Task Assignment Algorithms for Teams of UAVs in Dynamic Environments

by

Mehdi Alighanbari

M.S., Electrical Engineering, NC A&T State University, 2001
B.S., Electrical Engineering, Sharif University of Technology, 1999

Submitted to the Department of Aeronautics and Astronautics
and the Alfred P. Sloan School of Management
in partial fulfillment of the requirements for the degrees of

Master of Science in Aeronautics and Astronautics
and
Master of Science in Operations Research
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2004

Author .....
Department of Aeronautics and Astronautics
and Operations Research Center
May 14, 2004

Certified by .....
Jonathan P. How
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Eric Feron
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Edward M. Greitzer
Professor of Aeronautics and Astronautics
Chair, Departmental Committee on Graduate Studies

Accepted by .....
John N. Tsitsiklis
Professor of Electrical Engineering & Computer Science
Codirector, Operations Research Center

Task Assignment Algorithms for Teams of UAVs in Dynamic Environments

by

Mehdi Alighanbari

Submitted to the Department of Aeronautics and Astronautics
and
Alfred P. Sloan School of Management
on May 14, 2004, in partial fulfillment of the
requirements for the degrees of
Master of Science in Aeronautics and Astronautics
and
Master of Science in Operations Research

Abstract

For many vehicles, obstacles, and targets, coordination of a fleet of Unmanned Aerial Vehicles (UAVs) is a very complicated optimization problem, and the computation time typically increases very rapidly with the problem size. Previous research proposed an approach to decompose this large problem into task assignment and trajectory problems, while capturing key features of the coupling between them. This enabled the control architecture to solve an assignment problem first to determine a sequence of waypoints for each vehicle to visit, and then concentrate on designing paths to visit these pre-assigned waypoints. Although this approach greatly simplifies the problem, the task assignment optimization was still too slow for real-time UAV operations. This thesis presents a new approach to the task assignment problem that is much better suited for replanning in a dynamic battlefield. The approach, called the *Receding Horizon Task Assignment* (RHTA) algorithm, is shown to achieve near-optimal performance with computational times that are feasible for real-time implementation. Further, this thesis extends the RHTA algorithm to account for the risk, noise, and uncertainty typically associated with the UAV environment. This work also provides new insights on the distinction between UAV *coordination* and *cooperation*. The benefits of these improvements to the UAV task assignment algorithms are demonstrated in several simulations and on two hardware platforms.

Thesis Supervisor: Jonathan P. How
Title: Associate Professor

Thesis Supervisor: Eric Feron
Title: Associate Professor

Acknowledgments

In carrying out the research that went into this Masters thesis, 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 my advisors, Professor Jonathan How and Professor Eric Feron for directing and guiding me through this research. Next, I would like to thank my research colleagues, among them, Luca Bertuccelli, Louis Breger, Ian Garcia, Ellis King, Yoshiaki Kuwata, Megan Mitchell, Arthur Richards, Chung Tin and Steven Waslander. Thanks also to Professor How's administrative assistant, Margaret Yoon for her support throughout this time and her help to put final touches on this thesis.

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, and my sisters, Jila, Jaleh and Laleh.

This research was funded in part under DARPA contract # N6601-01-C-8075 and Air Force grant # F49620-01-1-0453. The testbeds were funded by DURIP Grant #F49620-02-1-0216.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the recent world events, there is a significant interest in extending the capabilities of future Unmanned Aerial Vehicles (UAVs) to support ground forces, provide timely intelligence, and execute the "dull, dirty tasks in harm's way" [1, 2, 3, 4]. UAVs can be sent into missions that would endanger the lives of the aircrews of manned vehicles, such as the Suppression of Enemy Air Defense (SEAD) missions for chemical manufacturing facilities with a high concentration of Surface to Air Missile (SAM) sites. UAVs can also stay aloft longer for surveillance and reconnaissance missions. One key extension envisaged for the future is using multiple UAVs to perform coordinated search, reconnaissance, target tracking, and strike missions. However, several fundamental problems in decision making and control must be addressed to ensure that these autonomous vehicles reliably (and efficiently) accomplish these missions. The main issues are high complexity, an uncertain and very dynamic environment, and partial/distributed information.

Figure 1.1 shows an example of a typical SEAD UAV mission with heterogeneous vehicles, fixed obstacles, removable no-fly-zones (NFZ) associated with the SAM sites, and targets of various types. The environment is uncertain, risky, and very dynamic. Figure 1.2 presents an abstraction of this problem with multiple aircraft with different capabilities, payloads, and sensors. Some UAVs will be tasked to find and suppress the air defenses, some will attempt to strike the high value targets, and others must assess the damage done. These tasks must be done in a specific order and the goal is

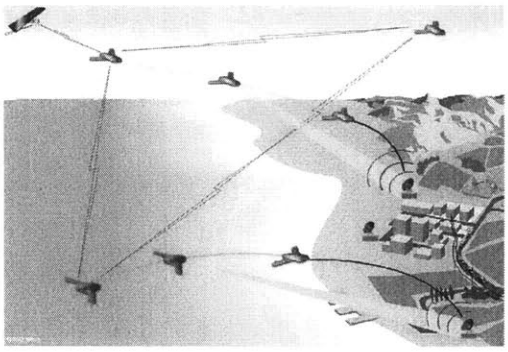


Fig. 1.1: Typical UAV mission [5].

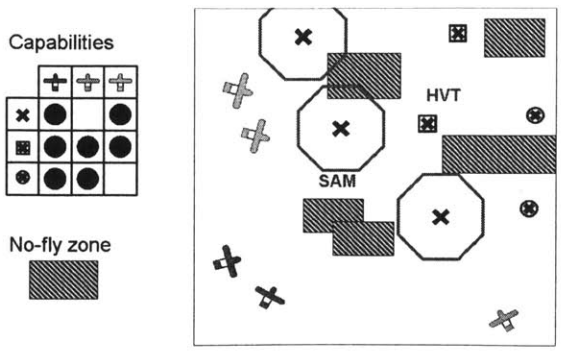


Fig. 1.2: Mission abstraction.

to maximize the team performance. Thus, new methods in planning and execution are required to coordinate the operation of the fleet of UAVs. In particular, an overall control system architecture must be developed that can perform optimal *coordination* of the vehicles, evaluate the overall system performance in real time, and quickly reconfigure to account for changes in the environment or the fleet. This thesis presents new results that address the issues of risk and variability in the environment in the task assignment part of the *cooperative path planning* (CPP) problem for UAVs.

For many vehicles, obstacles, and targets, fleet coordination is a very complicated optimization problem [6, 7, 8], and the computation time increases very rapidly with the problem size. J. Bellingham, et.al. [9] developed an approach to decompose this large problem into assignment and trajectory problems, while capturing key features of the coupling between them. This allows the control architecture to solve an allocation problem first to determine a sequence of waypoints for each vehicle to visit [8], and then concentrate on designing trajectories to visit these pre-assigned waypoints [10, 11]. The decomposition approach simplifies the coupling between the assignment and trajectory design problems by calculating and communicating only the key information that connects them [8]. The cost calculation is done using a straight line approximation of the feasible paths around the "obstacles" (*e.g.,* buildings, no-fly-zones) in the environment. These costs are then used in the assignment problem solved by a *petal* algorithm [12, 8], which is a heuristic that significantly decreases the computation time of the task assignment by reducing the problem size. This reduction is accomplished by enumerating the plans (petals) for each UAV and

then pruning infeasible plans and/or ones that are unlikely to be part of an optimal plan. The key here is that these cost calculations and pruning are done prior to the optimization, which then allows us to pose a much smaller optimization problem. While the resulting assignment algorithm was shown to be much faster than the original formulation, experiments have shown that, with timing constraints, it is still too slow to be considered for use in real-time applications [13]. Thus the goals of the research in this thesis were to address the following key problems:

1. Develop algorithms to perform UAV task assignment (with side constraints) in real-time for dynamic environments.
2. Develop modifications to these faster task assignment algorithms to reduce their sensitivity to noise and uncertainty in the environment.
3. Modify the task assignment formulation to ensure both *coordination* and *cooperation* between the UAVs to improve the fleet performance in a risky environment:

    - *Coordination*: allocating tasks to meet the constraints, avoid overlap, and optimize some objective.
    - *Cooperation*: coordinated TA with additional knowledge of the future implications of a UAV's actions on improving the expected performance of the other UAVs.

1.1 Literature Review

Numerous researchers have examined all aspects of the UAV assignment problem [4, 6, 7, 8, 14, 15, 16, 17, 18, 19, 20, 21, 22]. This includes traditional methods for vehicle routing problems (VRPs) from the operations research (OR) and artificial intelligence (AI) communities [23, 24]. Exact optimization methods such as Branch and Bound, Branch and Cut, Constraint Satisfaction Problems (CSPs), and Dynamic Programming (DP) have been used to solve the problem to optimality. While guaranteed to yield optimal results, these methods are computationally intensive, and this complexity becomes an important issue when the problem has *hard* side constraints [25].

These complexity issues make the exact methods intractable for many problems, either because they are too large or they take too long to solve. As a result, several approximation methods have been proposed to resolve the complexity issues.

Classical heuristic methods, such as constructive and two phase methods were used to solve larger size VRP problems in a reasonable amount of time [12]. These methods, however can generate solutions that are far from optimal. Different metaheuristics methods such as Tabu search, Simulated Annealing, and Genetic Algorithms have also been proposed in recent years for the VRP. These methods, which typically embed a classical heuristic inside them, often give better solutions to the problem than classical heuristic methods, but they also tend to be more time consuming [26, 27, 28]. These approximations help to reduce the computation time compared to the exact methods, but most of these methods are still computationally intractable for real-time replanning.

Iterative network flow algorithms, in which tasks are assigned to UAVs sequentially in a greedy fashion, have also been the focus of recent research on UAV task assignment [7]. These heuristic methods are shown to be able to calculate the assignment very rapidly compared to other existing methods. These methods however, can generate plans that are far from optimal. This disadvantage of greedy methods will be discussed extensively in this thesis.

Approaches to the allocation problem which emphasize timing constraints have also been proposed [4, 15, 16]. In these approaches, detailed paths are selected for each of the vehicles in order to guarantee simultaneous arrival at an anti-aircraft defense system, while minimizing exposure to radar along the way. However, these methods require that task assignment and trajectory design to be solved simultaneously which increases the problem size and makes it prohibitive for large problems.

This work presents a methodology to solve the UAV task assignment problem rapidly for real-time applications while the performance is kept close to optimal. A key distinction in this work is that, while it uses standard OR task assignment tools, we are not operating them "open-loop" (or at steady-state) as is typically done in the OR field, but instead we are using these algorithms in a "high bandwidth" closed-

loop system. As a result we might expect similar stability problems to occur from mis-modeling the environmental disturbances, noise in the sensor measurements, and uncertainty in the estimation of the environment.

1.2 Thesis Outline

Chapter 2 of this thesis presents a new approach to the task assignment algorithm in which an assignment is achieved for each UAV in a short time that makes it suitable for real-time replanning in dynamic battlefields. A reliable planning system must be able to account for changes such as moving targets, UAV losses, etc. and generate a new optimal plan rapidly. This chapter introduces *Receding Horizon Task Assignment* (RHTA) algorithm based on the "petal" algorithm and further illustrates that a good solution (close to optimal) can be achieved in a reasonable amount of time, suitable for real-time applications. Simulations using Boeing's Open Experiment Platform (OEP) [29] and results from a hardware testbed are presented in Chapter 5 to demonstrate the effectiveness of RHTA in dynamic and real-world environments.

Task assignment in the controls literature has been generally viewed as an open-loop optimization with deterministic parameters. The optimization is generally done once, and task reassignment occurs only when substantial changes in the environment have been observed. In reality, these information updates are continuously occurring throughout the mission due to the vehicle sensing capabilities, adversarial strategies, and communicated updates to the vehicle situational awareness (SA). In this case, the typical response to a change in the SA is to reassign the vehicles based on the most recent information. The problem of reassigning due to the effect of changes in the optimization has been addressed by R. Kastner, *et.al.* [30] in their use of incremental algorithms for combinatorial auctions. The authors propose that the perturbed optimization problem should also include a term in the objective function that penalizes changes from the original solution. The work of J. Tierno and A. Khalak [31] also investigates the impact of replanning, with the objective function being a weighted sum of the current objective function and the plan difference from the previous op-

timization to the current one. Both of these formulations rely on the plan generated prior to the current one as a reference. They do not consider the impact of noise in the problem, nor do they develop techniques to mitigate this effect on the replanning. Chapter 3 presents a modified formulation of the task assignment problem that can be used to tailor the control system to mitigate the effect of noise in the SA on the solution. The approach here is to perform the reassignment at the rate that the information is updated, which enables us to react immediately to any significant changes that occur in the environment.

Chapter 4 considers a stochastic Mixed-Integer Linear Programming MILP formulation of the task assignment problem, which maximizes the expectation of the mission's vale (score) and achieves cooperation between UAVs. This formulation addresses one of the most important forms of coupling in the assignment problem; the coupling between the mission that one UAV performs and the risk that other UAVs experience. Each UAV can reduce the risk for other UAVs by destroying the anti-aircraft defenses that threaten them. While the approach in Ref. [28] assumes a fixed risk for visiting each of the waypoints, the ability to reduce this threat is not addressed directly. The formulation in Chapter 4 optimizes the use of some vehicles to reduce risk for other vehicles, effectively balancing the score of a mission, if it were executed as planned, against the probability that the mission can be executed as planned.

Chapter 4 further extends the idea of cooperation to the Weapon Task Assignment (WTA) and proposes a Dynamic Programming algorithm as the way to achieve the cooperation in a WTA problem. To reduce the computation complexity and solve the curse of dimensionality associated with DP algorithm, two approximation DP methods are also proposed to solve this problem. It is shown that these methods can reduce the complexity of the problem considerably while keeping the performance close to optimal.

Chapter 2

Receding Horizon Task Assignment

2.1 Motivation

Battlefields are dynamic environments that change rapidly. Targets move, their values change with time, UAVs get shot down, etc. A reliable planning system therefore, must be able to account for these changes and generate a new optimal plan, including both the UAV task assignment and the detail trajectory design. However, the exact algorithms for UAV task assignments that give optimal plans are slow and typically cannot be implemented in real-time for reasonably large problems. To overcome these computational issues, several approximation methods have been developed that are much faster than the exact methods, but most of these are still not suitable for real-time implementation. More extreme approximations can solve the problem very rapidly, but usually yield poor performance results. Greedy algorithms, which are very fast and therefore can be implemented in real-time, also usually perform poorly. This chapter presents a new *receding horizon* methodology that can perform the UAV task assignment in real-time and yield close to optimal performance. Next section describes the petal algorithm which is an approximation method for UAV task assignment and is the base for Receding Horizon Task Assignment (RHTA) algorithm [12, 8].

2.2 Background

This section defines the UAV task assignment problem and establishes the basis for the new Receding Horizon Task Assignment (RHTA) approach to this problem. The RHTA is based on the petal algorithm [12, 8]. In using these algorithms (petal and RHTA), several assumptions are made. First, the set of tasks have been identified for each team of UAVs. Second, the tasks have been divided between the team of UAVs and the waypoints for each team have been identified. The location of the waypoints are presented by a $N_w \times 2$ matrix B as $[\ B_{\mathrm{wx}} \quad B_{\mathrm{wx}}\ ]$. Each team is made up of N_v UAVs with known starting points, speed, and capability (*i.e.,* strike, reconnaissance, etc.). The starting state (its initial position) of the UAV v is given by the v^{th} row of the matrix S_0 as $[\ x_{0\mathrm{v}} \quad y_{0\mathrm{v}}\ ]$. The amount of munitions available on each UAV is also known.

The UAV capabilities are represented by the $N_v \times N_w$ binary matrix K. $K_{vw} = 1$ represents a UAV v capable of performing the task associated with waypoint w ($k_{vw} = 0$ if it cannot perform the task). It is also assumed that there are polygonal "No Fly Zones" (shown as a rectangle for simplicity) in the environment. The location and size of the rectangles are designated by the coordinates of the lower-left corner of each obstacle j as $(Z_{\mathrm{j1}}, Z_{\mathrm{j2}})$ and the upper-right corner as $(Z_{\mathrm{j3}}, Z_{\mathrm{j4}})$. These two pairs together, make up the j^{th} row of the $N_z \times 4$ matrix Z.

Given this information, the problem is to assign the UAVs to the waypoints to optimally fulfill a specific objective. There are several possibilities for developing a generic objective function that can be specifically adjusted to different problems. The most common objective of these types of problems is to minimize mission completion time, which is defined as the time that the last UAV finishes its mission. The minimum time formulation is presented here and this formulation is then extended to consider the cost that reflects the value of the mission. The minimum time objective can be written as

$$\bar{t} = \max_{v}\ t_v \tag{2.1}$$

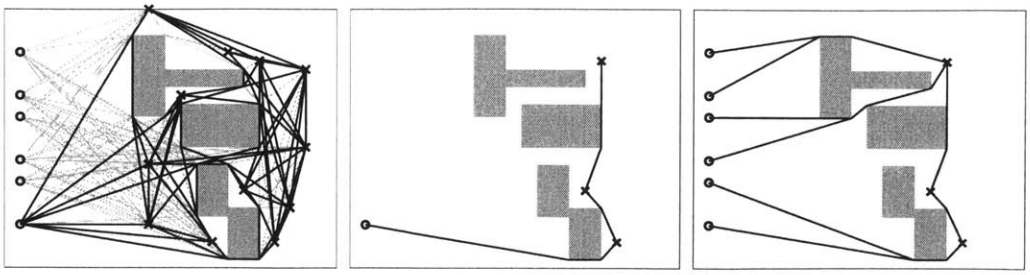
$$J_1(\bar{t}, \mathbf{t}) = \bar{t} + \frac{\alpha}{N_V} \sum_{v=1}^{N_V} t_v \tag{2.2}$$

where t_v is the time that UAV v finishes its mission. $\alpha \ll 1$ weights the average completion time compared to the maximum completion time. If the penalty on the average completion time were omitted (*i.e.*, $\alpha = 0$), the solution could include assigning unnecessarily long trajectories to all UAVs except for the last to complete its mission.

The minimum time coordination problem could be solved by first planning detailed trajectories for all the possible assignments of waypoints to the UAVs and all the possible orderings of those waypoints, and then choosing the detailed trajectories that minimize the cost function, denoted $J_1(\bar{t}, \mathbf{t})$ [32], but there are many such possibilities and designing each is computationally demanding. Instead of planning detailed trajectories for all possible task allocations, the petal algorithm constructs estimates of the finishing times for only a subset of the feasible allocations, and then performs the allocation estimated to best minimize the cost function [8].

The algorithm developed for this approach can be explained as follows. First, a list of all un-ordered feasible task combinations were enumerated for every UAV, given its capabilities. Next, the length of the shortest path made up of straight line segments between the waypoints and around obstacles was calculated for all possible order-of-arrival permutations of each combination. The construction of these paths can be performed extremely rapidly using graph search techniques [8]. The minimum finishing time for each combination was estimated by dividing the length of the shortest path by the UAV's maximum speed. Some of the task allocations and orderings have completion times that are so high that they can confidently be removed from the list to reduce the decision space of the optimization (pruning), and solve the allocation problem faster. Given these cost estimates, the allocation problem was then solved to find the minimum time solution.

