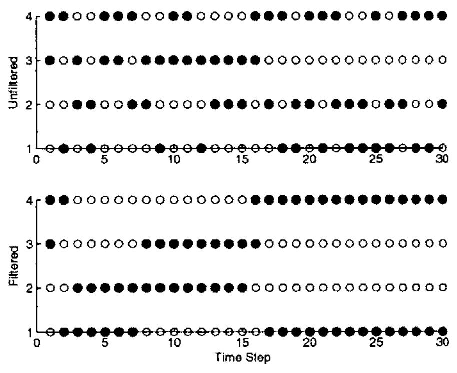
$$c_k = \tilde{c}_k + \delta c_k \tag{5.19}$$

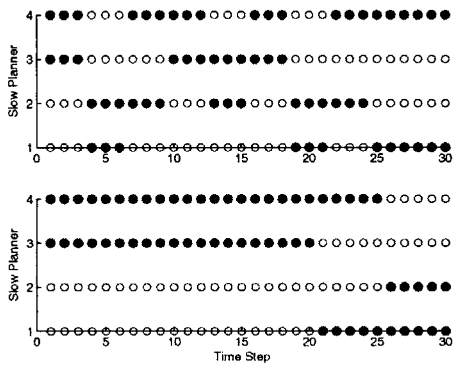
where $\tilde{c}_k$ is the actual target value at time k and δc_k is the noise added to this value (top plot of Figure 5.7). The nominal value for all targets is set to 5 and the noise is uniformly distributed in the interval $[-0.5, 0.5]$. The parameter b^l in Eq. 5.15 is set to 0.6. Solving this problem for $N = 4$, $M = 2$ for 30 iterations (time steps) results in 30 different plans that are directly affected by the noise, and Figure 5.7 shows the result of this simulation. In these figures, $\bullet$ represents 1 and $\circ$ represents 0. Thus, targets 1 and 2 are selected in assignment 1, and targets 1 and 4 are selected in assignment 2. Note that as the costs change in the top plot from one time period to the next, the unfiltered plan changes as well. However, the filtered solution (bottom plot) remains unchanged for much longer periods of time. Thus it is clear that the unfiltered solution tracks the noise in the cost coefficients to a much greater extent than the filtered plan. To demonstrate that the filtered plan is only rejecting the noise, the coefficient c_2 is increased by 0.7 at time step 7, and then decreased by 1.4 at time step 16. The results in Figure 5.7(b) show that the filtered plans follow these lower frequency changes. As noted previously, one approach to mitigate the impact of the sensing noise is to slow down the planning rate. Figure 5.7(c) shows the results of using this method for two cases. In the top figure, the replanning is done every 3 time steps, and it is clear that churning still exists; it is just at a lower rate. In the bottom figure the replanning is done every 5 time steps. In this case, the churning effect is not apparent, but the planner's response to the changes in the signal is slow (at $k = 16$, c_2 drops in value, but the planner responds to it at $k = 26$).

Another important issue is the lag that the binary filter implies on the assignment. It is shown in the figure that the input signal changes at times $k = 7,\ 16$ and the



(a) Noisy cost coefficients.





(b) (top): Plans with no filter; (bottom): Filtered plan.

(c) (top): Planning every 3 time steps (no filter); (bottom): Planning every 5 time steps (no filter).

Figure 5.7: Comparing the results of a filtered and an unfiltered plan (• represents 1 and ◦ represents 0).

filtered assignment responds to these changes at $k = 8,\ 17$. This clearly shows a lag of one time step, which is small compared to the lag introduced in the case where assignment is performed every 5 time steps in order to remove the churning (Figure 5.7(c)).

5.4 Robust FETA

This section presents the new Robust Filter Embedded Task Assignment (RFETA) algorithm, which synergistically combines the robust and FETA techniques introduced in Sections 5.2 and 5.3. Recall that the goal of the robust planning algorithm is to hedge against the uncertainty in the optimization data, and therefore takes into account the nominal scores and uncertainty of the targets, whereas FETA is designed to mitigate the effect of churning, while replanning with new information. While both algorithms successfully achieve their objectives, they individually have some limitations.

The robust planning algorithm does not explicitly consider the effects of new information on the plan changes. It only considers the effects of uncertainty on the overall mission planning but not on previous plan changes. Thus it is susceptible to churning. FETA corrects the vehicle churning problem, but does not directly take into account uncertainty in the problem. Therefore, if the measurement updates do not occur frequently, FETA is susceptible to the same problems as nominal planning.

A new algorithm that combines the strong features of each algorithm and improves upon their individual weaknesses is therefore of great importance. There are numerous ways to combine the concepts of robust planning with FETA (e.g., Refs. [15, 47]), but we use the approach suggested in Section 5.2, in which the nominal objective is replaced by an uncertainty-discounted objective; namely, replacing $\bar{c}_k$ with $\bar{c}_k - \lambda\sigma_k$ will make the optimization cognizant of the impact of the uncertainty. Based on this discussion, we propose the new optimization,

$$\max_{x_k \in \mathcal{X}_k} \ (\bar{c}_k - \lambda\sigma_k)^T x_k - \beta_k^T (x_k \oplus x_{k-1}) \tag{5.20}$$

This is a direct modification of the FETA algorithm that penalizes the scores of the individual targets by a fraction of their uncertainty, which we refer to as the RFETA optimization. As previously introduced, the parameter λ is a tuning parameter that varies the level of uncertainty in the optimization. A more general form of this RFETA optimization replaces the scalar, λ, with a time-varying vector, $\underline{\lambda}_k$,

$$\max_{x_k \in \mathcal{X}_k} \ (\bar{c}_k - \underline{\lambda}_k \cdot \sigma_k)^T x_k - \beta_k^T (x_k \oplus x_{k-1}) \tag{5.21}$$

where $\underline{\lambda}_k \cdot \sigma_k$ represents the element by element multiplication. This generalization is useful when the desired confidence levels of each target are not equal (for example, if the targets have different overall mission values); furthermore, the level of uncertainty, governed by the choice of λ, may increase or decrease as other targets are discovered or added to the mission list.

5.5 Numerical Simulations

This section presents the results of several numerical simulations that were performed to compare the various algorithms introduced in this chapter. The first set of examples compares RFETA with the FETA and robust algorithms graphically. The second set of simulations uses Monte Carlo simulations to analyze the different optimizations and compare their average performance as well as time required to achieve these levels of performance.