The steps of this algorithm are depicted in greater detail Figures 2.1(a)–2.1(c), in which a fleet of UAVs (designated $\circ$) must visit a set of waypoints (designated $\times$). First, the visibility graph between the UAV starting positions, waypoints, and



(a) Visibility graph and shortest paths between UAV 6, all waypoints

(b) Shortest path for UAV 6 over one combination of waypoints

(c) Shortest paths for all UAVs over same combination of waypoints

Fig. 2.1: Visibility Graph and Shortest Paths for the Allocation Problem.

obstacle vertices is found. The visibility graph is shown in Figure 2.1(a) as grey lines. Next, the algorithm searches this graph to find the shortest paths between all pairs of waypoints, and the starting position of each UAV and all waypoints (Figure 2.1(a) shows the results for UAV 6 with black lines). In Figure 2.1(b), one combination of fewer than $n_{\max}$ waypoints has been chosen, and the shortest path from UAV 6's starting position to visit all of them is shown. In Figure 2.1(c), the shortest path to visit the same combination of waypoints is shown for each vehicle. Note that, as expected, the best permutation of these waypoints is not the same for all vehicles.

The algorithm produces four matrices whose p^{th} columns, taken together, fully describe one permutation of waypoints. These are vector row $\mathbf{u}$, whose element u_p identifies which UAV participates in the p^{th} permutation; $\mathbf{V}$, whose V_{ip} entry is 1 if waypoint i is visited by permutation p and 0 if not; $\mathbf{T}$, whose T_{ip} entry is the time at which waypoint i is visited by permutation p, and 0 if waypoint i is not visited; and $\mathbf{C}$, whose element C_p is the time at which permutation p is completed. This procedure is described in detail in [8].

Once the approximate costs for a UAV to visit a set of waypoints were calculated, mathematical method was developed for allocating the waypoints to each UAV based on these costs and other constraints. The base of the task allocation problem was formulated as a Multidimensional Multiple-Choice Knapsack Problem (MMKP) [33].

The "knapsack" in this case is the complete mission plan. The **V** column corresponding to each of the N_M permutations makes up the multi-dimensional weight. The "multiple-choice" comes from choosing which permutation to assign to each of the N_V different UAVs (sets). The objective is to assign one permutation (element) to each vehicle (set) that is combined into the mission plan (knapsack), such that the cost of the mission (knapsack) is minimized and the waypoints visited (weight) meet the constraints for each of the N_W dimensions. The problem is given by

$$\begin{aligned} \min \ J_2 &= \sum_{p \in \mathcal{M}} C_p x_p \\ \text{subject to} \quad & \forall i \in \mathcal{W} : \sum_{p \in \mathcal{M}} V_{ip} x_p = 1 \\ & \forall v \in \mathcal{V} : \sum_{p \in \mathcal{M}_v} x_p = 1 \end{aligned} \tag{2.3}$$

where $\mathcal{M} = \{1, \ldots, N_M\}$, $\mathcal{M}_v \subseteq \mathcal{M}$ are the permutations that involve UAV v, and $\mathcal{W} = \{1, \ldots, N_W\}$ is the list of waypoints. The binary decision variable $x_p = 1$ if permutation p is selected, and 0 otherwise. The cost in this problem formulation minimizes the sum of the costs to perform each selected permutation. The first constraint enforces that waypoint i is visited once. The second constraint prevents more than one permutation being assigned to each vehicle.

The solution to the MMKP selects a permutation for each vehicle. The cost in Eq. 2.2 is a weighted combination of the sum of the individual mission times (as in the MMKP problem) as well as the total mission time. In order to include the total mission time in the cost, a new continuous variable, $\bar{t}$, is introduced and the following constraint is added to the original formulation in Eq. 2.3

$$\sum_{p \in \mathcal{M}} C_p x_p \leq \bar{t} \tag{2.4}$$

The constraint forces $\bar{t} = \max_v \ t_v$ and allows the total mission time to be included

in the cost. The new cost is as follows,

$$J_3 = \bar{t} + \frac{\alpha}{N_V} \sum_{p \in \mathcal{M}} C_p x_p \tag{2.5}$$

The problem is now a Mixed-Integer Linear Programming (MILP) problem that can be solved using commercially available software such as CPLEX [34]. The solution to the task allocation problem is now a set of ordered sequences of waypoints for each vehicle, which ensure that each waypoint is visited the correct number of times while minimizing the mission completion time.

2.2.1 Time-Discounted Value as the Objective

Minimizing the mission completion time is one of the many possible objectives that can be used in the allocation problem. In the min-time formulation, all of the target values are considered to be identical and the objective is to visit all of them in the minimum possible time. However, in real-world applications, the values of the targets are quite different and therefore it is desirable to assign UAVs to specific targets based on their values. In many situations, UAVs are either not capable of visiting all of the targets, or they are not interested in visiting risky, low-valued targets. This section introduces a different objective function that captures the notion of target value. Chapter 4 presents a more general objective function that accounts for both target values and mission risks. The value-based objective function can be written as

$$\max J_4 = \sum_{i=1}^{N_t} \lambda^{t_i} s_i x_i \tag{2.6}$$

where s_i is the value associated with the task at waypoint i, and t_i is the time in which waypoint i is visited. $0 \leq \lambda \leq 1$ is a discount factor that accounts for the decrease in target value with time. This factor is included in the objective function to better represent the real-world problems in which the value of visiting a target decreases proportional to the time in which it is visited. For example, consider the case of mobile Surface to Air Missile (SAM) sites. Once the enemy finds out that

their SAM sites have been identified, they will typically start moving the missiles to a new location and, as a result, the task at the original waypoint loses value over time. With this new objective function, the constraint in Eq. 2.3 that forces the problem to assign UAVs to all targets can be relaxed, which allows more general problems to be solved. In the next section this objective function is used to perform the task allocation in the *Receding Horizon Task Assignment* (RHTA) algorithm.

2.3 Iterative Methods

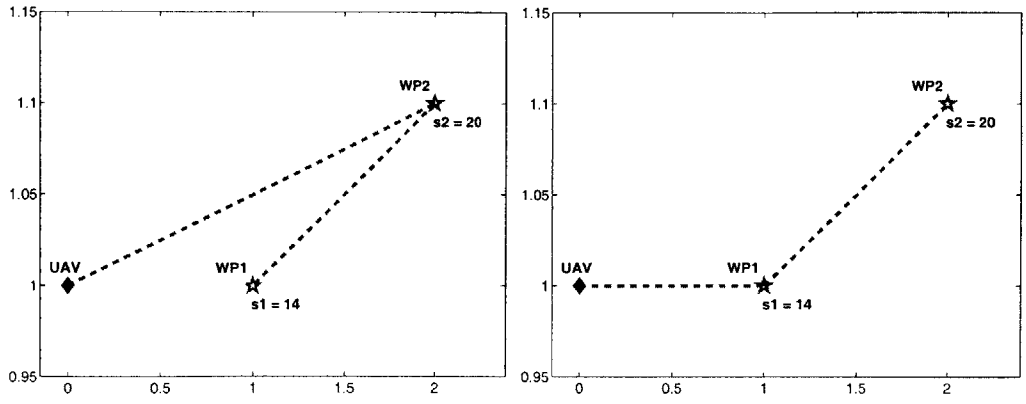
The petal algorithm explained in section 2.2 is fast compared to exact algorithms and results in optimal assignments when pruning is done properly. The degree of pruning needs to be balanced because pruning too many petals can lead to poor performance, but insufficient pruning can result in extensive computation time. It is shown in [8, 13] that the petal algorithm can be applied to fairly large problems, but the computation time increases rapidly as the size of the problem grows. This is due to an enumeration of all possible combinations of targets [8]. The *Receding Horizon Task Assignment* (RHTA) algorithm is proposed to solve the computation time issues by solving the large problem by breaking it down to smaller problems and iteratively solving the smaller problems. RHTA still uses the petal algorithm to generate possible assignments for each UAVs and solves a MILP to pick the best. The difference is that in RHTA the size of each combination (the size of each petal) is constrained. This limits the number of combinations that have to be analyzed and significantly reduces the size of the optimization problem. Of course, limiting the size of each combination (petal) will result in an incomplete set of assignments, in the sense that waypoints are left unassigned even though there are UAVs capable of visiting these waypoints. To solve the problem to completion, the same procedure is applied to the remaining targets to generate a new set of petals for each UAV. These petals (new assignments) are then added to the previous assignments. This process is continued to completion (*i.e.*, either all the waypoints are assigned or no more waypoints can be assigned due to munition limitation).

A greedy algorithm in which targets are assigned to UAVs one by one in an iterative fashion similarly reduces the size of each petal to 1. Greedy methods usually have a huge advantage in computation time over all other methods, but yield poor performance results, negating the savings in computation time. The problem with an iterated greedy and similar "myopic" methods is that, in the process of finding the best assignment (best petal) in the current iteration, the effect of the remaining waypoints is ignored, as illustrated in Figure 2.2 where the result of a greedy assignment is compared with the optimal assignment for a simple example.

In this example the objective is to maximize the time-discounted value achieved by one UAV visiting two targets with different values. The discount factor (λ in Eq. 2.6) is set to 0.75. The resulting assignment from the two algorithms are quite different. In the greedy assignment (Figure 2.2(a)), the high value waypoint WP_2 is visited first (this is the nature of greedy algorithms) and a value of 11.25 is achieved in the first iteration. In the second iteration WP_1 is visited and the total accumulated value for this assignment is 17.2. In the optimal solution, waypoint WP_1 is visited first, resulting in a value of 10.5 and waypoint WP_2 is visited next for a total accumulated value of 21.5. The flaw in the greedy assignment comes from ignoring the future. In the first stage it compares the time discounted value in WP_1 and WP_2. Because the discounted value is greater for WP_2, it chooses this waypoint first. The degradation in performance becomes crucial for larger problems since the greedy algorithm ignores a bigger portion of the problem by ignoring the future. The next section details how the Receding Horizon Task Assignment solves the performance issues associated with greedy methods while keeping the computational demands low.

2.3.1 Receding Horizon Task Assignment

Receding Horizon Control, used interchangeably with Model Predictive Control (MPC), is a well known methodology in controls literature [35, 36]. When applying an optimal control is not possible due to the complexity or size of the problem, MPC is a good replacement candidate. MPC approximates an infinite horizon optimal feedback control problem by using the online solution of a finite-horizon optimization.



(a) Greedy Solution (b) Optimal Solution

Fig. 2.2: Comparing greedy performance to the optimal solution

The first portion of the plan is executed and the process is repeated. This combines feedforward control, as the problem plans for future actions, with feedback, as each new plan uses new information. The RHTA essentially follows the same idea. In each iteration the plan for each UAV is limited to $m \ll N_w$ waypoint visits. This means in enumerating combinations of waypoints only the combinations of size less than or equal to m are enumerated . This number of combinations is far smaller than the number of combinations enumerated using the petal algorithm, especially for large problems. Having these combinations, the best permutation for each combination is calculated the same way as in the petal algorithm, generating a set of paths of maximum size m for each UAV. Having these sets of paths (also called petals), the following optimization is applied to generate the best path for each UAV.

$$\max J_5 = \sum_{v=1}^{N_v} \sum_{p=1}^{N_{vp}} S_{vp} x_{vp} \tag{2.7}$$

$$\text{subject to} \quad \sum_{v=1}^{N_v} \sum_{p=1}^{N_{vp}} A_{vpi} x_{vp} \leq 1 \tag{2.8}$$

$$\forall v \in 1 \ldots N_v \qquad \sum_{p=1}^{N_{vp}} x_{vp} = 1 \tag{2.9}$$

$$x_{vp} \in \{0, 1\} \tag{2.10}$$

where N_{vp} is the number of petals for $UAV\ v$. x_{vp} is a binary variable and equals 1 if the p^{th} petal of $UAV\ v$ is selected and 0 otherwise. S_{vp} represents the value of p^{th} petal of $UAV\ v$ and is calculated prior to the optimization:

$$S_{vp} = \sum_{i \in \mathcal{W}_p} \lambda^{t_{ip}} s_i \qquad \forall v \in 1 \ldots N_v \tag{2.11}$$

$\mathcal{W}_p$ is the index of waypoints in the p^{th} petal and t_{ip} is the time in which waypoint i is visited in petal p. In the first constraint (Eq. 2.8), A_{vpi} equals 1 if waypoint i is visited in permutation p of $UAV\ v$ and 0 otherwise. This constraint ensures that each waypoint is visited at most once. The second set of constraints limits each UAV to only one petal. The above optimization is a Mixed-Integer Linear Programming (MILP) and can be solved using CPLEX to find the best assignment for each UAV.

Having the best petal (list of waypoints to visit) for each UAV, the first waypoint in the list of each UAV is picked and assigned as the first waypoint to be visited by that UAV The assigned waypoints (one for each UAV) are then removed from the list of waypoints to visit, leaving a smaller list of waypoints to visit. The smaller list becomes a new assignment problem to be solved. The new position of UAVs will be the position of their assigned waypoints. The starting time for each UAV needs to be updated using the distance between the starting position of UAVs and the position of their first targets. The same procedure is then used to produce the next set of waypoints for the UAVs, and the procedure is repeated until all waypoints are assigned, or there are no resources left (*i.e.,* no munitions). This algorithm gives an ordered list of waypoints to be visited by each UAV. All the steps of this approach are shown in Algorithm 2.3.1. The performance advantages of this algorithm over other iterative methods such as greedy algorithm is the result of looking at the near future in each step of planning. In each step, the algorithm plans for $m > 1$ waypoints for each UAV but only uses the first waypoint for each UAV. This helps the algorithm to avoid "myopicness" which causes the iterative methods to degrade in performance.

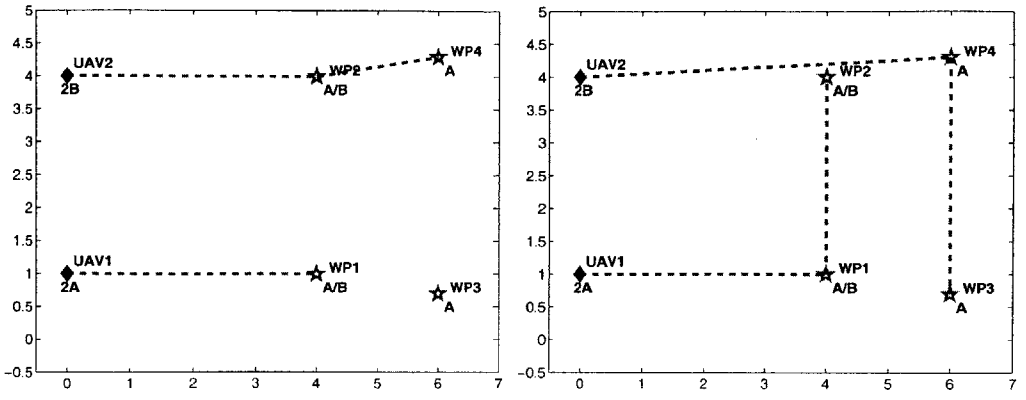
Find the shortest distance between all waypoint pairs (i, j) as $D(i, j)$ using straight lines around obstacles;
set $W = W_0$ (the set of all waypoints)
set $M_v = \emptyset, v := 1, \ldots, N_v$ (The mission for UAVs. Sequence of waypoints to visit)
set $T_v = 0, v := 1, \ldots, N_v$ (The initial time of UAVs)
while $W \neq \emptyset$ **do**
  for all UAVs v **do**
    $p := 1$;
    for all numbers n_c of waypoints to visit, $n_c := 1, \ldots, m$ **do**
      for all combinations $\mathcal{C}$ of n_c waypoints that v is capable of visiting **do**
        for all permutations i of waypoints $[w_1, \ldots, w_{n_c}]$ in $\mathcal{C}$, with $i := 1 \ldots n_c!$ **do**
          $T_{1i} := d(w_1)/v_{max} + T_v$;
          $S_i = \lambda^{T_{1i}} s_{w_1}$;
          for $d := 2 \ldots n_c$ **do**
            $T_{di} = T_{(d-1)i} + D(w_{d-1}, w_d)/v_{max}$; \\ {Cumulative time from start}
            $S_i \leftarrow S_i + \lambda^{T_{di}} s_{w_d}$;
          end for
          $P_i = [w_1, \ldots, w_{n_c}]$;
        end for
        $i_{max} = \text{argmax}_i S_i$; \\ {Choose the best permutation.}
        $S_{vp} = S_{i_{max}}$;
        $P_{vp} = P_{i_{max}}$;
        $p \leftarrow p + 1$;
      end for
    end for
  end for
  solve the optimization problem to find the best permutation for each UAV v
  $p_{v_{max}} = \text{argmax}_p S_{vp} : v := 1, \ldots, N_v$
  for all UAVs v **do**
    $w_v = P_{vp_{v_{max}}}(1)$; \\ {Picks the first waypoint in the permutation}
    $M_v \leftarrow [M_V \ \ w_v]$; \\ {Adds the waypoint to the mission list of UAV}
    $T_v = d(w_v)/v_{max} + T_v$; \\ {updates the time of each UAV}
    $W \leftarrow W - w_v$; \\ {removes the selected waypoints from the list}
  end for
end while

Algorithm 2.3.1: Receding Horizon Task Assignment Algorithm

2.3.2 Munition Constraints

The UAVs are assumed to be of different types and carry different types of munitions. The tasks at the waypoints are also of different types. One important objective in the UAV assignment problem is to assign the UAV with the right munition to a specific waypoint. If waypoint w can only be visited by UAVs that carry munitions of type A, a UAV without this munition should not be assigned to this waypoint. It is a simple constraint that can easily be included in the optimization problem when the assignment is done in a single optimization problem. However, it can cause a problem in any iterative method if it is not implemented correctly. For example consider the simple 2-UAV, 4-waypoint example in Figure 2.3 which compares the results of an iterated greedy method with the optimal solution. In this example, UAV_1 and UAV_2 have two munitions of type A and two munitions of type B, respectively. Waypoints WP_1 and WP_2 each can be visited by UAVs carrying munitions of either type, but waypoints WP_3 and WP_4, can only be visited by UAVs with type A munitions. In the optimal solution (Figure 2.3(b)), UAV_1 visits waypoints WP_3 and WP_4 and UAV_2, visits the rest of the waypoints. But in the greedy case, the first stage of planning ignores the future, so in order to assign the correct UAV to one target, the best assignment is UAV_1 to WP_1 and UAV_2 to WP_2. Given these assignments, in the next stage, UAV_2 is not able to visit the remaining waypoints WP_3 and WP_4, and therefore the mission will be incomplete. The incomplete mission can be viewed here with two perspectives: one perspective is to say that the greedy mission generates a feasible answer that is poor in performance. The second perspective is to say that it resulted in an infeasible assignment, in which not all the possible waypoints are visited. In either case, the greedy mission fails because it ignores the future.

To resolve this issue, a set of constraints is introduced here to be implemented in iterative methods. This set of constraints ensures that each waypoint is visited by the right type of munitions and also the number of munitions that is used by each UAV is less than or equal to its capacity. Suppose there are K types of munition, $M_1, M_2, \ldots, M_K$. Also assume that $M_1 > M_2 > \ldots > M_K$ which means, if a waypoint



(a) Greedy Solution (b) Optimal Solution

Fig. 2.3: A case where the greedy method fails to find feasible solution

can be visited by munition M_i it can also be visited by any munition of type M_j, where $j \leq i$. As an example, consider long-range missiles (LRM) and small-range missiles (SRM). A waypoint that needs to be targeted by SRMs can also be targeted by a LRM, but not vice versa. For each waypoint w there is an index i for which any munition of type $j \leq i$ can be used. This is represented by the binary matrix θ of size $K \times N_W$. A matrix ψ of size $K \times N_V$ represents the capacity of the UAVs. The element (i, v) of ψ represents the number of munitions of type M_i in the UAV v. The implementation of these constraints are done both inside and outside the optimization. Prior to the optimization, the required munition of each type to visit all the remaining waypoints is calculated.

$$\phi_i = \sum_{w=1}^{N_w} \theta_{iw} \qquad \forall i \in 1 \ldots K \tag{2.12}$$

For each petal p the required munition of each type is also calculated

$$\mu_{ip} = \sum_{w \in \mathcal{W}_p} \theta_{iw} \qquad \forall i \in 1 \ldots K \tag{2.13}$$

For each petal p and each UAV v, the remaining munition for UAV v, Ω_p is also calculated, if it gets assigned to petal p. This is not simply the difference between the

current UAV munitions and the required munition because we must account for the fact that a munition with a smaller index can substitute for a munition with a larger index. In calculating these values, if a UAV has less than the required munition for a specific petal, then that petal will be eliminated since it cannot be part of a feasible solution.

These values(ϕ, μ, Ω) are then written into a data file and used in the optimization. The constraints in the optimization environment are:

$$\varphi_i = \sum_{p \in P} \sum_{j=i}^{K} \Omega_{jp} x_p \qquad \forall i \in 1 \ldots K \tag{2.14}$$

$$\chi_i = \sum_{j=i}^{K} \phi_j \qquad \forall i \in 1 \ldots K \tag{2.15}$$

$$\varphi_i \geq \chi_i \qquad \forall i \in 1 \ldots K \tag{2.16}$$

φ_i is the remaining munition of type M_i after this iteration. χ_i is the required munition of type M_i for the rest of the mission and Eq. 2.16 ensures that the remaining munitions are enough to complete the mission.

2.3.3 Munition Constraints as an Effective Cost-To-Go

Another way of applying the munition constraints is to relax them and add them to the objective function as a penalty term. There are some advantages for doing that. One is that the same algorithm can be used for the case in which the available munitions are less than those required for the mission. In this case, if hard constraints are used, the resulting optimization will be infeasible. By relaxing these constraints and adding them to the objective function (soft constraints), the best assignment with the available munitions will be achieved. Another advantage of soft constraints is that if a good value for the penalty coefficient is chosen, the penalty function would play the role of an effective "cost-to-go" function. The optimization then has the flexibility of trading between optimality and feasibility (here by feasibility we mean visiting all the possible waypoints). To transform the munitions constraint

from a hard constraint in Eq. 2.15-2.16 to a soft constraint, Eq. 2.16 is replaced with

$$\xi_i = \begin{cases} 0 & \text{if} \quad \varphi_i \geq \chi_i \\ \varphi_i - \chi_i & \text{if} \quad \varphi_i \leq \chi_i \end{cases} \qquad \forall i \in 1 \ldots K \tag{2.17}$$

and the objective function is rewritten as

$$\max J_6 = \sum_{v=1}^{N_v} \sum_{p=1}^{N_{vp}} S_{vp} x_{vp} + \gamma \sum_{i=1}^{K} \xi_i \tag{2.18}$$

where γ is the penalty coefficient. Note that the best choice of γ is problem specific.

2.3.4 Time Constraints

Time and precedence constraints are crucial components of UAV planning. For any target, the sequence of identification, strike, and bomb damage assessment (BDA) should be kept in the right order. Synchronous observation of a target by several UAVs is another example of timing constraints. When assignment is done as a single MILP problem, formulation and inclusion of these constrains are simple, although the resulting problem will be much harder and take a much longer time to solve [13]. These types of constraints are referred to as "hard side constraints" in operations research literature. In [13], a method to include timing constraints with loitering in the petal algorithm is extensively discussed and the impact of different types of constraints in the computation time is examined. Here, the methodology is briefly presented and a much simpler method is introduced to include timing constraints in the RHTA, which does not change the complexity of the problem.

Suppose there are N_t timing constraints, which are represented by a $N_t \times 2$ matrix F and vector d of size N_t. The k^{th} row of F, $[i, j]$ along with the k^{th} element of d, d_k specify a timing constraint that forces the task j to be executed at least d_k time steps after task i. Defining *TOE* (Time Of Execution) as the vector that represents the execution time for each task gives each constraint the form of $TOE_j \geq TOE_i + d_k$. To make the problem more general, d_k need not be a positive number. The loitering

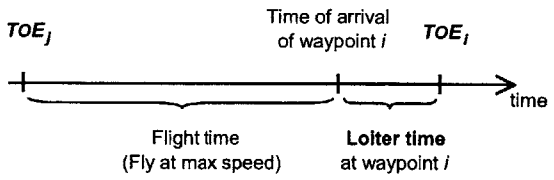


Fig. 2.4: Flight time, loiter time, time of arrival, and time of task execution.

time at waypoint i is defined as the time difference between the time of execution and the time of arrival at waypoint i (Figure 2.4). The UAVs are assumed to fly at the maximum speed between waypoints and expected to loiter before executing the task. Note that they can also be regarded as flying at a slower speed between the waypoints, or loitering at the previous waypoint. To represent loitering, the $N_w \times N_v$ loitering matrix L is introduced, whose L_{ij} element expresses the loiter time at the i^{th} waypoint when visited by UAV j. The elements of matrix L are a set of new parameters in the optimization. Note that $L_{ij} = 0$ if waypoint i is not visited by UAV j. We need to include the loitering matrix to ensure that if the timing constraints are consistent (or, the problem is feasible), the algorithm finds a feasible solution.

In the MILP formulation, the time of the task execution at waypoint i, TOE_i, is written as

$$TOE_i = \sum_{j=1}^{N_M} T_{ij}\, x_j + LB_i\,, \qquad i = 1, \ldots, N_W \tag{2.19}$$

where the first term expresses the flight time from the start point to waypoint i at $v_{\max}$, and LB_i is the sum of the loiter times before executing the task at waypoint i. Define the set $\mathcal{W}$ such that $\mathcal{W}_i$ is the list of waypoints visited on the way to waypoint i (including i), so that

$$LB_i = \sum_{j \in \mathcal{W}_i} \sum_{k=1}^{N_V} L_{jk}\,, \qquad i = 1, \ldots, N_W \tag{2.20}$$

Only one UAV is assigned to each waypoint, and each row of $\mathbf{L}$ has only one non-zero element. To express the logical statement "on the way to", we introduce a large number M, and convert the one equality constraint equation (Eq. 2.20) into two

inequality constraints.

$$LB_i \leq \sum_{j=1}^{N_W} \left(O_{ijp} \sum_{k=1}^{N_V} L_{jk} \right) + M \left(1 - \sum_{p=1}^{N_M} V_{ip} x_p \right) \tag{2.21}$$

$$LB_i \geq \sum_{j=1}^{N_W} \left(O_{ijp} \sum_{k=1}^{N_V} L_{jk} \right) - M \left(1 - \sum_{p=1}^{N_M} V_{ip} x_p \right) \tag{2.22}$$

$\mathbf{O}$ is a three-dimensional binary matrix that expresses waypoint orderings, and $O_{ijp} = 1$ if waypoint j is visited before waypoint i (including $i = j$) by permutation p, and 0 if not. When waypoint i is visited by permutation p, the second term on the right-hand side of the constraints in Eqs. 2.21 and 2.22 disappears, producing the equality constraint:

$$LB_i = \sum_{j=1}^{N_W} \left(O_{ijp} \sum_{k=1}^{N_V} L_{jk} \right) \tag{2.23}$$

which is the same as Eq. 2.20. Note that when waypoint i is not visited by permutation p, $O_{ijp} = 0$ for all j and $V_{ip} = 0$, so that both of the inequality constraints are relaxed and LB_i is not constrained.

The cost function J_3 of Eq. 2.5 to be minimized in the optimization can be rewritten as:

$$J_7 = \bar{t} + \frac{\alpha}{N_V} \sum_{p \in \mathcal{M}} C_p x_p + \frac{\beta}{N_W} \sum_{j=1}^{N_V} \sum_{i=1}^{N_W} L_{ij} \tag{2.24}$$

The first term gives the maximum completion time of the team, the second gives the average completion time, and the third gives the total loiter times. $\beta \geq 0$ is used to include an extra penalty that avoids excessive loitering. The above methodology for including timing constraints can be easily translated and implemented in the time-discounted value formulation.

Timing Constraints in RHTA

Since RHTA is an extension of the petal algorithm, these constraints can be implemented the same way in the RHTA algorithm. But as discussed in [13], these types of constraints can considerably increase the computation time. Since the motivation of RHTA was to decrease the computation time and make it suitable for real-time implementation, including timing constraints in this fashion could defeat the objective. However, in the case where we are willing to gain performance at the price of computation time, this method might be useful. The following presents a very simple alternative to include timing constraints into the algorithm without changing the degree of complexity. The basic approach is to apply the timing constraints outside the optimization. To do so, in each iteration all the waypoints that their precedent is not visited are removed from the waypoint list. They are added back to the waypoint list as soon as their precedence waypoint is visited. For example if waypoint i is required to be visited before j then, waypoint i is the precedence for waypoint i. In the algorithm, waypoint j will be removed from the waypoint list initially and gets added to the waypoint list after waypoint i is assigned. Using loitering we also ensure that the required delays between the visits are satisfied.

2.4 Results

This section presents several simulation results to demonstrate the capabilities of the receding horizon task assignment algorithm.

2.4.1 Dynamic Environment

The simulation in Figure 2.5 applies the Task Assignment presented in this chapter to a dynamic environment. The structure of the problem was motivated by the AlphaTech SHARC challenge problem [14]. The changes in the environment during the simulation occur with no prior knowledge and the fleet is reassigned as the environment changes.

The simulation includes two types of vehicles and two types of obstacles. The solid black areas are no-fly zones that no vehicle can pass through, such as mountains or buildings. The second obstacle type, marked with a square, is a Surface to Air Missile (SAM) site that can detect within the surrounding circle. The $\circ$ vehicles are stealth vehicles (vehicles that are not observable by radar) capable of evading SAMs and are responsible for removing the SAM sites. The $\triangle$, $\diamond$, and $\triangledown$ vehicles do not have stealth capability and cannot fly through SAM zones. These vehicles are responsible for removing the targets marked with $\times$. These vehicles also are only capable of flying 60% of the maximum speed of the stealth vehicles.

In order to discuss the simulation, the vehicles are numbered as shown in Figure 2.5(a). Vehicles 1 and 2 are stealth vehicles, while vehicles 3–5 are not. The simulation proceeds left to right, row by row through the plots. Figure 2.5(a) shows the original environment and initial allocation of vehicles to targets.

The second plot (Figure 2.5(b)) shows how the fleet is reassigned when vehicle 1 removes SAM site R1. The trajectory for vehicle 4 is modified to fly through the previously obstructed SAM zone reducing the mission time and hence the time-discounted value for this vehicle. As a result, vehicle 4 trades target T2 to vehicle 5 and takes over target T4 in order to reduce the total mission time for the fleet. This demonstrates both the complete coordination among the fleet of vehicles and the ability of this method to make decisions regarding not just the current task, but future tasks that contribute to the total cost of the mission.

In Figure 2.5(c), vehicle 2 has removed SAM site R2, which again allows vehicle 4 to shorten its trajectory. However, a new SAM site R5 is also detected by vehicle 1, which results in a reallocation of tasks for the stealth vehicles. Vehicle 1 is now tasked with removing R5, while vehicle 2 receives R3 in addition to the previous task of removing R4. Again this shows coordination in the determination for the task assignment and trajectory planning. Note that vehicle 3 must also change trajectories to avoid the new SAM area.

The fourth plot (Figure 2.5(d)) occurs after all the remaining SAM sites have been removed. Vehicle 5 flies through a gap in no-fly zones, producing another exchange

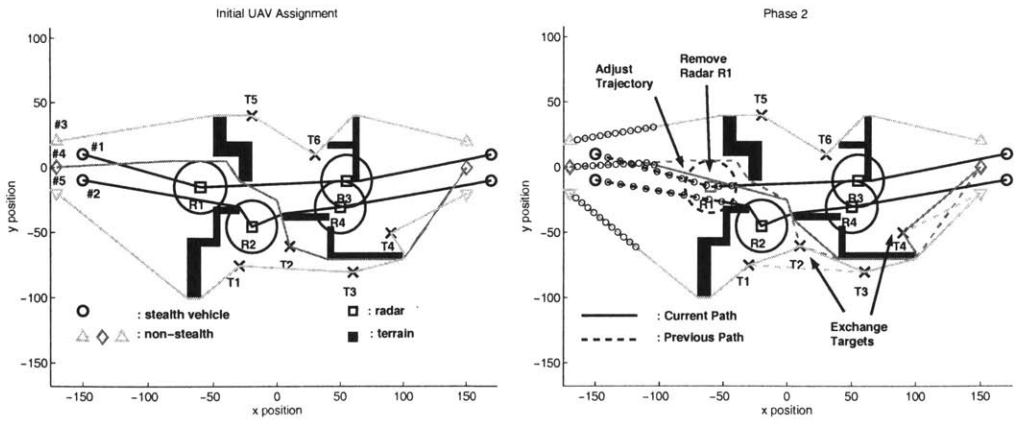
of target assignments with vehicle 4. Vehicle 4 is assigned T3, while vehicle 5 receives targets T1, T2, and T4. Vehicle 3 shortens its trajectory to fly below an obstacle once vehicle 2 removes SAM R3.

In Figure 2.5(e) a new target T7 is discovered by vehicle 5. Vehicle 5 is assigned the new target but trades T2 and T4 to vehicle 4 to make up for the increased cost in accepting the new target. The result is an allocation that still minimizes the mission time for the fleet. The final plot shows the trajectories flown by each vehicle throughout the optimization.

This simulation demonstrates the capability of the UAV task assignment algorithm presented in this chapter to adapt to changes in the environment. This simulation demonstrates how a complete reassignment of tasks leads to coordination between the vehicle in completing the total mission. This coordination is demonstrated by numerous exchanges of tasks to re-optimize the plan.

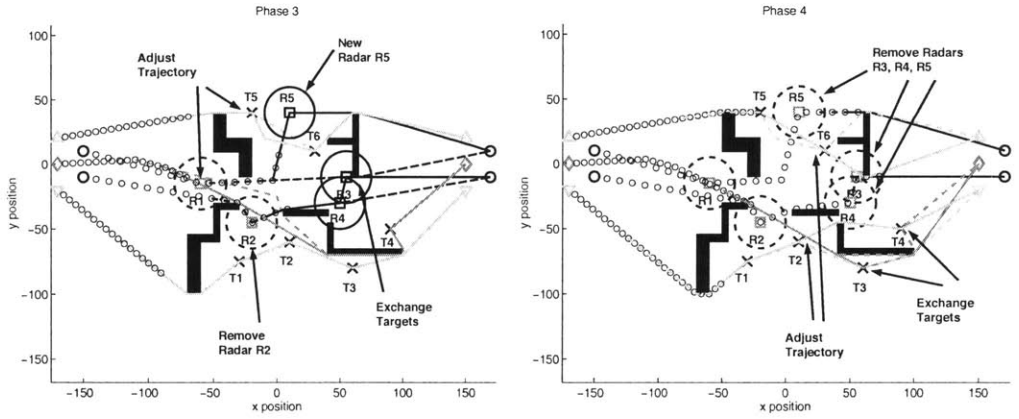
2.4.2 Optimal Value for Petal Size (m)

A very important parameter in designing the receding horizon task assignment (RHTA) algorithm is the maximum size of each petal, m in each iteration. The larger this value, the longer the paths that are generated in each iteration. Intuitively, as m increases, the performance and the computation time both increase. By setting m to a large enough value the solution of the petal algorithm can be achieved in one iteration. On the other hand, if $m = 1$ the algorithm will generate the same result as an iterated greedy algorithm. Thus, computation time is often sacrificed in order to gain better performance and vice versa. The RHTA algorithm, however, manages to work with a single parameter, m, and balances the computation time and performance. The objective is to find a value of m for which the algorithm generates a close-to-optimal answer in a reasonable amount of time. To experimentally find an optimal value for m, RHTA was applied to many randomly generated problems using different values for m. Table 2.1 illustrates the results of a set of simulations with 8 UAVs and 20 waypoints using four different values for m ($m = 1, 2, 3, 4$). As expected, as m increases, the accumulated value, or performance, increases as well



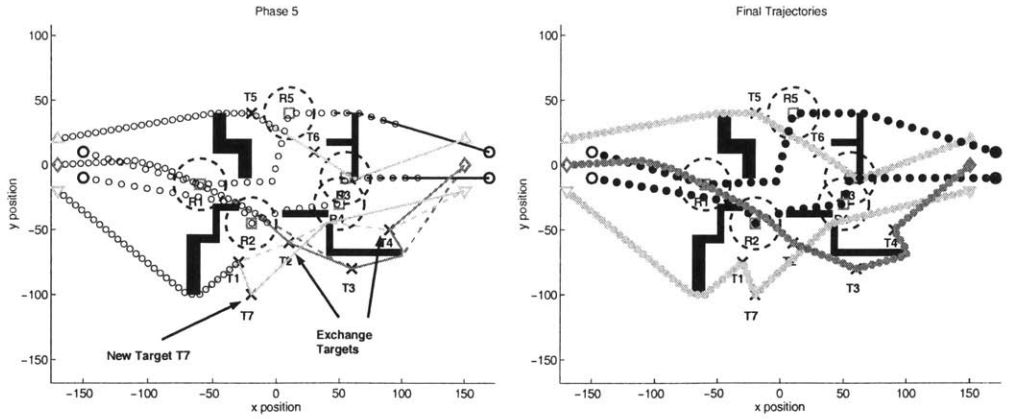
(a) Initial Plan

(b) Phase 2



(c) Phase 3

(d) Phase 4



(e) Phase 5

(f) Phase 6

Fig. 2.5: Dynamic Environment Simulation. The stealth vehicles, ∘ are capable of flying through and removing SAM sites noted with squares. The other vehicles are restricted from flying through SAM zones and must remove targets marked with ×'s.

as the computation time. The third column of the table shows the ratio of accumulated value (performance) to the best calculated accumulated value (the solution for $m = 4$). The solution for $m = 4$ and $m = 3$ are identical in performance, but the computation time is considerably higher when $m = 4$. Comparing $m = 1$ and $m = 2$ with $m = 3$, the performance is slightly better for $m = 3$ but the computation time is much longer. Based on this set of experiments, $m = 1$ and $m = 2$ are both good candidates in the sense that they can generate close-to-optimal solutions very quickly. The fourth and fifth columns of the table show the times when the last UAV finishes its mission (mission completion time) and the summation of all UAV finishing times. These two can also be used as a measure of performance. Comparing these values for different m, shows that $m = 2$ generates almost an identical solution to the best solution while $m = 1$ is much worse.

To see if the same results hold for other cases, two additional sets of simulations are used. Table 2.2 shows the results for many randomly generated scenarios with 8 UAVs and 30 waypoints. Note that in this case $m = 4$ is computationally intractable and therefore the results are just for $m = 1, 2, 3$. Table 2.3 illustrates the result of scenarios with 8 UAVs and 40 waypoints. The histograms in Figures 2.6, 2.7 compare the performance and the computation time respectively for the 3 values of m in this set of simulations. Both performance and computation time are normalized to their best values. These sets of results validate the conclusion that $m = 2$ is a good value for balancing performance and computation time.

Note that the value for computation time in these tables are the total computation time of all iterations. In real-time replanning where the environment is dynamic, the algorithm does not need to generate a complete plan at each replanning as the part of the plan for the future will change as the environment changes. Therefore in each replanning only one or two iterations of the algorithm are enough. This will reduce the computation time considerably and makes it feasible for real-time implementation.