5.5.1 Graphical Comparisons

Consider a simulation of a small environment with three targets and 1 UAV. Each target is assumed to have a nominal score of 10, but there are different initial variances for these scores. As shown in Table 5.1, target 1 has the lowest standard deviation, while target 3 has the highest standard deviation.

These simulations are implemented using Receding Horizon Task Assignment (RHTA), a task allocation algorithm introduced in Ref. [2]. This algorithm takes

Table 5.1: Target parameters for Section 5.5.1

Target #	Score	σ_0	Initial Distance from UAV
1	10	2	7.62
2	10	3.5	7.62
3	10	6	6.2

into account the distance in the overall cost evaluation of each target, discounting the target score by a quantity directly proportional to the time taken by the UAV to reach that target, and groups together the various combinations of targets into *petals*. In these simulations, the petals are of size 1 because it was assumed that the UAV can only visit one target. At each time step, the UAV obtains a measurement as described in Section 5.3. The results of these missions using four different algorithms (Nominal, Robust, FETA, and RFETA) are shown in Figure 5.8. The nominal plan is shown in the top left figure. Ignoring the uncertainty, the nominal plan sends the UAV to the closest target, which also has the highest uncertainty, while churning. Due to the updated information from the estimator, the UAV constantly replans and alternates between the different targets. Without any penalty on changing the plans, the UAV does what is optimal at *each time step*, thereby changing targets due to measurement noise. In the top right corner, the churning problem is mitigated by using FETA, but the UAV is still sent to the closest target that has the highest uncertainty.

The robust plan is shown in the figure in the bottom left corner. At each time step, the measurement is used to update both the estimate and the covariance ($\bar{c}_k$ and P_k). The planner uses the new updated values for these quantities to update the information and replan. Note that the robust plan sends the UAV to the targets with the lowest uncertainty, but churns, since there is no penalty incurred for this. In the bottom right corner, the robust plan using Eq. 5.20 (RFETA) is implemented. The planner uses the new updated values for these quantities to update the information, but the cost function contains a penalty term on the replanning, thereby reducing the churning significantly. Note that the robust FETA sends the UAV to the targets with the lowest uncertainty, with no churning. These simulations show single realizations

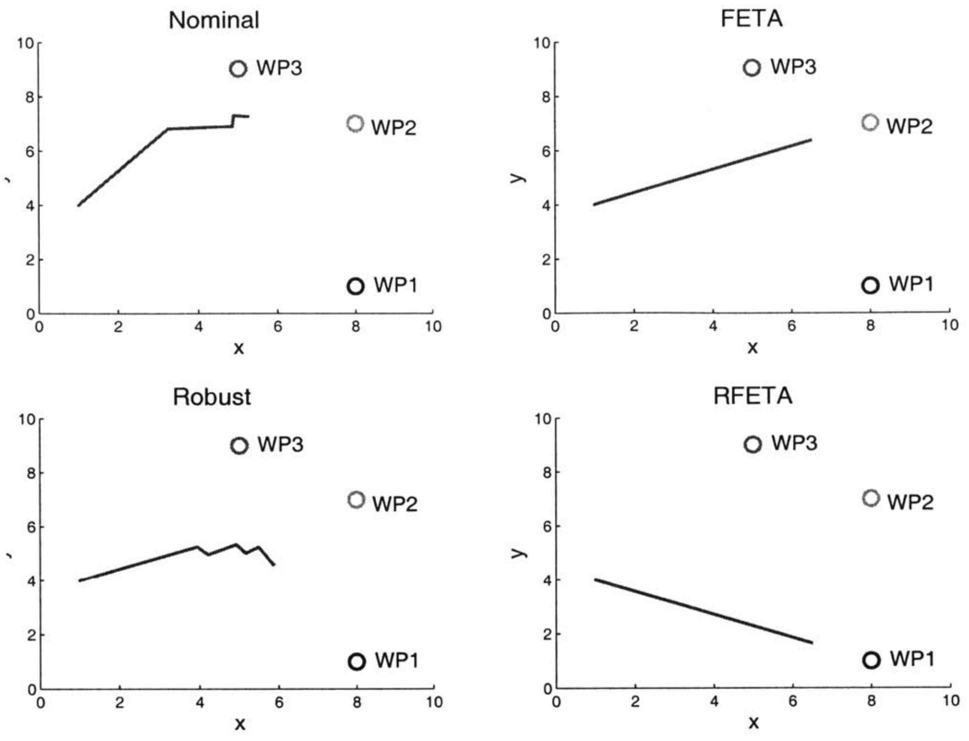


Figure 5.8: Top left: Nominal planning; Top right: Nominal planning with FETA; Bottom Left: Robust planning; Bottom Right: Robust planning with FETA.

of the RFETA implementation; the next subsection discusses large-scale Monte Carlo simulations that were conducted to study the performance of RFETA in greater detail.

5.5.2 Monte Carlo Simulations

Large-scale Monte Carlo simulation methods are used to evaluate and compare the performance of the Robust FETA algorithm with the other three algorithms: Nominal, Robust and FETA. The simulation setup is similar to the previous example. In these simulations the randomly generated parameters are the positions of the targets and the score associated with each target. The x- and y- target positions are created using a uniform distribution in $[0, 10]$. The scores are chosen randomly with normal distribution, with a mean of 10 and standard deviation of 2, 3.5, and 6 for the three targets, respectively. The implementation of each simulation is done similarly to the previous example. The result shown here is created by running 100 Monte Carlo simulations, each one created by 100 random runs.

Figure 5.9 shows the result of these simulations. The horizontal axis is the time elapsed since the start of the simulation, while the vertical axis shows the average accumulated score. Figure 5.10 focuses on the results after 20 time units and includes $1-\sigma$ error bars to better compare the values. The results of Figure 5.10 indicate the following:

1. The Robust algorithm has a similar initial performance response to the nominal approach, but yields a higher final accumulated value, which is consistent with the system exhibiting some churning but making better overall planning decisions.

2. The basic FETA algorithm yields a faster convergence to the final value than the nominal approach, but essentially the same final value, which is consistent with reducing the level of churning in the system but not necessarily improving the quality of the planning decisions.

3. The RFETA algorithm exhibits the fastest convergence, the highest final value, and the lowest standard deviation in the final values, all of which are consistent with the new approach reducing the extent of the churning and making good overall planning decisions.