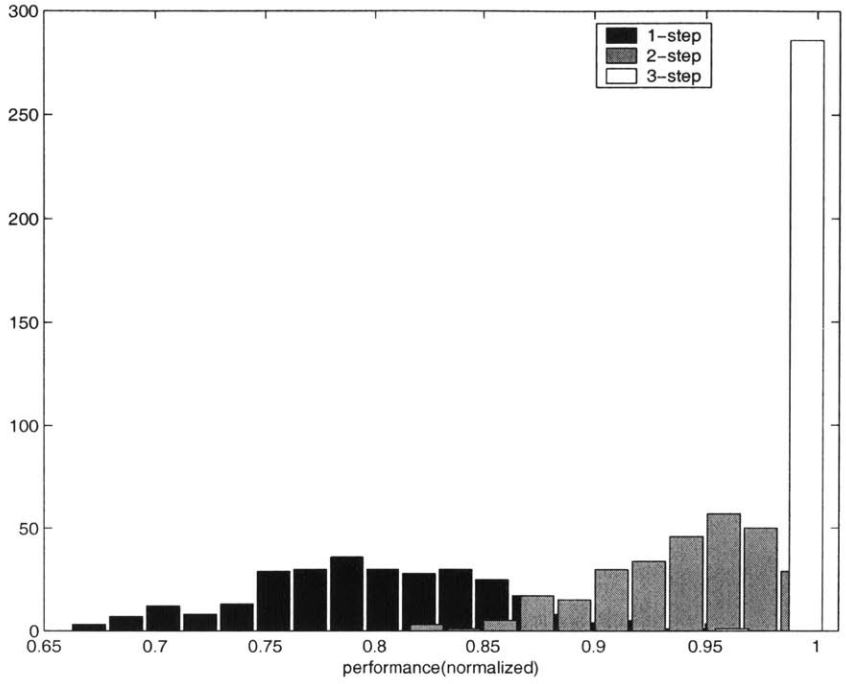


Fig. 2.6: Comparing performance for different values of m. The performance is the accumulated values. Here the accumulated values are normalized with respect to the best accumulated value.

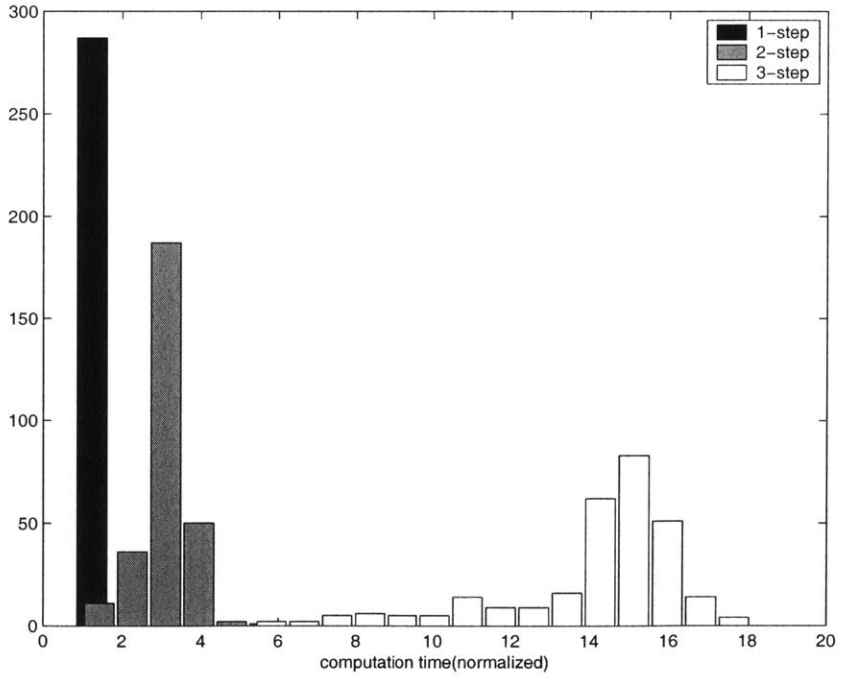


Fig. 2.7: Comparing computation time for different values of $m = (1, 2, 3)$. The computation times are normalized with respect with the smallest computation time for each instance.

Table 2.1: Simulation results for 8 UAVs and 20 waypoints

Petal Size (m)	Accumulated value	Ratio to the Best Solution	Mission Time	Total Finishing Times	Computational Time (sec)
1	580	0.954	15	84	0.4
2	606	0.997	12	74	.6
3	608	1.0	12	73	2.8
4	608	1.0	12	73	12.2

Table 2.2: Simulation results for 8 UAVs and 30 waypoints

Petal Size (m)	Accumulated value	Ratio to the Best Solution	Mission Time	Total Finishing Times	Computational Time (sec)
1	660	0.970	18	96	0.5
2	677	0.996	11	79	1.9
3	680	1.0	11	75	13.1

Table 2.3: Simulation results for 8 UAVs and 40 waypoints

Petal Size (m)	Accumulated value	Ratio to the Best Solution	Mission Time	Total Finishing Times	Computational Time (sec)
1	813	0.925	21	127	0.5
2	869	0.989	14	101	4.1
3	879	1.0	14	92	66.4

2.5 Conclusions

In this chapter we proposed a new approach for fast UAV task assignment problem that maintains a high degree of optimality. The concept of Model Predictive Control was implemented in the task assignment algorithm to decrease the complexity of the optimization problem by limiting the size of the problem. The resulting iterative method, which we call the Receding Horizon Task Assignment (RHTA) was shown to perform well for large problems in dynamic environments. It was shown to be suitable for the dynamic environment in which real-time planning is crucial. It was also shown that the planning horizon m was an important parameter that can be tuned to balance the performance and computation time. The results of many different simulations showed that $m = 2$ achieves the best balance between computation time and performance for large problems.

Chapter 3

Filter-Embedded Task Assignment

3.1 Introduction

Future autonomous vehicles will be required to successfully operate in inherently dynamic and uncertain environments [3, 4]. The vehicles will be required to make both low-level control decisions, such as path planning, and high-level decisions, such as cooperative task assignment, based on uncertain and noisy information. While the impact of uncertainty on feedback control has been analyzed in detail in the controls literature, equivalent formulations to analyze this impact on the high-level planning processes have only recently been developed [31, 37]. Uncertainty will inherently propagate down from the high-level decisions to the lower-level ones, and thus it is very important to extend these tools and algorithms to provide new insights on the behavior of these real-time higher-level guidance and control algorithms in the face of uncertainty.

Task assignment *in the controls literature* has been generally viewed as an open-loop optimization with deterministic parameters. The optimization is generally done once (possibly made robust to uncertainty in the problem [38, 39, 40]), and task reassignment occurs only when substantial changes in the environment have been observed (e.g., UAV loss or target classification [42, 13]). In reality, these information updates are continuously occurring throughout the mission due to vehicle sensing capabilities, adversarial strategies, and communicated updates of situational awareness

(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 R. Kastner, *et.al.* [30] in their use of incremental algorithms for combinatorial auctions. The authors propose that the perturbed optimization problem should also include a term in the objective function that penalizes changes from the original solution. The work of J. Tierno and A. Khalak [31] also investigates the impact of replanning, with the objective function being a weighted sum of the current objective function and the plan difference from the previous optimization to the current one. Both of these formulations rely on the plan generated prior to the current one as a reference. They do not consider the impact of noise in the problem, nor do they develop techniques to mitigate its effect on replanning.

The objective of this chapter is to develop a modified formulation of the task assignment problem that mitigates the effect of noise in the SA on the solution. The net effect will be to limit the rate of change in the reassignment in a well defined manner. The approach here is to perform reassignment at the rate that information is updated, which enables immediate reaction to any significant changes in the environment. We demonstrate that the modified formulation can be interpreted as a noise rejection algorithm that can be tuned to reduce the effect of variation in the uncertain parameters in the problem. Simulations are then presented to demonstrate the effectiveness of this algorithm.

3.2 Problem Statement

Consider the general weapon target assignment (WTA) problem expressed as a linear integer program (LIP). The following optimization can be solved to generate a plan, x_k at time k,

$$\begin{aligned} \max_{x_k} \quad & c_k^T x_k \\ \text{s.t.} \quad & x_k \in \mathcal{X}_k \\ & x_k \in \{0,1\}^N \end{aligned} \tag{3.1}$$

where $c_k \in \mathcal{R}^N$ is the cost vector and x_k is a vector of binary variables of size N. $x_k(i)$ is equal to one if target i is selected in the assignment at time k, and zero otherwise. Here $\mathcal{X}_k$ denotes the invariant feasible space for x_k. This space could represent general constraints such as limits on the total number of vehicles assigned to the mission.

Targets are assumed to have a value c_k, and the problem becomes one of selecting the "optimal" targets to visit subject to the afore mentioned constraints. In the deterministic formulation, the solution becomes a sorting problem, which can be solved in polynomial time. From a practical standpoint, these target values are uncertain as they could be the result of classification, battle situational awareness of the vehicle, and other a priori information. Furthermore, these uncertain values are likely to change throughout the course of the mission, and 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 the task assignment equivalent of a "high bandwidth controller", making it susceptible to tracking the sensor noise. From the perspective of a human operator, continuous reassignments of the vehicles in the fleet may also prove to be undesirable, especially if this effect is due primarily to sensing errors. Furthermore, since the optimization is continuously responding to new information, it is likely that the integer constrained assignment will vary continuously in time, resulting in a "churning" effect in the assignment, as observed in [41]. The noise tracking and churning features are undesirable both from a control and human operator perspective.

A simple example of churning is shown in Figure 3.1, 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

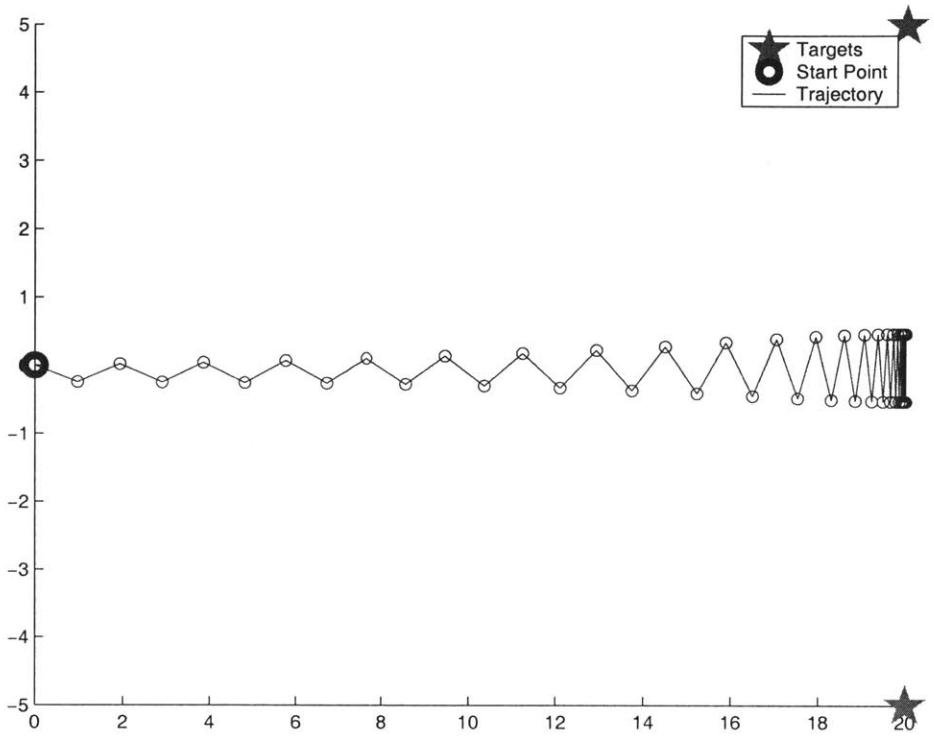


Fig. 3.1: Effect of churning on a simple assignment problem

assignment switches once again. The switching throughout the course of the mission is an extreme behavior of the churning phenomenon. In fact, it can be seen that as the mission progresses, the vehicle is still alternating the targets to visit, and never converges to an assignment that does not change. While this is a dramatic example of churning, it captures the notion that sensing noise alone could cause a vehicle to switch assignments throughout the course of the mission. Similar behavior was observed by Tierno [43] as a result of modeling errors in the cost calculations for the task assignment.

Clearly, Figure 3.1 shows an extreme situation. However, likely missions will involve multiple vehicles, each with unique information and independent sensors and noise sources. It might be quite difficult to identify and correct for churning behavior in a large fleet of UAVs. The subsequent sections in this chapter present methods of modifying the general task assignment problem to automatically avoid this phenomenon.

3.3 Frequency Domain Analysis

The general assignment problem was introduced in Eq. 3.1. Note that the main issue was that, even with small variations in the target values, the assignment problem results in highly differing solutions. Clearly, constraining the rate of change of the assignment would result in an attenuation of the churning phenomenon. As noted in [31], a key issue is which metric to use to measure the change in the assignment. There, the authors suggested a connection between time and frequency domains with regards to planning systems. In this section those ideas are extended by formally relating the time and frequency domain specifications for the assignment problem.

The approach proposed here is similar to that used in GPS (global positioning system) which uses correlation techniques to compare received satellite signals and locally generated Gold codes. A similar approach can be used if we consider the solution x_k as a length N binary "code". We restrict attention to a specific instance of Eq. 3.1, in which m vehicles are to assigned to N targets ($m < N$). Thus the following integer program is obtained:

$$\begin{aligned} \max_{x_k} \quad & c_k^T x_k \\ \text{s.t.} \quad & \sum_{i=1}^{N} x_k(i) = m \\ & x_k \in \{0,1\}^N \end{aligned} \tag{3.2}$$

First note that each solution assigns m vehicles to N targets, so they have the same auto-correlation:

$$x_{k-1}^T x_{k-1} \equiv x_k^T x_k \equiv m \tag{3.3}$$

The two solutions are then compared using the *cross-correlation*

$$R_{k-1,k} = x_{k-1}^T x_k \tag{3.4}$$

where $R_{k-1,k}$ provides a direct measure of the changes that have been made to the plan from one time-step to the next. At time-step k, if a previous solution x_{k-1}

already exists, then a constraint that limits the number of changes that are allowable in the current solution can be included:

$$\max_{x_k} \quad c_k^T x_k \tag{3.5}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} x_k(i) = m \tag{3.6}$$

$$x_{k-1}^T x_k \geq m - \alpha_k \tag{3.7}$$

$$x_k \in \{0,1\}^N \tag{3.8}$$

where the integer α_k indicates the number of changes that are allowed in the new plan.

One difficulty with this approach is that the new problem can become infeasible. This infeasibility can be avoided by converting the constraint in Eq. 3.7 to a soft constraint where α_k is a parameter that must be optimized

$$\max_{x_k,\alpha_k} \quad \begin{bmatrix} c_k^T & -\beta \end{bmatrix} \begin{bmatrix} x_k \\ \alpha_k \end{bmatrix} \tag{3.9}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} x_k(i) = m \tag{3.10}$$

$$\begin{bmatrix} -x_{k-1}^T & -1 \end{bmatrix} \begin{bmatrix} x_k \\ \alpha_k \end{bmatrix} \leq -m \tag{3.11}$$

$$0 \leq \alpha_k \leq \Gamma \tag{3.12}$$

$$x_k \in \{0,1\}^N \tag{3.13}$$

where Γ represents an upper bound on the rate of change of the plan and β represents the relative cost of making changes to the plan. The key point in this formulation is that Γ represents the largest possible decorrelation of the plan from one time-step to the next. Therefore

$$R_{0,1} \geq m - \Gamma$$

$$R_{0,2} \geq m - 2\Gamma$$

$$\vdots \tag{3.14}$$

$$R_{0,k} \geq m - k\Gamma$$

Note that since variables are binary, the cross correlation must remain positive semi-definite, and we must ensure that $R_{0,j} \geq 0 \ \forall j$. To demonstrate the above statements (Eq. 3.14), define the plan $P_k = x_k$ and the change in the plan, ΔP_{k+1} as the difference between P_{k+1} and P_k

$$P_1 = P_0 + \Delta P_1 \tag{3.15}$$

$$P_2 = P_1 + \Delta P_2 \tag{3.16}$$

Then, by taking the cross-correlation between plans P_0 and P_1,

$$\begin{aligned} P_0^T P_1 = P_0^T (P_0 + \Delta P_1) &\geq m - \Gamma \\ \Rightarrow P_0^T (\Delta P_1) &\geq -\Gamma \end{aligned} \tag{3.17}$$

Likewise, for the cross-correlation between plans P_0 and P_2, by substituting the above result,

$$P_0^T P_2 = P_0^T (P_0 + \Delta P_1 + \Delta P_2) \geq m - \Gamma + P_0^T \Delta P_2. \tag{3.18}$$

Now consider the case with $\Delta P_1^T \Delta P_2 = 0$ (implying that the difference between two plans is orthogonal), therefore

$$P_1^T \Delta P_2 = (P_0 + \Delta P_1)^T \Delta P_2 = P_0^T \Delta P_2 \geq -\Gamma \tag{3.19}$$

Combining these results,

$$P_0^T P_2 \geq m - \Gamma + P_0^T \Delta P_2 \geq m - 2\Gamma \tag{3.20}$$

The above can be extended by induction to any value of k, giving the result that

$$R_{0,k} \equiv P_0^T P_k \geq m - k\Gamma \tag{3.21}$$

This corresponds to a triangular correlation plot that is of the form of the *Bartlett Window* typically used in lag windows [44]. Insights into the frequency content of the equivalent controller can be developed by converting the linear correlation plot into the frequency domain via the Fourier transform. This conversion is straightforward since the correlation plot is equivalent to the Bartlett window

$$w_{\tau,n} = \begin{cases} 1 - |\tau|/n, & |\tau| < n \\ 0, & |\tau| \geq n \end{cases} \tag{3.22}$$

Here n is the *window parameter*; we then have

$$W_n(f) = \Delta t \sum_{\tau=-n}^{n} w_{\tau,n} e^{-i2\pi f \tau \Delta t} = \frac{\Delta t}{n} \left(\frac{\sin(n\pi f \Delta t)}{\sin(\pi f \Delta t)} \right)^2 \tag{3.23}$$

There are various measures of the bandwidth of $W_n(f)$, one being $\beta_W = 1.5/(n\Delta t)$ [44]. In this case, $n = \text{ceil}(m/\Gamma)$ and $\Delta t = T_s$, so

$$\beta_W \approx \frac{1.5}{((m/\Gamma)T_s)} = \frac{1.5\Gamma}{mT_s}$$

which clearly shows that increasing Γ (the maximum decorrelation rate) increases the effective bandwidth of the controller, as might be expected. A typical plot for this conversion is shown in Figure 3.2. This analysis establishes an explicit link between the time and frequency domains for the task assignment problem, a relation that has been exploited in many other areas of control design and analysis because it is often insightful to discuss the control problem in the time domain, and modeling errors and sensing noise in the frequency domain.

3.4 Filter Design

Section 3.3 introduced time and frequency domain interpretations of the task assignment problem. In this section, the correlation concept introduced in the Section 3.3 is extended to develop a filter for the assignment problem. This filter rejects the effect

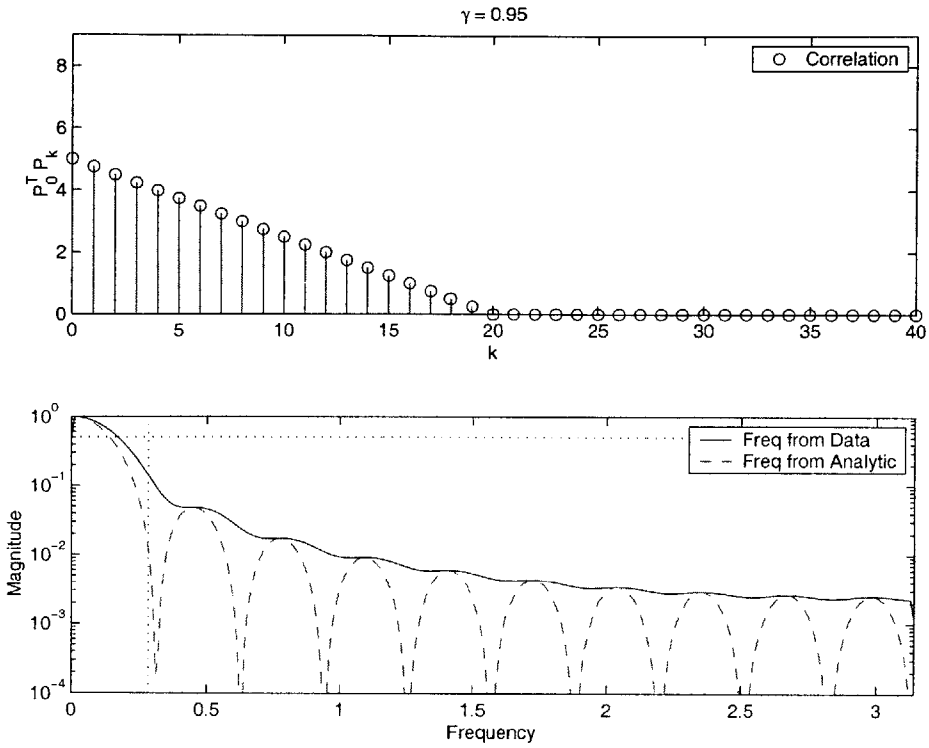


Fig. 3.2: Equivalent frequency response of the planning system with a constraint on the correlation with the previous plan. The control system is effectively a low-pass filter, with a bandwidth proportional to $\Gamma/(mT_s)$, shown by the vertical dashed line.

of noise in the information (parameter) vector and can be tuned to capture different noise frequencies.

3.4.1 Binary Filter

Similar to section 3.3, assume that the plan is a binary vector of size N. Also assume that the *length* of this vector stays constant in each replanning. Define a binary filter of size r, a system whose binary output changes at most with the rate of once every r time steps. Figures 3.3 shows the input and output to two binary scalar filters with lengths $r = 3$, $r = 5$. As illustrated in Figure 3.3-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 3.3-middle), and every 5 steps (Figure 3.3-bottom). These figures show the filter for a single binary value, but

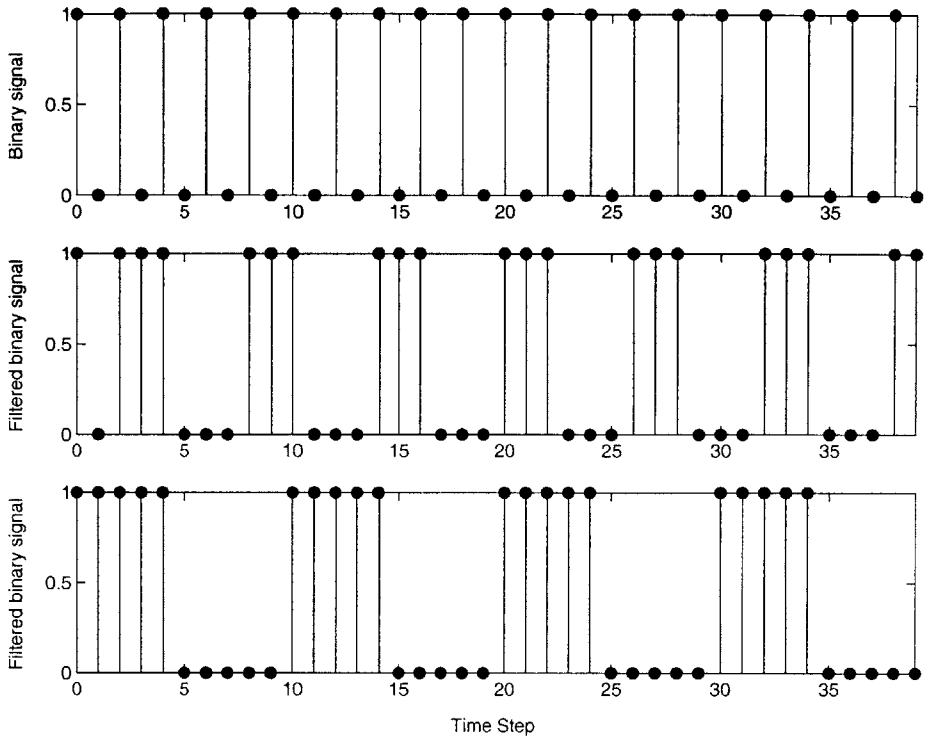


Fig. 3.3: (top): Input to the binary filter; (middle): Filtered signal ($r = 3$); (bottom): Filtered signal ($r = 5$)

the same idea can be extended to a binary vector.

Now we present equations that result in a binary filter of size r. Let us define a single binary input signal to the filter at time k, x_k, and a filtered output signal y_k. Define $\delta y_{k,i}$ as the changes in the value of y in the past iterations:

$$\delta y_{k,i} = \begin{cases} x_k \oplus y_{k-1} & i = 1 \\ y_{k-i} \oplus y_{k-i+1} & i = 2, \ldots, r \end{cases} \tag{3.24}$$

where $\oplus$ represents the exclusive OR (XOR) operation. Having these differences, define z_k as follows:

$$z_k = \delta y_{k,1} + \frac{1}{r} \sum_{i=2}^{r} \delta y_{k,i} \tag{3.25}$$

z_k is thus a weighted summation of the difference in plans from time $k - r$ to k.

Now if $z_k > 1$, then y has changed at least once in the previous r steps and x_k is different from y_{k-1}; therefore if $z_k > 1$, y_k should equal $y_{k-1} = (\sim x_k)$ and $y_k = x_k$

otherwise. y_k can be calculated as follows:

$$y_k = \begin{cases} x_k & \text{if} \quad z_k \leq 1 \\ \sim x_k & \text{otherwise} \end{cases} \tag{3.26}$$

where $\sim$ denotes the *NOT* operation. Having defined the binary filter, the following sections demonstrate how to implement it inside the planning algorithm.

3.4.2 Assignment With Filtering: Formulation

Starting with the simple assignment problem of Eq. 3.1; the idea here is to replan for the same system in each time step and suppress the effect of parameter noise (uncertainty) on the output of the system (generated plan). If the above problem is solved in each iteration, the optimization parameters will be directly impacted by the noise and the assignment can be completely different at each step. To avoid large variations in the solution, a binary filter is integrated into the problem to limit the rate of change in the assignment problem.

The modified assignment problem can be written as follow:

$$\max_{y_k, z_k} c_k^T y_k \tag{3.27}$$

$$\text{s.t.} \quad y_k \in \mathcal{Y}_k \tag{3.28}$$

$$z_k = y_k \oplus y_{k-1} \tag{3.29}$$

$$t_{k,i} = z_k^T \delta y_{k,i} = 0 \qquad i = 2, \ldots, r \tag{3.30}$$

where $\delta y_{k,i}$ represents the changes in the assignment in previous plans and functions as an input to the optimization problem:

$$\delta y_{k,i} = y_{k-i} \oplus y_{k-i+1} \quad i = 2, \ldots, r \tag{3.31}$$

Note that $\delta y_{k,i}$ can be calculated prior to optimization, as the previous plans have been stored. Also note that the constraint in Eq. 3.30 restricts changes in the current plan from the previously generated plans.

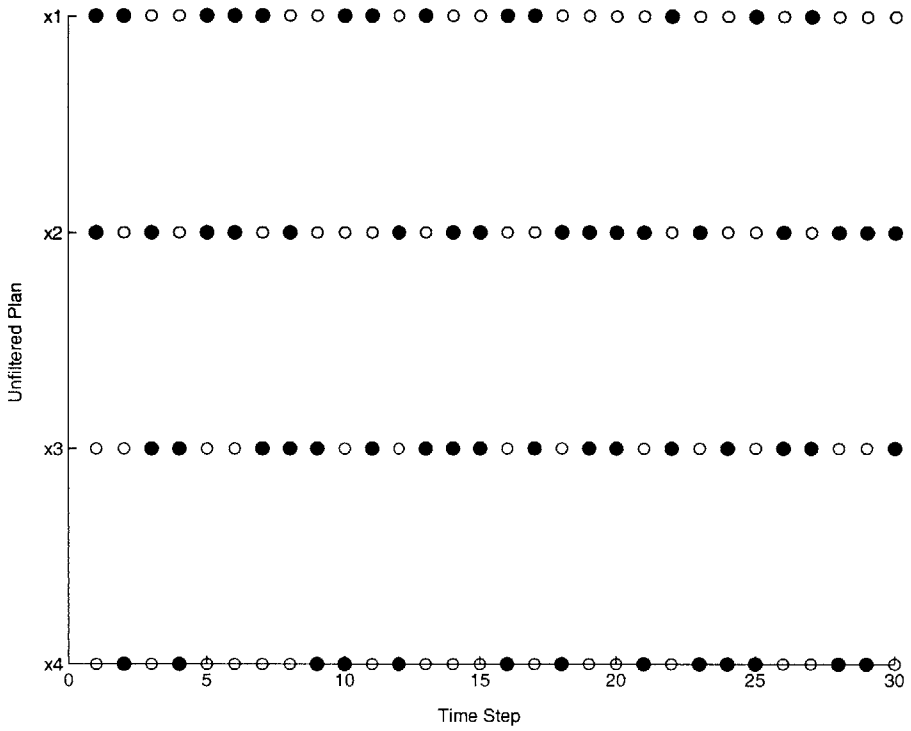


Fig. 3.4: Unfiltered plan (*i.e.*, optimal plan) showing which two of the four targets are chosen at each time step.

3.4.3 Assignment With Filtering: Implementation

This section presents the result for a simple but general assignment problem and compare the results of the unfiltered and filtered formulations. The objective of this problem is to pick m targets from N existing targets ($m < N$) in order to maximize the total value of the selected targets. Each target has a value associated with it that is affected by noise:

$$c_k = c + \delta c_k \tag{3.32}$$

where c is the nominal target value and δc_k is the noise added to these values. The nominal value for all targets is set to 5. Solving this problem for $N = 4$, $m = 2$ for 30 iterations results in 30 different plans which are directly affected by the noise. Figure 3.4 shows the result of this simulation. In the following figures, filled circles represent 1 and unfilled circles represent 0. Thus, targets 1 and 2 are selected in assignment 1, targets 1 and 4 are selected in assignment 2, etc.

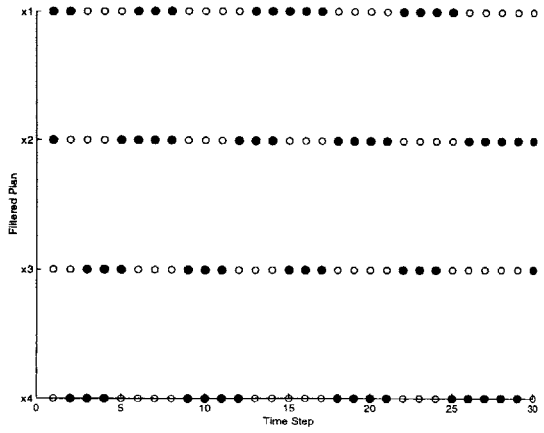


Fig. 3.5: Filtered plan with $r = 3$

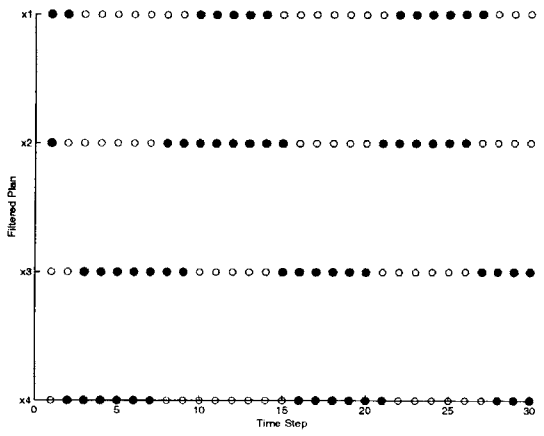


Fig. 3.6: Filtered plan with $r = 5$

The filter is implemented by converting the constraint in Eq. 3.29 to linear form for LIP implementation:

$$z_k(i) \geq -y_{k-1}(i) + y_k(i) \tag{3.33}$$

$$z_k(i) \geq y_{k-1}(i) - y_k(i) \tag{3.34}$$

$$z_k(i) \leq y_k(i) + y_{k-1}(i) \tag{3.35}$$

$$z_k(i) \leq 2 - y_k(i) - y_{k-1}(i) \tag{3.36}$$

$$z_k(i) \in \{0, 1\} \quad i = 1, \ldots, N \tag{3.37}$$

One potential issue with this formulation, is that it might make the problem infeasible. This means that if the basic problem has a solution, a solution is not guaranteed in the filtered formulation. The constraint that is capable of making the filtered formulation infeasible is $t_{k,i} = 0$, $i = 2, \ldots, r$. This limits the number of bit changes in the plan compared to previous plan and can make the problem infeasible. To avoid this difficulty, this set of constraints can be relaxed and added to the cost function as a penalty function. The new cost function can be written as:

$$\max_{y_k, t_{k,i}} c_k^T y_k + \sum_{i=1}^{r} d_i t_{k,i} \tag{3.38}$$

Figures 3.5 and 3.6 show the results of applying filtering to the previous example,

with values of $r = 3$, $r = 5$ used for the filter length. Comparing these results with the result of the unfiltered formulation, we can see noise mitigation in the filtered formulation. Although this result helps us to somehow attenuate the noise effect and reduce the rate of change in the plan, it does not completely reject noise as expected. The filter clearly prevents sudden changes in the plan; a change in the i^{th} element of x_k, $x_k(i)$, can happen only after being in its current state for r time steps. The ultimate goal is to make the filter respond to low frequency changes and suppress high frequency changes, which will help us to reject the high frequency noise while responding to low frequency changes in the environment (system). This will reduce churning which is the result of noise and/or the changes in the environment that occur too rapidly to track, and may therefore be treated as noise as well. In the above formulation, the filter has memory which allows it to use the previous outputs to generate the current output. The solution can be written as:

$$y_k = f(x_k, y_{k-1}, y_{k-2}, \ldots, y_{k-r}) \tag{3.39}$$

where x_k is the current optimal solution (interpreted as the input to the system) and $y_{k-1}, \ldots, y_{k-r}$ are the previous plans (outputs of the system). In addition to these values, the previous unfiltered plans can also be used as input to the filter. At each iteration, both filtered, y_k, and unfiltered, x_k, plans can be generated. Having generated x_k, a filter of the following form can then be designed :

$$y_k = f(x_k, x_{k-1}, \ldots, x_{k-q}, y_{k-1}, y_{k-2}, \ldots, y_{k-r}) \tag{3.40}$$

Figure 3.7 gives a block diagram representation of assignment with filtering. Here, FTA and $UFTA$ represent the filtered and unfiltered task assignments respectively. TA represents the overall task assignment and Z^{-1} represents a bank of unit delays.

To explain how this can reduce the impact of high frequency noise in the parameters, suppose the objective is to reject noise with frequency 1 but also track changes occurring with frequency of 0.5. This means the effect of noise makes the system parameters change at each time step, while the changes in the parameters are oc-

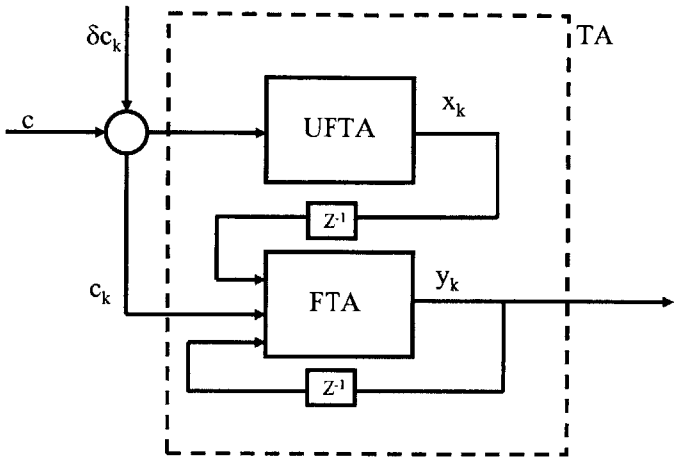


Fig. 3.7: Block diagram representation of assignment with filtering

curring every two time steps. Now consider at time k the unfiltered solution to two previous plans: x_{k-1} and x_{k-2}. In order to calculate y_k given x_k and these values, the difference between x_{k-2}, x_{k-1}, and x_k are used:

$$\delta x_k = x_{k-1} \oplus x_k \quad \text{and} \quad \delta x_{k-1} = x_{k-2} \oplus x_{k-1}$$

Then calculate the output as

$$y_k(i) = \begin{cases} x_k(i) & \text{if} \quad \delta x_k(i)\delta x_{k-1}(i) = 0 \\ y_{k-1}(i) & \text{if} \quad \delta x_k(i)\delta x_{k-1}(i) = 1 \end{cases} \tag{3.41}$$

If the value of the bit $x_{k-1}(i)$ had changed from its previous value,$x_{k-2}(i)$ and the value of the same bit is changed from $x_{k-1}(i)$ to $x_k(i)$ then it means that this change is an effect of changes with the frequency of 1 which is intended to be canceled. Therefore the change is ignored and $y_k(i) = y_{k-1}(i)$. Now this idea of filtering noise is used to implement an assignment algorithm that is robust to high frequency noise in the environment. In other words the effect of noise in the information vector is suppressed in the assignment. To avoid the difficulties of adding hard constraints to the assignment problem, the constraints are relaxed and added to the objective function. A filter that rejects noise with frequencies greater than $\frac{1}{2^q}$ can be formulated

as follows:

$$\max \quad c_k^T y_k - \beta_k^T (y_k \oplus y_{k-1}) \tag{3.42}$$

$$\text{s.t.} \quad y_k \in \mathcal{Y}_k \tag{3.43}$$

$$\delta x_j = x_{k-j} \oplus x_{k-j+1} \tag{3.44}$$

$$\beta_k = \sum_{j=1}^{q} b_j \delta x_j \tag{3.45}$$

where b_j is the weighting coefficient and x_j is the unfiltered solution at step j and is given as an input to the optimization. The second term in the objective function (Eq. 3.42) is the penalty associated with changes that are made in the current plan compared to the previous plan. β_k is the coefficient vector that penalizes each change (each bit in the assignment vector). Each element of this coefficient vector is a weighted summation of previous changes in the optimal plan. Since the optimal plan will track environmental noise, measuring changes in the optimal plan results in a good metric to identify and suppress such disturbances. This is implemented in Eqs. 3.44–3.45. The coefficients b_j tune the effect of previous plans in the current plan. By setting these coefficients, we define the importance of the changes in the previous plans and so the *bandwidth* of the filter. A good candidate for b_j is

$$b_j = \frac{b}{2^j} \tag{3.46}$$

where b is a constant that can be set based on the problem. This set of coefficients will attenuate the effect of the changes that happened in the far past (larger j), compared to the recent changes in the plan. As j increases then, the weighting on the past plans is decreased.

The formulation above is a special case of the general filter in Eq. 3.40 with $r = 1$, which was described for simplicity. A more comprehensive form of this filter ($r > 1$) can also be implemented to obtain more general filtering properties. For instance, by combining the formulation given above with the formulation in the previous section, a filter with the properties of both can be generated. This filter will reject noise with

high frequency while limiting the rate of change of the plans. Current research is investigating the best way to include a more general form of assignment filter (binary filter) in the task assignment problem.

3.4.4 Assignment With Filtering: Simulation Results

Figure 3.8 present the results of the unfiltered and filtered formulations for the example introduced in section 3.4.3. The cost coefficients change randomly with time (top plot of Figure 3.8). The noise is uniformly distributed in the interval $[-0.5, 0.5]$.The middle plot shows that the unfiltered plan tracks the noise in the cost coefficients to a much greater extent than the filtered plan (bottom plot). To demonstrate that the filtered plan is only rejecting the "noise", the coefficient c_2 is changed at time step 7 by increasing it by 0.7 and at time step 16 by decreasing it by 1.4; the results show that the filtered plans were modified to follow these lower frequency changes.

Figure 3.9 makes another comparison between the unfiltered and filtered assignments. An example with 40 targets and 20 vehicles was simulated for 100 time steps. At each time step, the current plan was correlated with the previous plan, and the results are shown in the plot. As expected, the filtered plans exhibit much higher correlations than the unfiltered plans.