To further compare the RFETA with the Nominal algorithm, Figure 5.11 shows the histogram of the mission completion times for the two algorithms respectively. These results are summarized in Table 5.2. The results show that the mission completion time for the RFETA algorithm is lower on average and it also has a smaller standard deviation. We can see that RFETA reaches 90% of the final value much faster than the Nominal algorithm, and its average and standard deviations of the mission completion time are also much lower than those resulting from the Nominal algorithm. Note the similarities in mean and standard deviation of the finishing times between the RFETA and FETA algorithms, underscoring the importance of

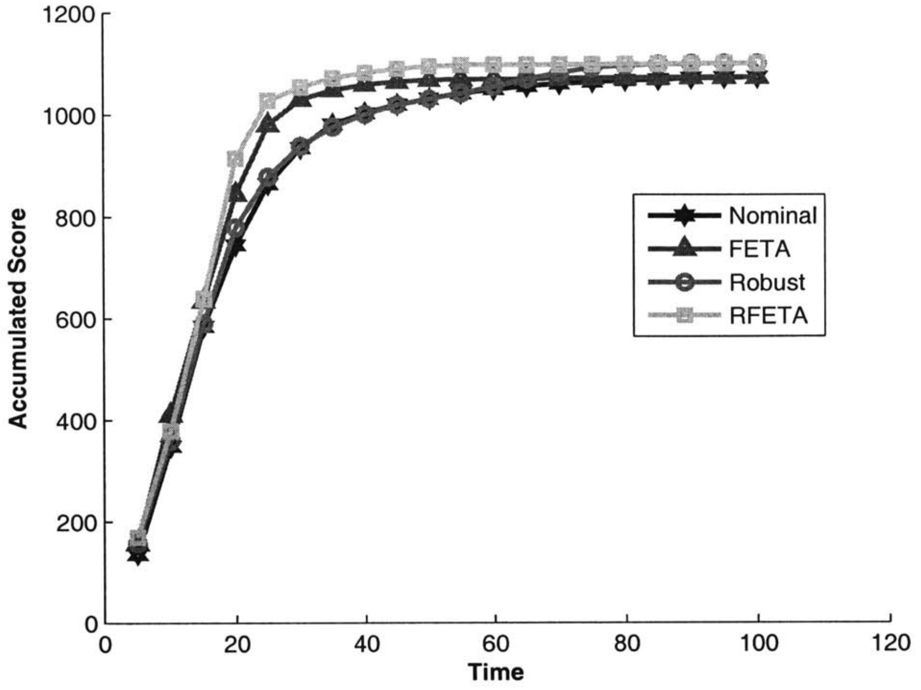


Figure 5.9: Result of the Monte Carlo simulation for the four algorithms.

Table 5.2: Comparison of convergence rate of the four algorithms.

Algorithm	Nominal	Robust	FETA	RFETA
Avg. of mission completion time	17.77	18.41	14.12	14.15
Std. of mission completion time	14.45	15.40	8.72	8.36
Time to 90% of final value	40	40	25	20

the FETA component in reducing the churning behavior that would otherwise plague both the Nominal and Robust algorithms.

5.5.3 Impact of Measurement Update Rate

The previous simulations used a measurement update at each time step in the mission, and the vehicles replanned at each time step with a new piece of information. It is, however, unlikely that the UAVs will be able to update their information at such rates, and it is thus desirable to investigate the effects of slower information updates

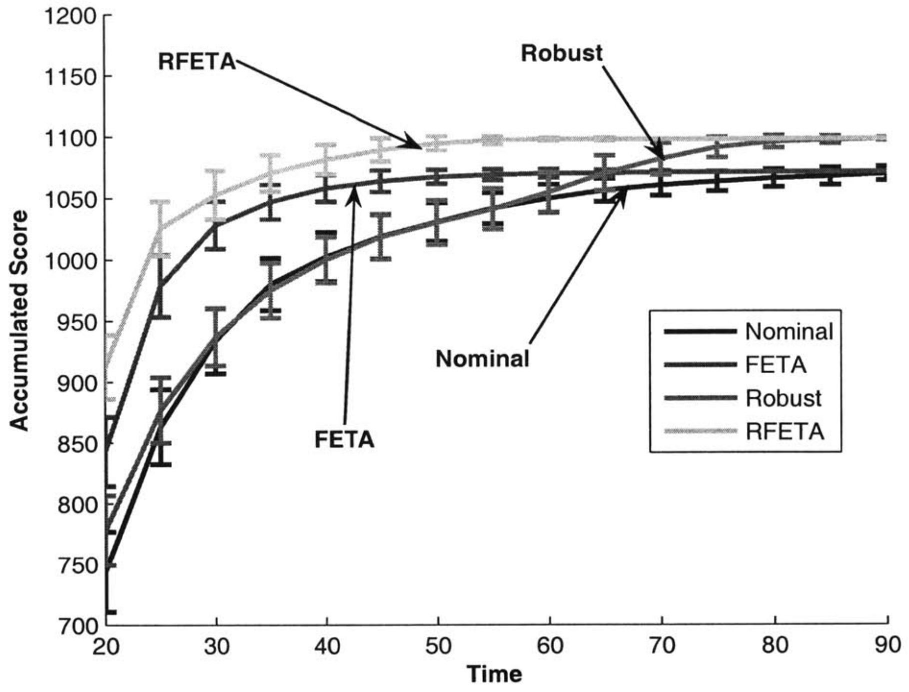


Figure 5.10: Comparing the accumulated score and its confidence level for the four algorithms.

on the algorithm performance, where the *information update* is the rate that the UAVs make observations on the score of the targets, and update them using, for example, a Kalman filter. Note that even though this information update rate is reduced in the following simulations, the replanning still occurs at each measurement update. The key motivation behind these simulations is that by receiving information updates less frequently, RFETA will generate robust plans over longer mission segments, while FETA or the Nominal algorithm will not generate plans that are hedged against the uncertainty.

In order to investigate the effect of intermittent information updates, the simulations in section 5.5.2 are performed using information update every 2, 5, 7 and 10 time steps. That is, in the first set of simulations measurements are received every $\Delta T = 2$, while in the last set of simulations they are received every $\Delta T = 10$ steps. Target scores and initial distances are selected randomly.