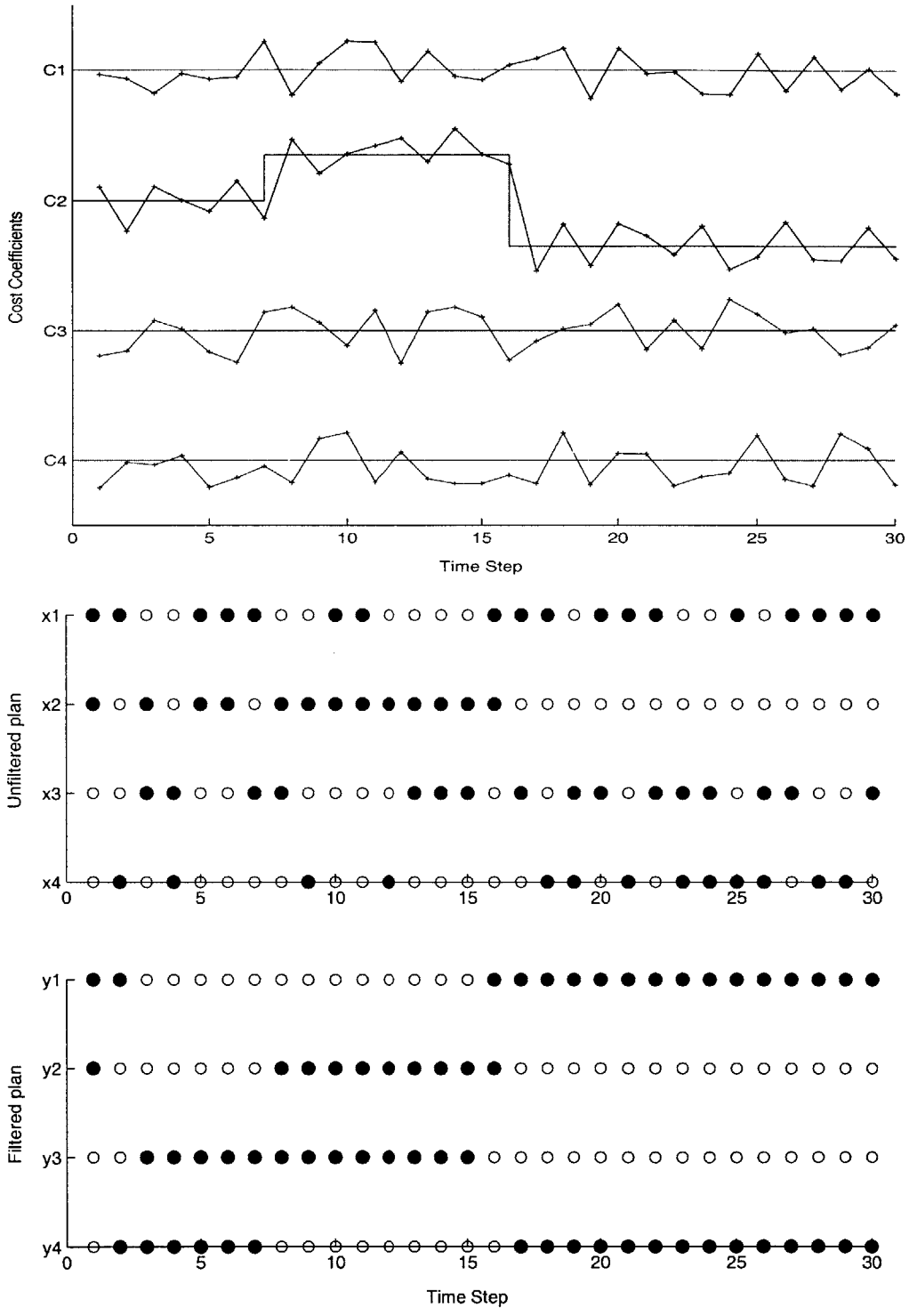


Fig. 3.8: (top):Noisy cost coefficients; (middle): Plans with no filter; (bottom): Filtered plan

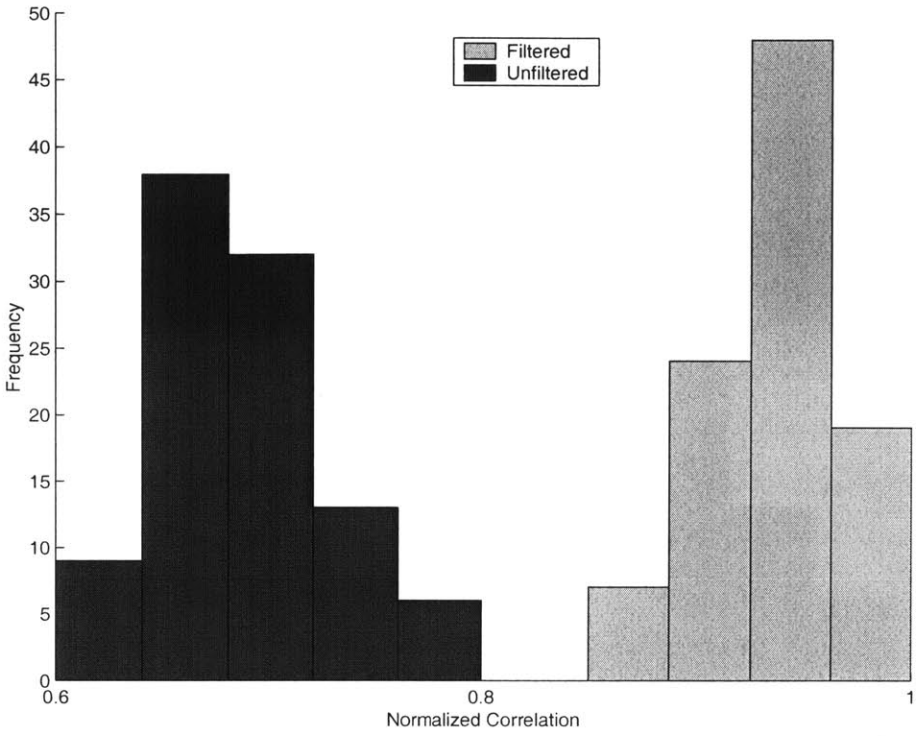


Fig. 3.9: Histogram showing correlation of filtered and unfiltered assignments

3.5 Conclusions

This chapter has formulated a modification of the classical task assignment under noisy conditions. We have extended the frequency domain interpretation as originally formulated in [31] using correlation as a metric. A formulation that limits the rate of change of the assignment to mitigate the churning phenomenon is developed . To attenuate the effect of noise in the assignment problem, we have developed a noise rejection scheme that is incorporated in the planning algorithm. We demonstrated the effectiveness of this scheme in simulation, and showed good signal-tracking and noise-rejection properties. Future work will investigate the role of robust approaches to the task assignment and their relation to the recently developed noise rejection algorithms.

Chapter 4

Cooperative Planning

4.1 Motivation

Previous chapters addressed the problem of UAV task assignment for cases where the environment is dynamic. Chapter 2 proposed a Receding Horizon Task Assignment (RHTA) algorithm as a method for fast replanning of the problems in which the environment changes rapidly. Chapter 3 addressed high rate replanning issues and proposed a filtering algorithm to eliminate these problems. The algorithms in the previous chapters generate a coordinated assignment for each team of UAVs, assigning tasks to each UAV to maximize an overall objective. The plan is coordinated since each UAV does not maximize its own objective, but gets assigned to optimize the overall objective. The timing of the planning (*i.e.,* a UAV performs a BDA task after another UAV has performed strike on a specific target) shows coordination of all the activities. In UAV planning, coordination is essential for optimizing the team objective and is achieved in the algorithms that were proposed in previous chapters. However coordination is not sufficient for many problems and UAVs must also *cooperate* to accomplish certain missions. Here we define a cooperative plan as one in which the action of one agent (here, a UAV) directly improves the performance of another agent. The plan for each UAV in a cooperative plan is tightly coupled with the plan of other UAVs, for example some of the actions of UAVs are planned to create a shorter path with less risk for other UAVs. The following sections formulate this

concept in different ways for different applications.

4.2 Cooperative UAV Task Assignment in Risky Environments

Real-world air operations planners rely on cooperation between aircraft in order to manage the risk of attrition. Missions are scheduled so that one group of aircraft opens a corridor through anti-aircraft defenses to allow a second group to attack higher value targets, preserving their survival. The main planning challenges involve utilizing the maximum integrated capabilities of the team, especially when each UAV can perform multiple functions (*e.g.*, both destroy anti-aircraft defenses and attack high value targets). Cooperation is not just desirable, it is crucial for designing successful missions in heavily defended environments. A successful method of performing task allocations cannot assume the mission will always be executed as designed, but must account for an adversary in the environment who is actively attempting to cause failure. The simulations presented show that ignoring the probability of UAV loss results in mission plans that are quite likely to fail. Furthermore, techniques that model this probability [45, 28], but ignore its coupling to each UAV's mission can result in very poor performance of the entire fleet.

Clearly, a UAV mission planning formulation must recognize the importance of managing UAV attribution, and have the capability to use the same strategies as real-world air operations planners. The new formulation in this section approaches this by capturing not only the value of the waypoints that each UAV visits and of returning the UAV safely to its base, but also by capturing the probability of these events. In order to maximize the mission value as an expectation, this stochastic formulation designs coordination plans that optimally exploit the coupling effects of cooperation between UAVs to improve survival probabilities. This allocation recovers planning strategies for air operations and provides significant improvements over prior approaches [45, 28].

4.2.1 Stochastic Formulation

This section discusses the extension of the petal algorithm (discussed in Section 2.2) in order to capture the stochastic effect of the environment and achieve a plan with cooperation between UAVs. The constraints presented in Section 2.2 are applied to force the following variables to take their desired values. V_{wvp} equals 1 if waypoint w is visited by vehicle v on its p^{th} permutation and 0 if not, t_{w} is the time that waypoint w is visited, $t_{0\mathrm{v}}$ is the aircraft's time of departure from its starting point, P_{dvwp} equals 1 if the d^{th} destination visited by permutation p for vehicle v is waypoints w and 0 if not, and T_{dv} is the length of time after its departure that vehicle v visits its d^{th} waypoint.

A new stochastic optimization formulation will be presented that maximizes the expected value. This optimization exploits phasing by attacking the anti-aircraft defenses before the high value targets, and preserves the survival of the UAV that visits the high value target. To determine whether an anti-aircraft defense is in operation while a UAV flies within its original range, the waypoint visitation precedence is evaluated. If the time that UAV v begins the leg leading to its d^{th} destination is less than the time waypoint w is visited, then waypoint w is considered to threaten the UAV on this leg from $d-1$ to d, and the binary decision variable A_{dvw} is set to 1 to encode this waypoint visitation precedence. The logical equivalence

$$A_{\mathrm{dvw}} = 1 \Leftrightarrow t_{0\mathrm{v}} + T_{(\mathrm{d}-1)\mathrm{v}} \leq t_{\mathrm{w}} \tag{4.1}$$

can be enforced with the constraints

$$\begin{aligned} t_{0\mathrm{v}} + T_{(\mathrm{d}-1)\mathrm{v}} &\leq t_{\mathrm{w}} + M(1 - A_{\mathrm{dvw}}) + \epsilon \\ t_{\mathrm{w}} &\leq t_{0\mathrm{v}} + T_{(\mathrm{d}-1)\mathrm{v}} + M(1 - A_{\mathrm{dvw}}) + \epsilon \end{aligned}$$

where ϵ is a small positive number and M is a large positive number. With this precedence information available, constraints can be formulated to evaluate the probability q_{dv} that vehicle v survives to visit the d^{th} waypoint on its mission. The probability

$\tilde{q}_{\mathrm{dvw}}$ of vehicle v not being destroyed on the leg leading to its d^{th} destination by an intact air defense at waypoint w for the selected permutation is evaluated as

$$\tilde{q}_{\mathrm{dvw}} = \tilde{q}_{\mathrm{dvpw}} x_{\mathrm{vp}} \tag{4.2}$$

If waypoint w is visited before the vehicle starts the leg to destination d, then the anti-aircraft defense at w is assumed not to threaten the vehicle. Thus the actual probability q_{dvw} that vehicle v is *not* destroyed by an anti-aircraft defense at w is 1. Otherwise, it is $\tilde{q}_{\mathrm{dvw}}$.

$$q_{\mathrm{dvw}} \leq \tilde{q}_{\mathrm{dvw}} + M(1 - A_{\mathrm{dvw}}) \quad \text{and} \quad q_{\mathrm{dvw}} \leq 1 \tag{4.3}$$

The actual probability q_{dv} of reaching each destination can be found by evaluating Eq. 4.8 in terms of the actual probability of surviving each anti-aircraft defense q_{dvw}.

$$q_{\mathrm{dv}} = q_{(\mathrm{d}-1)\mathrm{v}} \prod_{\mathrm{w}=1}^{N_W} q_{\mathrm{dvw}} \tag{4.4}$$

Again, $d = 0$ corresponds to the vehicle's starting position and $q_{0\mathrm{v}} = \tilde{q}_{0\mathrm{v}} = 1.0$. Because Eq. 4.4 is nonlinear in decision variables q_{dvw} and q_{dv}, it cannot be included directly in the formulation, but can be transformed using logarithms as

$$\log q_{\mathrm{dv}} = \log q_{(\mathrm{d}-1)\mathrm{v}} + \sum_{\mathrm{w}=1}^{N_W} \log q_{\mathrm{dvw}} \tag{4.5}$$

While this equation form accumulates the effects of each of the anti-aircraft defense sites on the survival probability over each leg of the mission, it only provides $\log q_{\mathrm{dv}}$. Evaluating the expected value requires q_{dv}, which can be recovered approximately as q'_{dv} by raising 10 to the exponent $\log q_{\mathrm{dv}}$ using a piecewise linear function that can be included into a MILP accurately using 3 binary variables. The exact function is nearly linear in the range of interest where probabilities are above 0.3 [42, 46].

The expectation of the mission value is then found by summing waypoint values multiplied by the probability of reaching that waypoint. If the value of the d^{th} way-

point visited by vehicle v in its p^{th} permutation is $\tilde{s}_{\text{dvp}}$, then the expectation of the value s_{dv} that will be received from visiting the waypoint is

$$\forall p \in \{1, 2, \ldots, N_P\} \ : \ s_{\text{dv}} \leq q'_{\text{dv}} \tilde{s}_{\text{dvp}} + M(1 - x_{\text{vp}}) \tag{4.6}$$

and the objective of the stochastic formulation is

$$\max_{x_{\text{vp}}, t_{0\text{v}}} J = \sum_{\text{d}=1}^{n_{\max}} \sum_{\text{v}=1}^{N_V} s_{\text{dv}} - \alpha_1 \bar{t} - \frac{\alpha_2}{N_V} \sum_{\text{v}=1}^{N_V} (t_{0\text{v}} + T_{\text{n}_{\max}\text{v}}) \tag{4.7}$$

4.2.2 Advantages of Cooperative Assignment

To show the advantages of cooperation achieved by using the stochastic formulation, two alternative formulations ("purely deterministic" and "deterministic equivalence") [42] are presented and the results of an example using the three formulations are compared.

Purely Deterministic Formulation

This formulation ignores the risk in the environment and uses the maximum value formulation discussed in Chapter 2 to generate the optimal plan. Figure 4.1 shows the results of the purely deterministic formulation for an example of 3 UAVs and 5 waypoints. The central waypoint has a score of 100 points, and the other waypoints have a score of 10. The UAVs each receive a score of 50 for returning to their starting point, representing the perceived value of the UAVs relative to the waypoints. In this work, the probability that a UAV is destroyed is calculated as proportional to the length of its path within the anti-aircraft defenses range. In the nominal threat level case, the constant of proportionality was chosen so that a path to the center of the smaller anti-aircraft defense would have a probability of survival of 0.96. The formulations were also applied in environments in which the nominal constant of proportionality was multiplied by factors of 3 and 7, respectively. These particular selections are arbitrary, but the results of this comparison illustrate important trends in performance as the threat level increases. Under the nominal threat levels, this

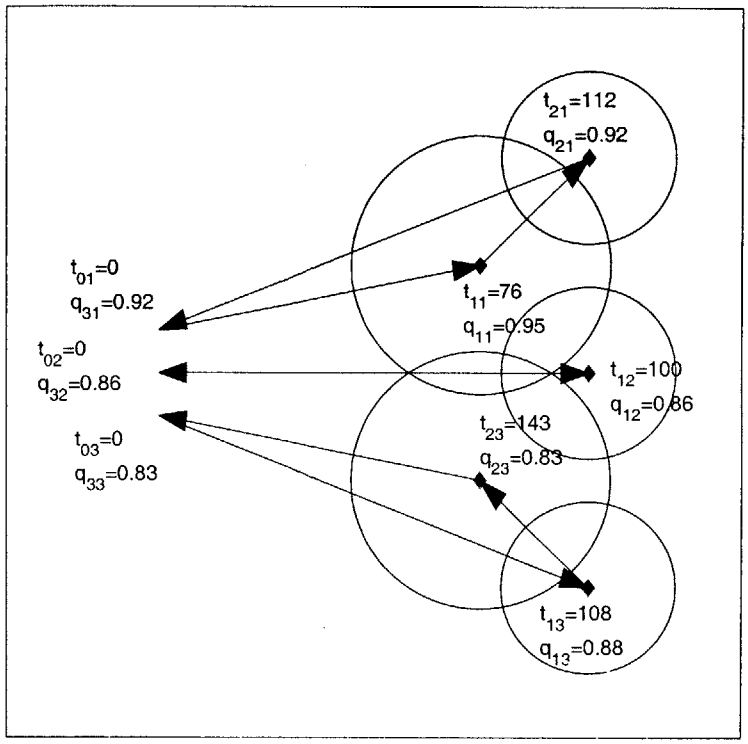


Fig. 4.1: Example of purely deterministic allocation. The vehicles start at left and visit waypoints at right. t_{dv} gives the time at which point d is reached on UAV v's mission, including departure time from starting point. q_{dv} is the probability that point d is reached on UAV v's mission. The probability of being shot down is assumed to be proportional to the length of the path in the anti-aircraft defense's range, shown with circles. Note that the middle vehicle aircraft does not delay its departure, and that the bottom vehicle passes through the large anti-aircraft defense second from the bottom once without destroying it.

formulation gave a probability of 0.86 that the high value target at the center would be reached by the UAV to which it was allocated. When the probability of destruction on each leg was increased by a factor of 3, the probability of reaching the high value target was 0.57, and when the probability of destruction was increased by a factor of 7, the probability of reaching the high value target was 0.25. Thus, in well-defended environments, the deterministic formulation plans missions that are highly susceptible to failure.

Deterministic Equivalence Formulation

In the deterministic equivalence formulation the threat that each waypoint poses to the UAVs is a fixed quantity, so that destroying it does not decrease the risk to other vehicles. This formulation reduces the problem to multiplying the value associated with each waypoint along a UAV's mission by the probability that the UAV reaches that waypoint. This calculation can be done for every permutation before the optimization is performed, so no probabilities are explicitly represented in the optimization program itself.