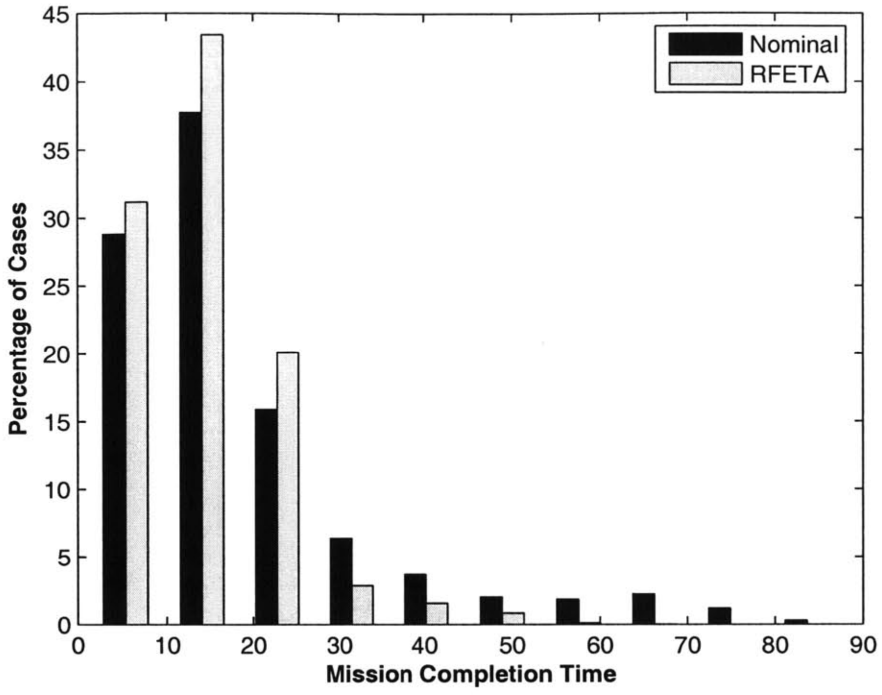


Figure 5.11: Histogram comparing the mission completion time of the Nominal and RFETA algorithms for the Monte Carlo simulations of subsection 5.5.2.

Table 5.3 shows a comparison of the finishing times for the Nominal algorithm and RFETA. For an information update of $\Delta T = 5$, the Nominal algorithm completes on average in 19.1 steps, while RFETA finishes in 16.04 steps. The longer finishing time for the Nominal algorithm demonstrates that with slower updates, the Nominal algorithm exhibits churning, while RFETA mitigates this problem, leading to a reduction in the overall mission time by approximately 15%. By updating the information every $\Delta T = 10$ time steps, the Nominal algorithm's finishing time is 16.8 steps compared to 13.9 time steps for RFETA, an overall improvement of approximately 17%. The table shows that RFETA both reduces the mission completion time and its standard deviation. Thus, there is a much higher probability of obtaining longer finishing times greater than 25 steps for the Nominal algorithm than RFETA.

By updating every 2 time steps, the Nominal algorithm churns frequently in response to the sensing noise. At each measurement update, there is no penalty on

Table 5.3: Comparison of finishing times for different algorithms and measurement updates.

Algorithm	Nominal ($\Delta T = 5$)	Nominal ($\Delta T = 10$)	RFETA ($\Delta T = 5$)	RFETA ($\Delta T = 10$)
Avg. of Finishing Time	19.1	16.8	16.04	13.9
Std. of Finishing Time	4.45	4.77	3.54	2.30

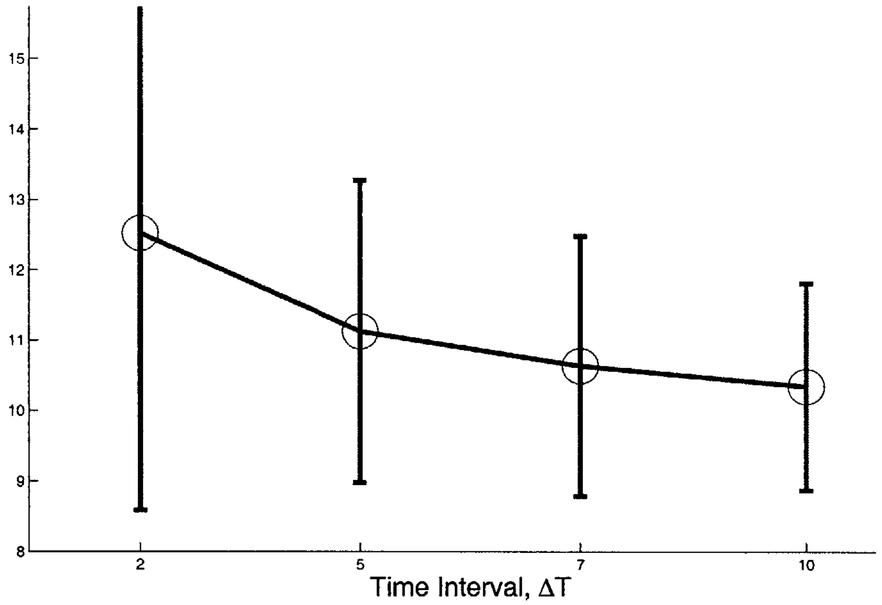


Figure 5.12: Impact on Measurement Update Time Interval (ΔT) on Overall Score. As the time between information updates increases, RFETA performs identically to robust algorithms. A slight decrease in performance occurs with increased certainty of this performance objective.

changing the plans and there is no explicit inclusion of the uncertainty in the optimization. For an increased interval of $\Delta T = 10$ steps, the algorithm assigns the UAV to the target with the highest realization of the score, disregarding the uncertainty (or variance) in the score of that target. RFETA makes up for these limitations by explicitly taking the uncertainty into account in the optimization. Furthermore, by embedding the FETA component, there is a penalty on changing the plans too frequently, thereby minimizing the finishing times of the mission.

Changes in the *mission score* of RFETA are also seen by evaluating the performance objectives of RFETA for different ΔT values. As ΔT increases and the information updates occur less frequently, RFETA relies on the robustness component to ensure that the vehicle is assigned to target scores hedged against the uncertainty. The simulations are evaluated by varying ΔT and maintaining the discount factor of $\lambda = 1$ for all the $N = 100$ simulations, and results are shown in Figure 5.12. This figure plots the time interval ΔT between measurements versus the *time discounted score* that the vehicles obtain at the final time step in the mission. These time discounted scores are calculated as $S_j = c_{N_F}(j)\varsigma^{T_F(j)}$ where $c_{N_F}(j)$ is the realization of the target score and $T_F(j)$ is the finishing time for simulation j. The time discounted score is the expected value of the simulation scores,

$$S = \frac{1}{N}\sum_j c_{N_F}(j)\varsigma^{T_F(j)} \tag{5.22}$$

Note that as ΔT increases, the information updates are less frequent; hence, the covariance in the scores of the targets is not updated as frequently, and is reduced less rapidly than with more frequent measurements. Note also that the standard deviations of this score decrease as ΔT increases. Recall that the robust planning techniques generally exhibit a loss in performance in exchange for higher certainty in the performance. This figure shows that the robust component of RFETA exchanges this small performance loss for an increase in the *certainty* of obtaining this performance objective.

5.6 Conclusions

This chapter has presented new results in the design of real-time missions that are robust to uncertainty in the environment. The most recent literature has introduced algorithms that are either made robust to the uncertainty in the optimization, or that adapt to the information in the environment, but little or no literature has addressed the *combined* problem of robustness and adaptiveness.

This chapter discussed three key aspects of the robust planning problem. First, it was demonstrated that an extended version of a simple robustness algorithm in the literature is as effective as more complex techniques. However, since it only has the same computational complexity as the Nominal algorithm, it is significantly easier to implement than the other techniques available, and thus is well-suited for real-time implementation in the UAV problem. We also provided a new algorithm which accounts for changes in the situational awareness during replanning. Our approach to mitigate churning is unique in that the coefficient weights that penalize changes in the assignment are tuned online based on previous plan changes. Finally, we combined these robust and adaptive approaches to develop a fully integrated solution to the UAV task planning problem, and discussed the interactions between the two techniques in a detailed simulation. The resulting RFETA is shown to be well-suited for real-time calculation and yields superior performance to using robustness or FETA alone.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Unbiased Kalman Consensus (Chapter 2)

The performance of the Kalman Consensus Algorithm was investigated for a team of agents with static data. It was shown that, although this algorithm converges for the general case of strongly connected communication networks, it can result in a biased estimate when the outflow of the agents is not equal. An extension to this algorithm was then presented which was shown in simulations to converge to the true centralized estimate for different network structures. This algorithm was further proved to converge to an unbiased estimate for both static and dynamic communication networks.

Robust Decentralized Task Assignment (Chapter 3)

The success of the implicit coordination approach, in which the central assignment algorithm is replicated on each UAV, strongly depends on the assumption that all UAVs have the same situational awareness, and the examples showed that this consensus is necessary, but potentially time consuming. This chapter presented an extension of the basic implicit coordination approach that assumes some degree of data synchronization, but adds a second planning step based on shared planning data. The resulting Robust Decentralized Task Assignment method uses these shared candidate plans to overcome any residual disparity in the information at the end of the (possibly truncated) consensus stage. The simulations demonstrated the advantages of this new method in generating feasible plans that reduced the conflicts in the assignments and improved the performance compared to implicit coordination.

Further results demonstrated the effect of communication on the performance of assignment in different stages of the planning. The performance of the RDTA algorithm for different communication network topologies was also analyzed and it was shown that the communication during the planning phase introduced in this new technique is crucial to achieve high performance. This is especially true for sparse communication networks, where the slow convergence of the information consensus results in decentralized activity planning based on inconsistent data. To analyze the sensitivity of the overall performance to the candidate plan set, four selection algorithms were presented. A comparison of the performance for these algorithms clearly show the importance of accounting for the potential actions of other UAVs in the selection process. A modification of the original candidate plan selection algorithm was also presented to further improve the overall performance by increasing the robustness to inconsistencies in the information across the team.

Auction-Based Task Assignment (Chapter 4)

In this chapter some of the communication issues of the existing decentralized task assignment algorithms were raised. It was discussed that although RDTA produces conflict-free assignment with limited communication requirements, it requires that agents have the capability of relaying information from one neighbor to the other neighbors. The nominal auction algorithms have similar limitations, where either all the agents have to be able to communicate to a centralized auctioneer or they have to be able to relay information. Implicit coordination does not require the relaying capability but requires a high bandwidth, and was shown to be communication inefficient. A new Auction-Based Task Assignment (ABTA) algorithm was developed in this chapter to overcome these limitations. The algorithm uses the ideas of both consensus and auction algorithms and creates conflict-free assignment with limited communication and without the requirement of having relaying capabilities. The algorithm was proved to converge to a conflict-free complete assignment. Simulation results were also presented to further show the communication advantages of the proposed algorithm over the existing algorithms.

Robust Filter-Embedded Task Assignment (Chapter 5)

This chapter presented new results in the design of real-time missions that are robust to the uncertainty in the environment. The most recent literature has introduced algorithms that are either made robust to the uncertainty in the optimization, or that adaptively adapt to the information in the environment, but little or no literature has addressed the *combined* problem of robustness and adaptiveness.

This chapter discussed three key aspects of the robust planning problem. First we demonstrated that an extended version of a simple robustness algorithm in the literature is as effective as more complex techniques. However, since it only has the same computational complexity as the Nominal algorithm, it is significantly easier to implement than the other techniques available, and thus is well suited for real-time implementation in the UAV problem. We also provided a new algorithm for accounting for changes in the situational awareness during replanning. Our approach to mitigate churning is unique in that the coefficient weights that penalize changes in the assignment are tuned online based on previous plan changes. We combined these robust and adaptive approaches to develop a fully integrated solution to the UAV task planning problem, and discussed the interactions between the two techniques in a detailed simulation. The resulting Robust Filter Embedded Task Assignment (RFETA) is shown to provide an algorithm that is well suited for real-time calculation and yields superior performance to using robustness or FETA alone.

6.2 Future Work

This thesis addressed some of the issues of the decentralized planning and proposed solutions to these problems. But there are still problems that need to be answered and improvements that could be made, as discussed in the following.

Chapter 2 proposed a Kalman Consensus Algorithm that converges to the desired centralized solution for the general communication structures. The assumption in this work is that each agent knows how many neighbors it has and it uses this information in the formulation. It is also implicitly assumed that once a message is sent to an agent, the receiver receives the message. This assumption, however, might not always be satisfied. In some cases, there is a probability associated with receiving the message by the receiver agent. This probability is usually a function of distance and environment. Future research should investigate the possibility of achieving the desired solution in the Kalman Consensus Algorithm for this type of communication network.