Let $\tilde{q}_{\mathrm{dvp}}$ be the probability that vehicle v reaches the d^{th} destination on its p^{th} permutation, and let $d = 0$ correspond to the vehicle's starting position. Then $\tilde{q}_{0\mathrm{v}} = 1.0$ for all permutations, and

$$\tilde{q}_{\mathrm{dvp}} = \tilde{q}_{(\mathrm{d}-1)\mathrm{vp}} \prod_{\mathrm{w}=1}^{N_W} \tilde{q}_{\mathrm{dvwp}} \tag{4.8}$$

$\tilde{q}_{\mathrm{dvwp}}$ is the probability that an anti-aircraft defense at waypoint w does not shoot down UAV v between its $(d-1)^{\mathrm{th}}$ and d^{th} destinations. Then, the cost function of the deterministic equivalent formulation is

$$\max_{x_{\mathrm{vp}}, t_{0\mathrm{v}}} J = -\alpha_1 \bar{t} - \frac{\alpha_2}{N_V} \sum_{\mathrm{v}=1}^{N_V} (t_{0\mathrm{v}} + T_{\mathrm{n_{max}v}}) + \sum_{\mathrm{d}=1}^{n_{\max}} \sum_{\mathrm{v}=1}^{N_V} \sum_{\mathrm{p}=1}^{N_P} \tilde{q}_{\mathrm{dvp}} \tilde{s}_{\mathrm{dvp}} x_{\mathrm{vp}} \tag{4.9}$$

where $\tilde{q}_{\mathrm{dvp}}\tilde{s}_{\mathrm{dvp}}$ is evaluated in the cost estimation step, and is passed into the optimization as a parameter.

The plans from the deterministic equivalent formulation are shown in Figure 4.2. This formulation includes a notion of risk, but does not recognize the ability of UAVs to cooperate to decrease the probability of attrition. As the threat level of the environment increases, this formulation tends to result in "pessimistic" plans, in which some of the waypoints are not visited. In the case that the risk is 3 times the nominal value, only 3 of the waypoints are visited and one UAV is kept in the base since the paths are too risky (Figure 4.2(b)). As the risk increases to 7 times its nominal value, the optimal plan is to keep all the UAVs in the base and do nothing (Figure 4.2(c)).

This situation occurs when the contribution to the expected value of visiting the remaining waypoints is offset by the decrease in the expected value due to a lower probability of surviving to return.

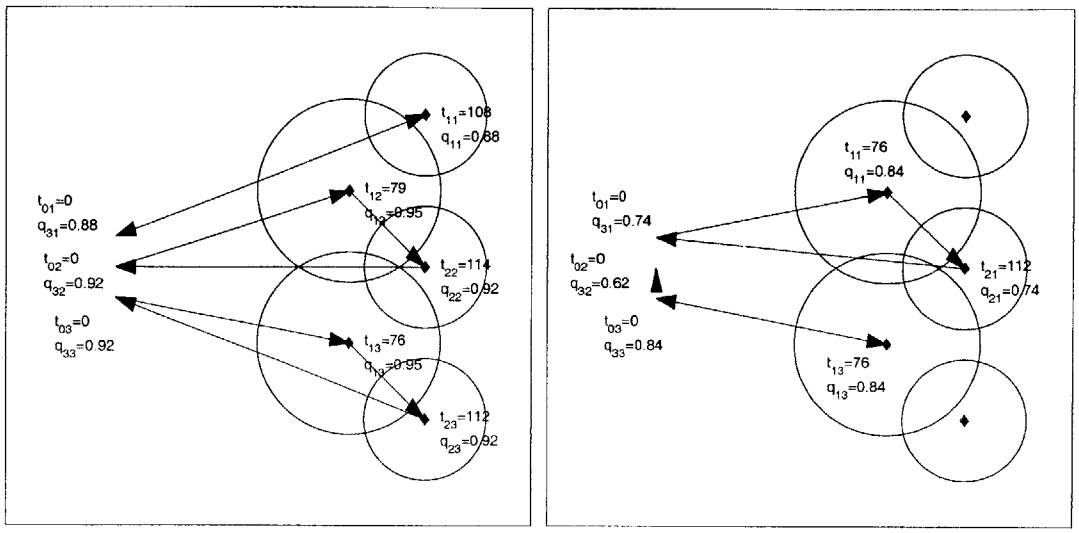
Figure 4.3 shows the optimal cooperative assignments for this problem using a stochastic formulation. A careful analysis shows that this formulation recovers phasing (*e.g.*, $t_{02} = 76$) and preserves the UAV that visits the high value target. As the threat level in the environment increases, the upper and lower waypoints are ignored (Figure 4.3(c)). The ability to reduce risk through cooperation is captured by evaluating the actual risk during optimization as a function of the waypoint visitation precedence.

Analysis of the Results

After the assignment problem was solved for nominal threat values using the three formulations described above, the resulting allocation solutions were evaluated using the model of the stochastic formulation of Section 4.2.1. Table 4.1 shows a comparison of the resultant expected value, mission completion time, and probability of survival of the three formulations. The computation time of each formulation is also shown. The expected values of the purely deterministic and stochastic formulations are very different, although the waypoint combinations assigned to each UAV are the same. The allocation differs mainly in timing, emphasizing the importance of timing of the activities.

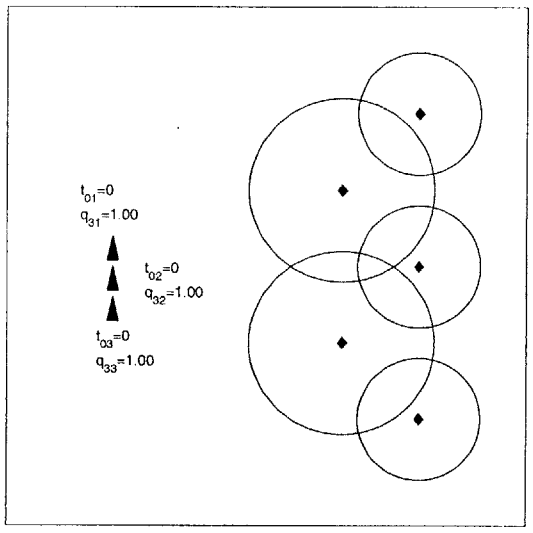
While some improvement over the purely deterministic formulation is seen in the deterministic equivalent formulation, the stochastic formulation achieves the highest expected value. Although this formulation also does the best job of protecting the survival of the UAV that visits the high value target, it is the most computationally demanding formulation.

The results of applying all three formulations in high threat environments are shown in Tables 4.2 and 4.3, which indicate that (in high threat environments) the purely deterministic and deterministic equivalent approaches are incapable of recovering a higher expected value than would be achieved by keeping the UAVs at their



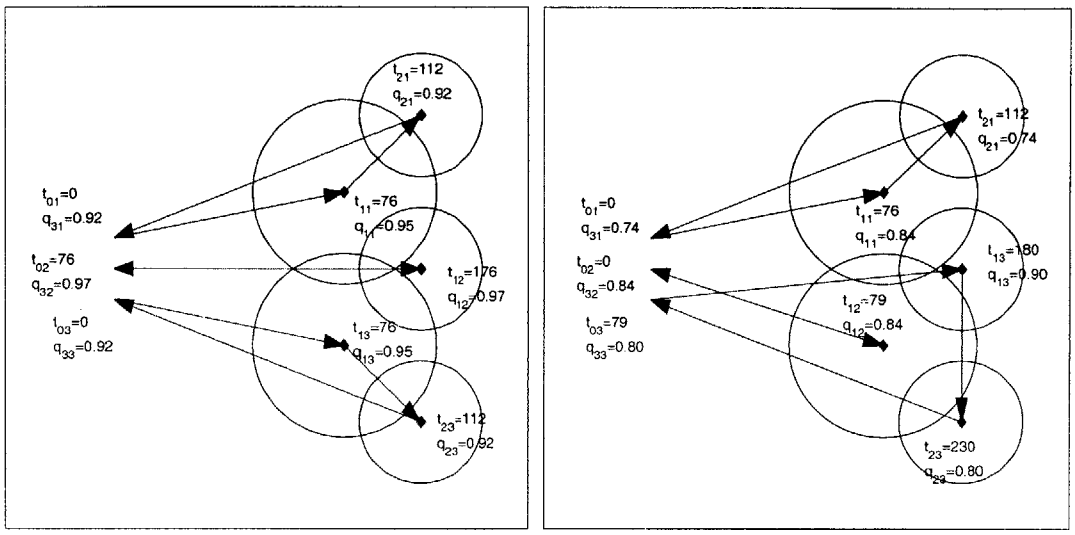
(a) Risk : Nominal

(b) Risk : 3× Nominal



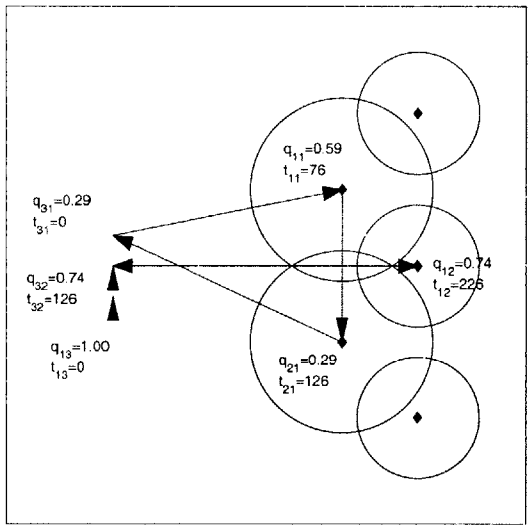
(c) Risk : 7× Nominal

Fig. 4.2: Example of deterministic equivalent allocations. Nominal probabilities of destruction at top left, increased by a factor of 3 at top right and increased by factor of 7 at bottom. At top left, UAV 1 could exploit phasing by waiting for UAV 2 to destroy the anti-aircraft defense second from the top threatening 1 on its way to its target. However, the probabilities are fixed quantities, so the benefits of cooperation between UAVs are not recognized, and UAVs 1 and 2 leave simultaneously at $t = 0$. As the threat level of the environment increases, the allocation that maximizes the expectation of value keeps the UAVs at their base in order to collect the reward for safe return, and the high value waypoint is not visited.



(a) Risk : Nominal

(b) Risk : 3× Nominal



(c) Risk : 7× Nominal

Fig. 4.3: Example of maximum expected value allocation. Nominal probabilities of destruction at top left, increased by a factor of 3 at top right and increased by a factor of 7 at bottom. Note that in all 3 cases, phasing is employed: the two larger anti-aircraft defense sites have been visited before the UAV that visits the high value begins its mission, and this UAV retains the highest probability of survival. As the threat level of the environment increases, only the high value target and the anti-aircraft defenses that threaten the route to it are visited.

Table 4.1: Results of the three formulations in risky environments (nominal threat levels)

Formulation	Exp. value	Mission Time $\bar{t}$	Computation Time (sec)
Purely Deterministic	251.3	219.5	6.5
Deterministic Equivalent	263.8	219.5	7.0
Stochastic	273.1	276.5	27.1

Table 4.2: Expected values in threatening environments. Various probabilities of destruction (nominal, and 3 and 7 times higher).

Formulation	Expected value		
	Nominal	$\times 3$	$\times 7$
Purely Deterministic	251.3	173.1	81.4
Deterministic Equivalent	263.7	219.6	150.0
Stochastic	273.15	239.9	208.7

Table 4.3: Probability of reaching high value target. Various probabilities of destruction (nominal, and 3 and 7 times higher).

Formulation	Probability		
	Nominal	$\times 3$	$\times 7$
Purely Deterministic	0.86	0.57	0.25
Deterministic Equivalent	0.92	0.74	0.00
Stochastic	0.97	0.9	0.74

base. Also, these two formulations are not capable of designing a plan that is likely to reach the high value target (the probability of reaching the high value target for the purely deterministic approach is 0.25 and is 0 for the deterministic equivalent).

Chapter 5 presents the results of a purely deterministic formulation on Boeing's Open Experiment Platform (OEP). The results of the experiments validate the above arguments. By ignoring the risk in the assignment, UAVs are assigned to targets aggressively and therefore they get shot down as they implement the plan. These results express the need for taking into account the risk of the environment and planing cooperatively to reduce this risk.

4.3 Cooperative Weapon Target Assignment

The UAV task assignment problem is closely related to the Weapon Target Assignment (WTA) problem, which is a well-known problem that has been addressed in the literature for several decades [45, 47, 48, 49, 50]. The problem consists of N_w weapons and N_t targets, and the goal is to assign the weapons to the targets in order to optimize the objective, which is typically the expected accumulated value of the mission. Each target i has a value (score) of s_i and, if it is targeted by weapon j, then there is a probability p_{ij} that the target will be destroyed. Therefore the expected value of assigning weapon j to target i will be $p_{ij}s_i$. Several extensions of the general problem have been addressed and solved using different methodologies. This section looks at the WTA problem from a different perspective, but the main idea is similar to the case of *cooperation* discussed for the UAV task assignment.

The problem addressed is that of weapon target assignment in a risky environment. Two formulations will be presented. The first is simple to solve, but the objective function ignores the effect that the tasks performed by some of the weapons can have on the risk/performance of the other weapons. The resulting targeting process is shown to be *coordinated*, but because it ignores this interaction, it is *non-cooperative*. The second formulation accounts for this interaction and solves for the *optimal cooperative strategy* using Dynamic Programming (DP). Two approximation methods are also discussed later as an alternative approach to solve these problems and achieve an answer that is close to optimal in a reasonable computation time.

Consider the WTA problem where the targets are located in a risky environment and a weapon can get shot down while flying over these targets. (Some of these targets represent SAM sites that can shoot down UAVs or weapons). Targets have different values that get discounted with time, meaning that if the target is hit now its value is higher than if it is hit in the future. Including this time discount is particularly important for environments with targets that pop-up and then disappear/move. Since the weapons are at risk of being shot down, there is a limited probability of success for each weapon aiming at the target; this will be a function of the *risk* associated

with the regions it must fly over.

The problem is to assign weapons to targets in different stages (time steps) in order to maximize the expected accumulated value. Note that "time" and "stage" are used interchangeably in this formulation. The expected value for target i, with value of s_i at time t, is $p_i(t)\lambda_i^t s_i$, where $(\lambda_i \leq 1)$ is the time discount factor. $p_i(t)$ represents the probability of success in destroying target i at time t and is a function of the existing SAM sites at time t. The problem then can be formulated as

$$\max_{x_{it}} \sum_{t=1}^{N} \sum_{i=1}^{N_t} p_i(t)\lambda_i^t s_i x_{it} \tag{4.10}$$

$$\text{s.t.} \quad \sum_{t=1}^{N} x_{it} \leq 1 \quad \forall i \in \{1 \ldots N_t\}$$

$$\sum_{t=1}^{N} \sum_{i=1}^{N_t} x_{it} \leq N_w \tag{4.11}$$

$$x_{it} \in \{0,1\} \quad \forall i \in \{1 \ldots N_t\},\ \forall t \in \{1 \ldots N\}$$

where decision variable, x_{it} equals 1 if target i is assigned to be hit at stage t. The total number of stages (time horizon) is N. The first constraint ensures that each target is assigned at most once, and the second constraint limits the number of assigned targets to the number of available weapons.

With the time discount, it is typically desirable to hit the targets as soon as possible (*i.e.,* in the first stage). However, since the risk in the environment will be reduced in later stages as SAM sites are removed, the probability of success, $p_i(t)$, will increase with time, which increases the expected score. Therefore there is a trade-off between time and risk that must be captured in the optimization problem.

4.3.1 Non-cooperative Formulation

The first formulation is defined as an assignment in which the effect of weapons on the performance of other weapons is ignored. In this case the probability of success $p_i(t)$ is not a function of time and the objective function in Eq. 4.12 can be rewritten

as

$$\max_{x_{it}} \sum_{t=1}^{N} \sum_{i=1}^{N_t} p_i \lambda_i^t s_i x_{it} \tag{4.12}$$

Since the survival probabilities are constant in this formulation, the time discount $\lambda_i < 1$ forces the targets to be assigned in the first stage. As a result, the optimization simplifies to a sorting problem in which the targets are sorted based on their expected value $p_i s_i$ and N_w targets that have the largest expected values get assigned. Chapter 5 presents the results of this formulation, showing that the task assignment is coordinated, but not cooperative.

4.3.2 Cooperative Formulation

This section presents a more cooperative weapon target assignment approach that can be solved as a Dynamic Programming (DP). To proceed, define the *state* of the system at each time step (stage) t to be the list of remaining targets, r_t, and the number of remaining weapons, m_t. Several assumptions have been made to simplify the notation: the weapons are assumed to be similar; the time discount factor λ_i is assumed to be equal for all targets; and the risk associated with the SAM sites are assumed to be equal. However, the same algorithm can be used and the same discussion holds for the general case.

At any stage t, the decision (control), u_t is defined to be the list of targets to be hit at that stage. Bellman's equations for this problem can be written as

$$\begin{aligned} J_t^*(r_t, m_t) &= \min_{u_t, |u_t| \leq m_t} \left\{ S(u_t) + \lambda J_{t+1}^*(r_t - u_t, m_t - |u_t|) \right\}, \ t \in \{0, ..., N-1\} \\ J_N^*(r_N, m_N) &= 0 \qquad \text{where} \qquad S(u_t) = \sum_{s_i \in u_t} p_i(t) s_i \end{aligned} \tag{4.13}$$

and $|u_t|$ is the size of u_t (*i.e.,* the number of targets assigned at stage t). $p_i(t)$ represents the survival probability associated with the path that the weapon takes to the target. Note that it can be an arbitrary function of this path (*e.g.,* simply proportional to the time that weapon is inside each SAM range) or it can also be a function of the distance from the center of SAMs.

Solving the DP in Eq. 4.13 for r_0 equal to the list of all the targets, and m_0 equal to the number of available munitions, gives a sequence of optimal u_t^* that defines which targets to hit at each stage. $J_0^*(r_0, m_0)$ is the optimal expected score. Note that the horizon in the above DP problem, N, is finite and is less than the number of targets ($N \leq N_t$). It is trivial to show that in any optimal assignment all the targets are targeted before stage N. In this work, $N = N_t$. Because $p_i(t)$ is a function of time, the benefit of removing SAM sites in reducing the risk for other weapons is captured in the formulation. The DP solution will thus provide the optimal balance between risk and time. Furthermore, since weapons will be assigned to targets specifically to reduce the risk for other UAVs, the solutions will be both coordinated and cooperative.

4.3.3 A Simple Example

The first example is similar to Figure 4.3, which was used to show the effectiveness of the cooperative assignment. The problem in Figure 4.4 has 5 targets (all SAM sites) with different values and ranges (the circles around the SAM sites show their range). The score of each target is shown next to it. The dotted lines show the trajectory (assumed to be straight lines between weapon and target) to each target from the weapon site, and the solid portion corresponds to the risky part of the path that goes over the SAM. The position of the weapons is shown by $\triangle$ and the total number of available weapons at the beginning of the mission is shown next to the weapons ($N_w = 3$ in this example).