Chapters 3 and 4 dealt with the decentralized task assignment algorithms. Different existing decentralized algorithms were analyzed and their advantages and disadvantages were discussed. Two new decentralized task assignment algorithms were developed to addresses the issues associated with the existing methods. Each of these algorithms performs better than others for a certain situation with certain communication and computation capabilities and a certain objective. A good planning system is an adaptive system that can achieve the best performance for any situation by using the best possible algorithm. A future direction is to combine the different decentralized algorithms and create an adaptive planning system that at any time can adapt to the objectives of the mission and limitations of the environment and provides the best possible plan. The performance of the resulting system degrades gracefully with an increase of the limitations posed by the environment/vehicles.

The algorithms proposed in this thesis were tested in simulations and their performance was confirmed by theoretical proofs and simulation results. But future works should implement these algorithms in real hardware and test the validity of the algorithms in real environment, which is essential when dealing with complex problems. Although computer simulations can create preliminary results to validate the algorithm, many issues only arise when the algorithm is implemented in real hardware.

An immediate future work for the RFETA algorithm developed in Chapter 5 will be to further study the effects of intermittent measurements on the performance of RFETA and highlight the benefits of this approach over using robust planning and/or FETA alone. Furthermore, developing distributed implementations of this algorithm is of great practical interest for large-scale teams of UAVs.

Bibliography

[1] M. Alighanbari, Y. Kuwata, and J. P. How, "Coordination and Control of Multiple UAVs with Timing Constraints and Loitering," Proceedings of the IEEE *American Control Conference,* Denver, Colorado, June 2003, pp. 5311-5316.

[2] M. Alighanbari, "Task Assignment Algorithms for Teams of UAVs in Dynamic Environments," *S. M. Thesis, MIT,* 2004.

[3] M. Alighanbari, L. Bertuccelli and J. How, "Filter-Embedded UAV Task Assignment Algorithms for Dynamic Environments," Proceedings of the AIAA *Guidance, Navigation and Control Conference,* Providence, Rhode Island, Aug. 2004, AIAA-2004-5251.

[4] M. Alighanbari and J. How, "Decentralized Task Assignment for Unmanned Aerial Vehicles," Proceedings of the IEEE *European Control Conference and Conference on Decision and Control,* Seville, Spain, Dec. 2005, pp. 5668-5673.

[5] M. Alighanbari and J. P. How, "Robust Decentralized Task Assignment for Cooperative UAVs," Proceedings of the AIAA *Guidance, Navigation and Control Conference,* Keystone, Colorado, Aug. 2006, AIAA-2006-6454.

[6] M. Alighanbari and J. P. How, "An Unbiased Kalman Consensus Algorithm," Proceedings of the IEEE *American Control Conference,* Minneapolis, Minnesota, June 2006, pp. 3519-3524.

[7] Y. Bar-Shalom, T. Kirubarajan, and X. R. Li, Estimation with applications to tracking and navigation: Theory Algorithms and Software, John Wiley & Sons, New York, 2004.

[8] R. W. Beard and V. Stepanyan, "Synchronization of Information in Distributed Multiple Vehicle Coordinated Control," Proceedings of the IEEE *Conference on Decision and Control,* Maui, Hawaii, Dec. 2003, pp. 2029-2034.

[9] R. Beard and T. McLain, "Multiple UAV Cooperative Search Under Collision Avoidance and Limited Range Communication Constraints," Proceedings of the IEEE *Conference on Decision and Control,* Maui, Hawaii, Dec. 2003, pp. 25-30.

[10] J. S. Bellingham, M. J. Tillerson, A. G. Richards and J. P. How, "Multi-Task Assignment and Path Planning for Cooperating UAVs," *Conference on Coordination, Control and Optimization,* Nov. 2001.

[11] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-Task Allocation and Path Planning for Cooperative UAVs," *Cooperative Control: Models, Applications, and Algorithms,* Editors: S. Butenko, R. Murphey, and P. M. Pardalos, Kluwer Academic Publishers, 2003, pp. 23-41.

[12] A. Ben-Tal and A. Nemirovski, "Robust Solutions of Uncertain Linear Programs," *Operations Research Letters,* 1999, vol. 25, pp. 1-13.

[13] D. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Prentice Hall, 1989.

[14] D. Bertsekas, "The auction algorithm for assignment and other network flow problems," *LIDS Technical Report*, 1989.

[15] D. Bertsimas and M. Sim, "Robust Discrete Optimization and Network Flows," *Mathematical Programming,* Series B, 2003, vol. 98, pp. 49-71.

[16] D. Bertsimas, K. Katarajan, and C. Teo, "Probabilistic Combinatorial Optimizations: Moments, Semidefinite Programming, and Asymptotic Bounds," *SIAM Journal of Optimization,* 2005, vol. 15, pp. 185-209.

[17] D. Bertsimas, D. Pachamanova, and M. Sim, "Robust linear optimization under general norms," *Operations Research Letters,* 2004, vol. 32, pp. 510-516.

[18] L. Bertuccelli, M. Alighabari, and J. P. How, "Robust Planning for Coupled Cooperative UAV Missions," Proceedings of the IEEE *Conference on Decision and Control,* Paradise Island, Bahamas, Dec. 2004, pp. 2917-2922.

[19] L. Bertuccelli, "Robust Planning for Heterogeneous UAVs in Uncertain Environments," *S. M. Thesis, MIT,* 2004.

[20] V. Blondel, J. Hendricks, A. Olshevsky, and J. Tsitsiklis, "Convergence in Multiagent Coordination, Consensus, and Flocking," Proceedings of the IEEE *European Control Conference and Conference on Decision and Control,* Seville, Spain, Dec. 2005, pp. 2996-3000.

[21] J. Boskovic, R. Prasanth and R. Mehra, "An Autonomous Hierarchical Control Architecture for Unmanned Aerial Vehicles," Proceedings of the AIAA *Guidance, Navigation, and Control Conference,* Providence, Rhode Island, Aug. 2004, AIAA-2002-4468.

[22] P. Chandler, S. Rasmussen, and M. Pachter, "UAV Cooperative Path Planning," Proceedings of the AIAA *Guidance, Navigation and Control Conference,* Denver, Colorado, Aug. 2000, AIAA-2000-4370.

[23] P. Chandler and M. Pachter, "Hierarchical Control for Autonomous Teams," Proceedings of the AIAA *Guidance, Navigation, and Control Conference,* Montreal, Canada, Aug. 2001, AIAA-2001-4149.

[24] P. Chandler, M. Pachter, D. Swaroop, J. Fowler, J. K. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard, "Complexity in UAV Cooperative Control," Proceedings of the IEEE *American Control Conference,* Anchorage, Alaska, May 2002, pp. 1831-1836.

[25] P. Chandler, "Decentralized Control for an Autonomous Team," Proceedings of the 2nd AIAA *Unmanned Unlimited Conference,* San Diego, California, Sept. 2003, AIAA-2003-6571.

[26] P. R. Chandler, M. Pachter, S. R. Rasmussen and C. Shumacher, "Distributed Control for Multiple UAVs with Strongly Coupled Tasks," Proceedings of the AIAA *Guidance, Navigation and Control Conference,* Austin, Texas, Aug. 2003, AIAA-2003-5799.

[27] J. Curtis and R. Murphey, "Simultaneous Area Search and Task Assignment for a Team of Cooperative Agents," Proceedings of the AIAA *Guidance, Navigation and Control Conference,* Austin, Texas, Aug. 2003, AIAA-2003-5584.

[28] J. Curtis, "Cooperative Instability: The Churning Effect and its Causes," *Recent Developments in Cooperative Control and Optimization,* Editors: S. Butenko, R. Murphey, and P. M. Pardalos, Kluwer Academic Publishers, 2004, pp. 105-116.

[29] J. Feddema, C. Lewis and D.A. Shoenwald, "Decentralized Control of Cooperative Robotic Vehicles: Theory and Application," IEEE Transactions on *Robotics and Automation,* Oct. 2002, vol. 18, pp. 852-864.

[30] M. Flint, T. Khovanovay, and M. L. Curry, "Decentralized Control Using Global Optimization," Proceedings of the AIAA *Infotech Conference and Exhibit,* Rohnert Park, California, May 2007, AIAA-2007-2906.

[31] A. Gibbons, Algorithmic Graph Theory, Cambridge University Press, 1985.

[32] D. Goldfarb and G. Iyengar "Robust portfolio selection problems," *Mathematics Of Operations Research,* 2003, vol. 28, pp. 1-38.

[33] B. Grocholsky, H. Durrant-Whyte, and P. Gibbens, "An Information-Theoretic Approach to Decentralized Control of Multiple Autonomous Flight Vehicles," *Sensor Fusion and Decentralized Control in Robotic Systems III,* Oct. 2000, vol. 4196, pp. 348-359.

[34] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte, "Information-Theoretic Coordinated Control of Multiple Sensor Platforms," Proceedings of the IEEE *International Conference on Robotics and Automation,* Taipei, Taiwan, May 2003, pp. 1521-1526.

[35] R. A. Horn and C. R. Johnson, Matrix Analysis, Cambridge University Press, 1985.

[36] T. Ito, N. Fukuta, T. Shintani, and K. Sycara, "BiddingBot: A Multiagent Support System for Cooperative Bidding in Multiple Auctions," Proceedings of the IEEE *Fourth International Conference on MultiAgent Systems,* Boston, Massachusetts, July 2000, pp. 399-400.

[37] G. Iyengar, "Robust Dynamic Programming," *Mathematics Of Operations Research,* May 2005, vol. 30, pp. 257-280.

[38] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules," IEEE Transactions on *Automatic Control,* June 2003, vol. 48, pp. 988-1001.

[39] M. Jun and D. Jeffcoat, "Convergence Properties of Continuous-Time Markov Chains with Application to Target Search," Proceedings of the IEEE *American Control Conference,* Portland, Oregon, June 2005, pp. 662-667.

[40] R. Kastner, C. Hsieh, M. Potkonjak, and M. Sarrafzadeh, "On the Sensitivity of Incremental Algorithms for Combinatorial Auctions," Proceedings of the 4th IEEE *International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS),* Newport Beach, California, June 2002, pp. 81-88.

[41] E. King, M. Alighanbari, Y. Kuwata, and J. How, "Coordination and Control Experiments on a Multi-Vehicle Testbed, " Proceedings of the IEEE *American Control Conference,* Boston, Massachusetts, June 2004, pp. 5315-5320.

[42] Y. Kim and K. Hong, "Decentralized Information Filter in Federated Form," Proceedings of the IEEE *SICE 2003 Annual Conference,* Fukui, Japan, Aug. 2003, pp. 2176-2181.

[43] Y. Kim and K. Hong, "Decentralized Sigma-Point Information Filters for Target Tracking in Collaborative Sensor Networks," IEEE Transactions on *Acoustics, Speech, and Signal Processing,* Aug. 2005, vol. 53, pp. 2997-3009.

[44] A. Kott, "Advanced Technology Concepts for Command and Control," *Xlibris Corporation,* Feb. 2004.

[45] P. Kouvelis and G. Yu, <u>Robust Discrete Optimization and Its Applications,</u> Kluwer Academic Publishers, 2003.

[46] P. Krokhmal, R. Murphey, P. Pardalos, S. Uryasev, and G. Zrazhevsky, "Robust Decision Making: Addressing Uncertainties in Distributions," In *Cooperative Control: Models, Applications and Algorithms*, Kluwer Academic Publishers, 2003, pp. 165-185.

[47] P. Krokhmal, J. Palmquist, and S. Uryasev, "Portfolio Optimization with Conditional Value-At-Risk Objective and Constraints," *The Journal of Risk,* 2002, vol. 4, no. 2.

[48] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How, "Decentralized Robust Receding Horizon Control for Multi-Vehicle Guidance," Proceedings of the IEEE *American Control Conference,* Minneapolis, Minnesota, June 2006, pp. 2047-2052.

[49] A. M. Kwasnica, J. O. Ledyard, D. Porter, and C. DeMartini, "A New and Improved Design for Multi-Object Iterative Auctions," *Journal of Management Science,* Mar. 2005, vol. 51, pp. 419-434.

[50] G. Laporte and F. Semet, "Classical Heuristics for the Capacitated VRP," In *The Vehicle Routing Problem,* edited by P. Toth and D. Vigo, SIAM, Philadelphia, 2002.

[51] C. Lin and U. Wen, "Sensitivity Analysis of the Optimal Assignment," *European Journal of Operational Research,* 2003, vol. 149, pp. 35-46.

[52] Y. Jin, A. Minai, and M. Polycarpou, "Cooperative Real-Time Search and Task Allocation in UAV Teams," Proceedings of the IEEE *Conference on Decision and Control,* Maui, Hawaii, Dec. 2003, pp. 7-12.

[53] T. Lemaire, R. Alami and S. Lacroix, "A Distributed Task Allocation Scheme in Multi-UAV Context," Proceedings of the IEEE *International Conference on Robotics and Automation,* New Orleans, Louisiana, Apr. 2004, pp. 3622-3627.

[54] D. Li and J. Cruz, "A Robust Hierarchical Approach to Multi-Stage Task Allocation Under Uncertainty," Proceedings of the IEEE *European Control Conference and Conference on Decision and Control,* Seville, Spain, Dec. 2005, pp. 3375-3380.

[55] W. McEneaney and B. Fitzpatrick, "Control for UAV Operations Under Imperfect Information," Proceedings of the First AIAA *UAV Symposium,* Portsmouth, Virginia, May. 2002, AIAA-2002-3418.

[56] G. Minkler, Theory and Applications of Kalman Filtering, Magellan Book Company, 1990.

[57] M. Moser, D. Jokanovic, and N. Shiratori, "An Algorithm for the Multidimensional Multiple-Choice Knapsack Problem," IEICE Transaction on *Fundamentals,* Mar. 1997, vol. E80-A, pp. 582-589.

[58] J. Mulvey, R. Vanderbei, and S. Zenios, "Robust optimization of large-scale systems," *Operations Research,* 1995, vol. 43, pp. 264-281.

[59] R. Olfati-Saber and R. M. Murray, "Consensus Problems in Network of Agents With Switching Topology and Time-Delay," IEEE Transaction on *Automatic Control,* Sept. 2004, vol. 49, pp. 1520-1533.

[60] R. Olfati-Saber, "Distributed Kalman Filter with Embedded Consensus Filter," Proceedings of the IEEE *European Control Conference and Conference on Decision and Control,* Seville, Spain, Dec. 2005, pp. 8179-8184.

[61] R. Olfati-Saber, "Consensus Filters for Sensor Networks and Distributed Sensor Fusion," Proceedings of the IEEE *European Control Conference and Conference on Decision and Control,* Seville, Spain, Dec. 2005, p. 6698-6703.

[62] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," Proceedings of the *IEEE* , Jan. 2007, vol. 95, pp. 215-233.

[63] C. L. Ortiz, R. Vincent and B. Morisset, "Task Inference and Distributed Task Management in the Centibots Robotic Systems," Proceedings of the *International Conference on Autonomous Agents and Multiagent Systems,* Utrecht, The Netherlands, July 2005, pp. 860-867.

[64] D. Pachamanova, "A Robust Optimization Approach to Finance," *Ph.D. Thesis, MIT Operations Research Center,* 2002.

[65] D. C. Parkes and L. H. Unger, "Iterative Combinatorial Auctions: Theory and Practice,", Proceedings of the AAAI *17^{th} National Conference on Artificial Intelligence,* Austin, Texas, July 2000, pp. 74-81.

[66] M. Polycarpou, Y. Yang, and K. Passino, "A Cooperative Search Framework for Distributed Agents," Proceedings of the IEEE *International Symposium on Intelligent Control,* Mexico City, Mexico, Sept. 2001, pp. 1-6.

[67] W. Ren and R. Beard, "Consensus of Information Under Dynamically Changing Interaction Topologies," Proceedings of the IEEE *American Control Conference,* Boston, Massachusetts, June 2004, pp. 4939-4944.

[68] W. Ren, R. Beard, and D. Kingston, "Multi-Agent Kalman Consensus with Relative Uncertainty," Proceedings of the IEEE *American Control Conference,* Portland, Oregon, June 2005, pp. 1865-1870.

[69] W. Ren, R. Beard, and E. Atkins, "A Survey of Consensus Problems in Multi-Agent Coordination," Proceedings of the IEEE *American Control Conference,* Portland, Oregon, June 2005, pp. 1859-1864.

[70] R. Rockafellar and S. Uryasev, "Optimization of Conditional Value at Risk," Available online at http://www.ise.ufl.edu/uryasev/cvar.pdf, last accessed March 2007.

[71] T. Schouwenaars, E. Feron, and J. P. How, "Safe Receding Horizon Path Planning for Autonomous Vehicles," Proceedings of the *40th Allerton Conference on Communication, Control, and Computation,* Monticello, Illinois, Oct. 2002.

[72] C. Schumacher and P. Chandler, "Task Allocation for Wide Area Search Munitions," Proceedings of the IEEE *American Control Conference,* Anchorage, Alaska, June 2002, pp. 1917-1922.

[73] E. Seneta, Non-negative Matrices, John Wiley & Sons, New York, 1973.

[74] A. Soyster, "Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming, " *Operations Research,* 1973, pp. 1154-1157.

[75] D. M. Stipanovic, G. Inalhan, R. Teo and C. Tomlin, "Decentralized Overlapping Control of a Formation of Unmanned Aerial Vehicles," Proceedings of the IEEE *Conference on Decision and Control,* Las Vegas, Nevada, Dec. 2002, pp. 2829-2835.

[76] J. Tierno and A. Khalak, "Frequency Domain Control Synthesis for Time-Critical Planning." Proceedings of the IEE *European Control Conference,* Cambridge, UK, Sept. 2003.

[77] Unmanned Aircraft Systems Roadmap 2005–2030. Office of Secretary of Defense, Aug. 2005.

[78] R. S. Varga, Geršgorin and His Circles, Springer, 2004.

[79] J. Wolfowitz, "Product of Indecomposable, Aperiodic Stochastic Matrices," Proceedings of the *American Mathematical Society,* 1963, vol. 15, pp. 733-736.

[80] Y. Yang, A. Minai and M. polycarpou, "Decentralized Cooperative Search by Networked UAVs in an Uncertain Environment," Proceedings of the IEEE *American Control Conference,* Boston, Massachusetts, June 2004, pp. 5558-5563.

[81] Y. Yang, A. Minai, and M. Polycarpou, "Evidential Map-Building Approaches for Multi-UAV Cooperative Search," Proceedings of the IEEE *American Control Conference,* Portland, Oregon, June 2005, pp. 116-121.

[82] S. Zhu, D. Li, S. Wang, "Risk Control Over Bankruptcy in Dynamic Portfolio Selection: A Generalized Mean-Variance Formulation," IEEE Transactions on *Automatic Control,* vol. 49, pp. 447-457.

[illegible]