To calculate the survival probability of flying over SAM site j for d_j units of distance, the following equation is used

$$\tilde{p_j} = p^{d_j} \tag{4.14}$$

where $0 \leq p \leq 1$ and $1 - p$ is the probability of getting shot down for flying over the SAM site for 1 unit of distance. The overall survival probability for a weapon flying

over a set J of SAM sites to target i can be calculated as

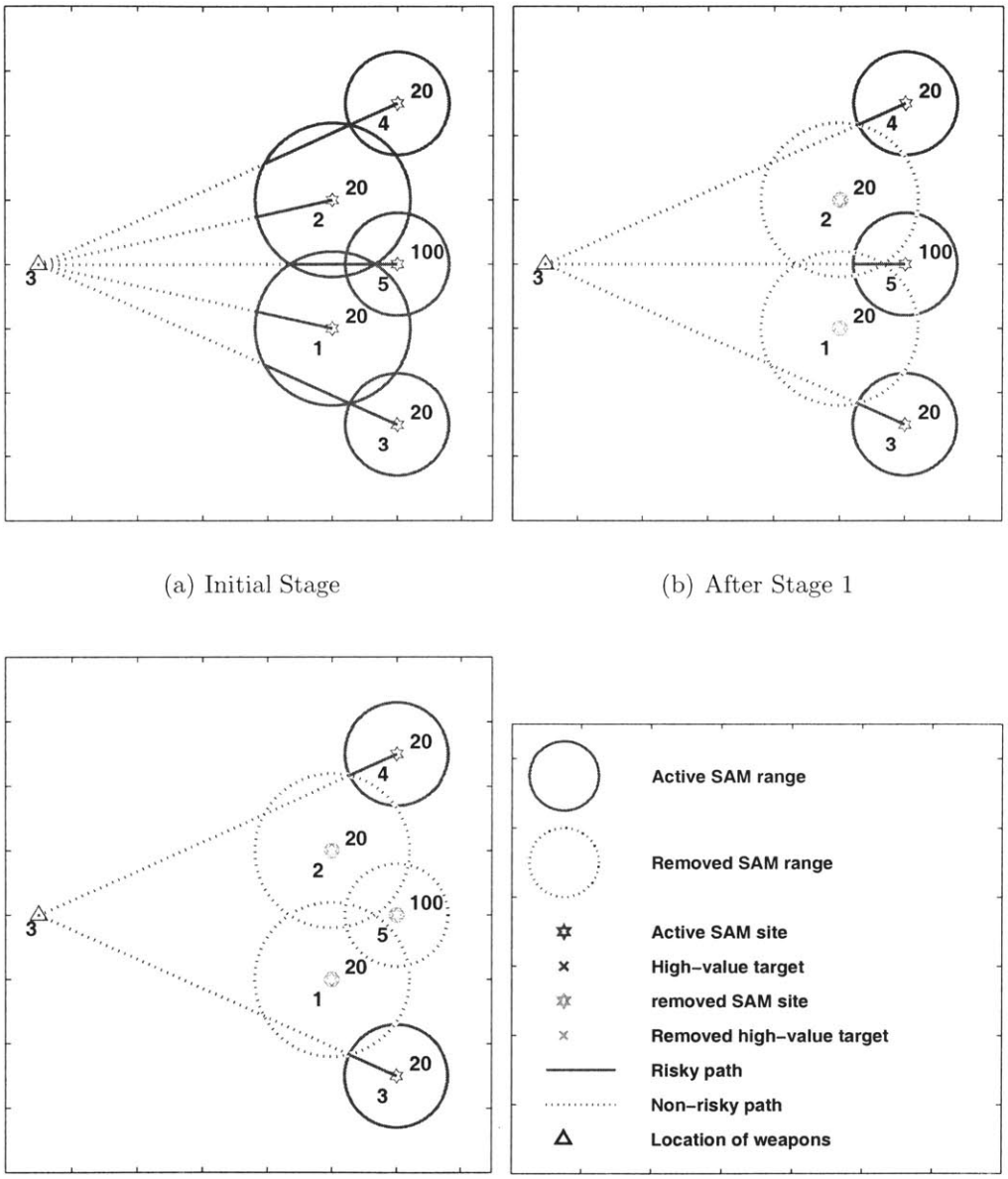
$$p_i = \prod_{j \in J} \tilde{p}_j \tag{4.15}$$

The survival probability, p is set to 0.95 and the time discount coefficient, λ, is set to 0.9 for this example. Figure 4.4 shows the optimal DP solution to this problem. Figure 4.4(a) is the initial state of the environment. In stage 1 (Figure 4.4(b)), SAM sites 1 and 2 are removed, reducing the risk along the path to SAM site 5, which has a much higher value (*e.g.,* a command post). The dotted circles show that the SAM site has been removed and risk is no longer associated with flying over that region. In stage 2 (Figure 4.4(c)), the last weapon is assigned to the high value target through a low-risk path.

To see the advantages of cooperation in this formulation, the expected value of this assignment is compared to the first formulation in Eq. 4.12. This approach assigns the three highest value targets in a single stage. The expected value for the two assignments for different values of time discount factor, λ, and survival probability, p, are shown in Table 4.4. For a fixed value of λ, as the survival probability decreases, the difference between the expected value of cooperative and non-cooperative assignments increases. This shows that cooperation is crucial in high risk environments. For a fixed p, as the value of λ decreases the difference between the two assignments decreases, showing that when time is very important in the mission, planning in stages is not as attractive. Figure 4.5 shows the same results for a continuous range of p and different values of λ.

4.3.4 Larger Simulations

In this section a larger problem ($N_t = N_w = 10$) is used to better show the cooperation achieved by this formulation. The value of survival probability, p is set to 0.9 and the time discount factor, λ is set to 0.9 as well. Figure 4.6 shows the result of the optimal cooperative assignment using the DP algorithm. Figure 4.6(a) illustrates the initial stage of the environment. In this example, targets 4 and 7 are high value targets and



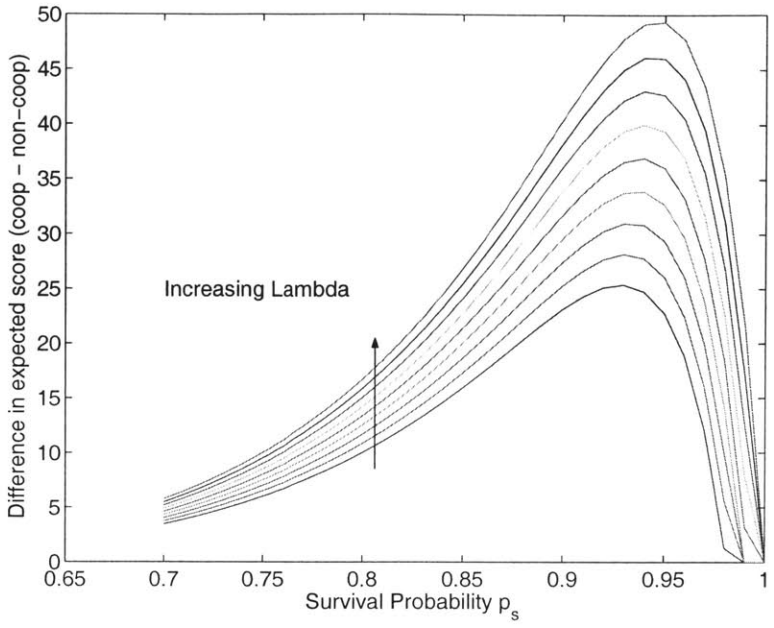
(a) Initial Stage

(b) After Stage 1

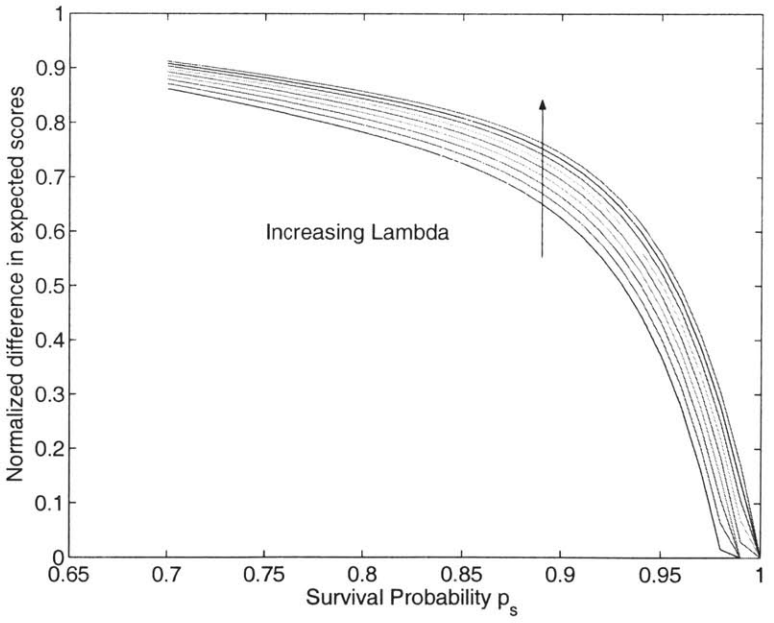
(c) After Stage 2

(d) Legend

Fig. 4.4: The solution to the cooperative weapon target assignment for a problem of 3 weapons and 5 targets in a risky environment.



(a) Absolute difference



(b) Normalized difference

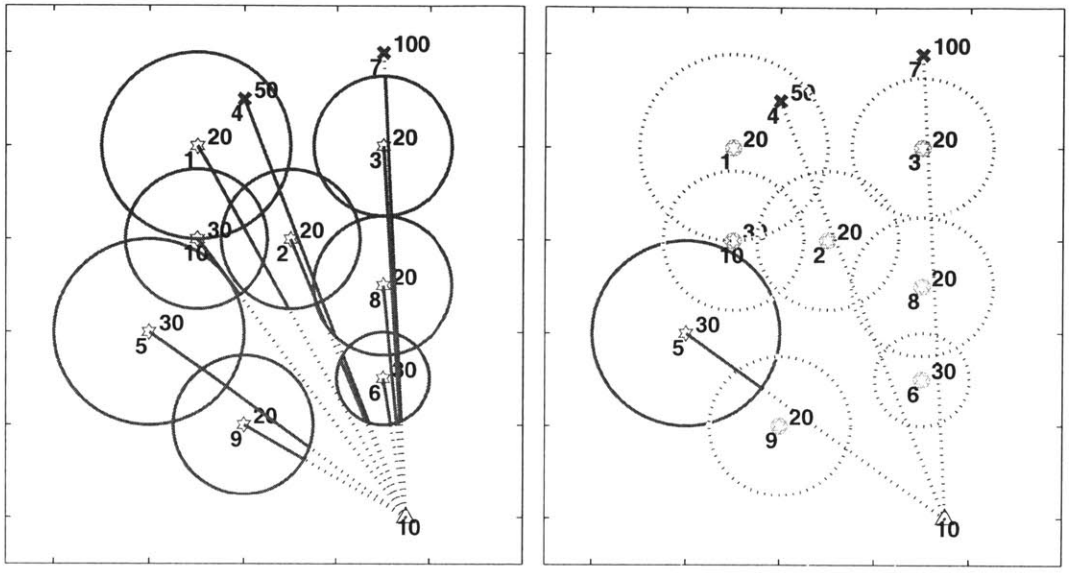
Fig. 4.5: The effect of survival probability p_s and time discount factor λ on the advantages of a cooperative assignment over a non-cooperative assignment in the WTA problem.

Table 4.4: Comparison of the cooperative and non-cooperative assignment for different values of λ and p_s.

	Cooperative solution				Non-cooperative
p_s	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 0.9$	$\lambda = 1.0$	solution
0.80	12.8	16.2	17.8	19.5	2.8
0.90	37.1	45.7	50.0	54.3	13.9
0.95	61.4	74.7	81.3	88.0	38.6
0.98	82.4	99.4	108.0	116.5	81.2

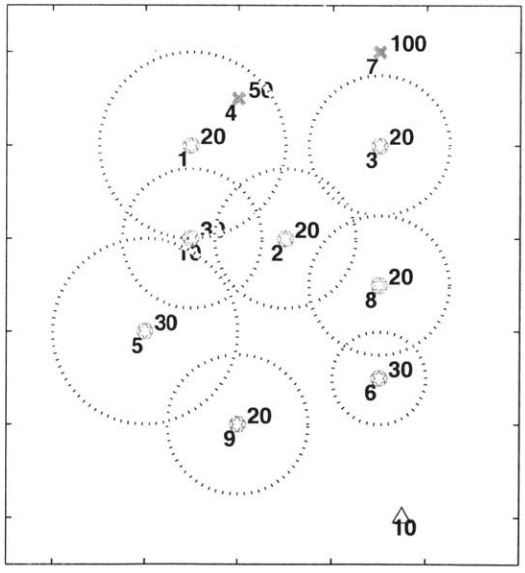
the rest of the targets are SAM sites with different ranges and values. Figure 4.6(b) shows the environment after stage 1. At stage 1, all the SAM sites that make the path to high value targets risky are removed. Note that SAM sites 9 and 10, which are not threatening any paths, are also removed. This is due to the fact that postponing the assignment of these targets will just reduce their expected value.

In stage 2 (Figure 4.6(a)) the remaining weapons are assigned to the remaining targets. To show the effect of the discount factor in the results, the same problem is solved for $\lambda = 0.97$. The optimal answer in this case assigns weapons to targets in 4 stages (Figure 4.7). Since the time discount is very close to 1, the effect of time on the values of targets is very small and therefore the algorithm assigns the weapons to targets in order to maximize their expected value $p_i(t)s_i$. This situation forces the weapons to be assigned to targets sequentially. In the first stage (Figure 4.7(a)), SAM sites $6, 8, 9$, and 10 that are on the way to the rest of the SAM sites are removed. In stage 2 (Figure 4.7(b)), SAM sites $2, 3$, and 5, whose paths were cleared in the previous stage, are assigned to be hit. Figure 4.7(c) shows the 3^{rd} stage where high value target 7, which now has a no-risk path, is removed. SAM site 1 is also removed in this stage to clear the path to high-value target 4. These two examples clearly show cooperation in the assignment, in which the objective of the assignment is not only to achieve value for each weapon, but also to increase the probability of success for other weapons. This cooperative approach which results in an assignment with a much higher overall expected value.



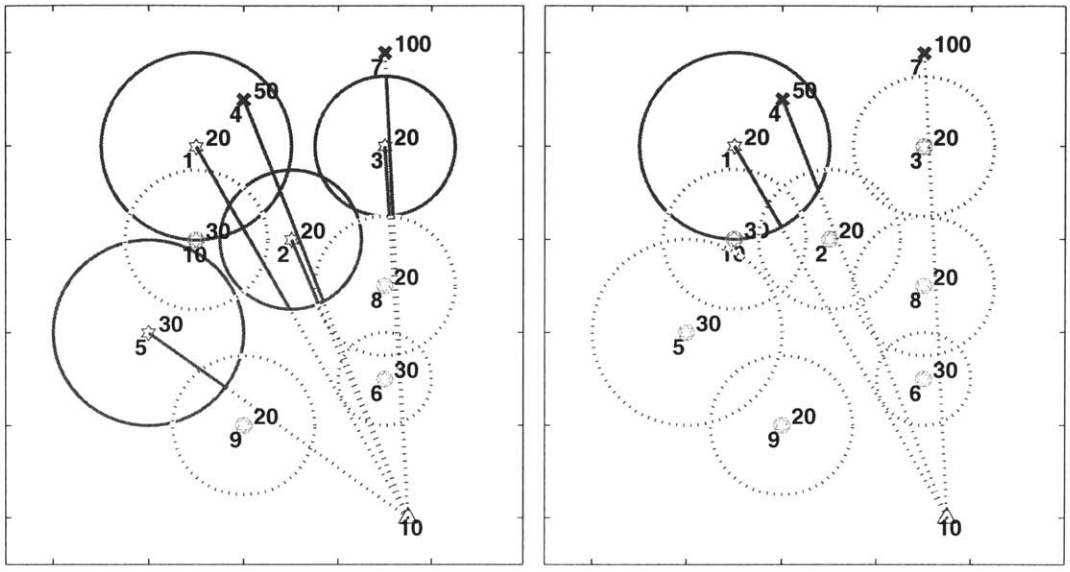
(a) Stage 0

(b) After stage 1



(c) After stage 2

Fig. 4.6: Optimal solution for the problem of 10 weapons and 10 targets, $p_s = 0.9$ and $\lambda = 0.9$. The mission is implemented in 2 stages and the expected value = 160.



(a) After stage 1

(b) After stage 2

(c) After stage 3

(d) After stage 4

Fig. 4.7: Optimal solution for the problem similar to Figure 4.6 with $p_s = 0.9$ and $\lambda = 0.97$. The mission is implemented in 4 stages and the expected value = 174.

4.4 Approximate Dynamic Programming

The DP algorithm generates an optimal cooperative weapon target assignment in a risky environment, but as the dimension of the problem (number of targets N_t) grows, the computation time grows exponentially for this approach. In this section two approximation DP methods are proposed to solve computation issues for large problems.

4.4.1 One-step Lookahead

In order to reduce the computation required by DP, an effective way is to reduce the horizon at each stage based on the lookahead of a small number of stages [51]. This idea is very similar to the receding horizon task assignment in which the planning horizon is limited to reduce the computation. The simplest possibility is to use a one-step lookahead where at stage t and state r_t the control u_t minimizes the expression

$$\min_{u_t, |u_t| \leq m_t} \{S(u_t) + \lambda \bar{J}_{t+1}(r_t - u_t, m_t - |u_t|)\}, t \in \{0, ..., N-1\} \tag{4.16}$$

$\bar{J}_{t+1}$ is an approximation of the true cost-to-go function, J^*_{t+1}, with $\bar{J}_N = 0$. In the one-step lookahead, having the approximation $\bar{J}$, the calculation reduces to one minimization per stage, which is a significant savings compared to an exact DP. The performance of the one-step lookahead policy depends on how well $\bar{J}$ approximates the true cost-to-go. A good cost-to-go can be calculated using complex algorithms and results in a close to optimal answer, but the computation complexity associated with calculating the cost-to-go itself might defeat the purpose. Therefore, while a good approximate cost-to-go is desirable, the calculation must be simple. A simple approximation of cost-to-go for the problem of the weapon target assignment is introduced here that can be calculated very fast. At stage t and state (r_t, m_t), $\bar{J}_t(r_t, m_t)$ is the solution to the non-cooperative formulation in Eq. 4.12.

$$\bar{J}_t(r_t, m_t) = \max_{x_{it}} \sum_{t=1}^{N} \sum_{i=1}^{N_t} p_i \lambda_i^t s_i x_{it} \tag{4.17}$$

$$\text{s.t.} \quad \sum_{t=1}^{N} x_{it} \leq 1, \qquad \forall i \in r_t \tag{4.18}$$

$$\sum_{i=1}^{N_t} \sum_{t=1}^{N} x_{it} \leq N_w \tag{4.19}$$

This cost-to-go approximation assumes that all the remaining weapons are assigned to targets in the next stage. This is a simple approximation cost-to-go that can be calculated very easily and as a result, the computation time required to generate the assignment is much lower than the exact DP algorithm. To compare the result of the one-step lookahead approximation with the optimal solution from the exact DP algorithm, the problem of 10 weapons and 10 targets discussed in Section 4.3.4 is used. The results of the approximation method for $\lambda = 0.9$ are shown in Figure 4.8 and are compared to the optimal result in Table 4.5. In the optimal solution, the mission is accomplished in 2 stages while in the one-step lookahead solution it is accomplished in 5 stages. This assignment has resulted in lower performance compared to the optimal solution, but the computation time is considerably reduced. In the next section, the two-step lookahead algorithm will be discussed to increase the performance compared to one-step lookahead.

4.4.2 Two-step Lookahead

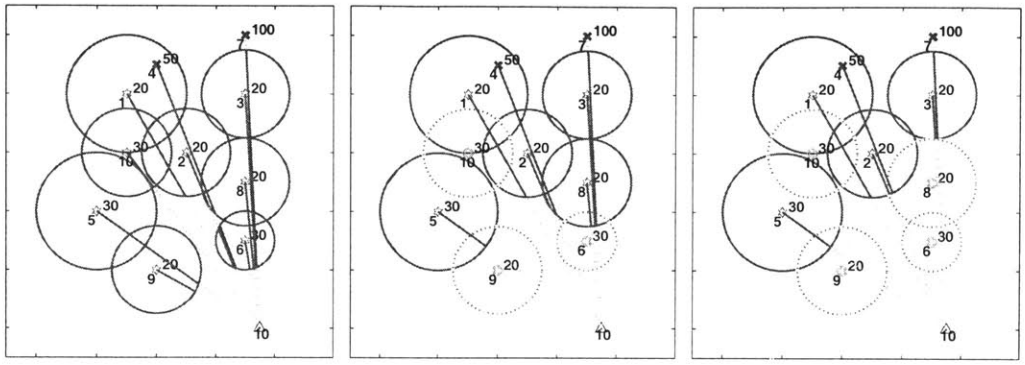
The two-step lookahead policy applies at stage t and state (r_t, m_t), and the control u_t minimizes

$$\min_{u_t, |u_t| \leq m_t} \{S(u_t) + \lambda \tilde{J}_{t+1}(r_t - u_t, m_t - |u_t|)\}, t \in \{0, ..., N-1\} \tag{4.20}$$

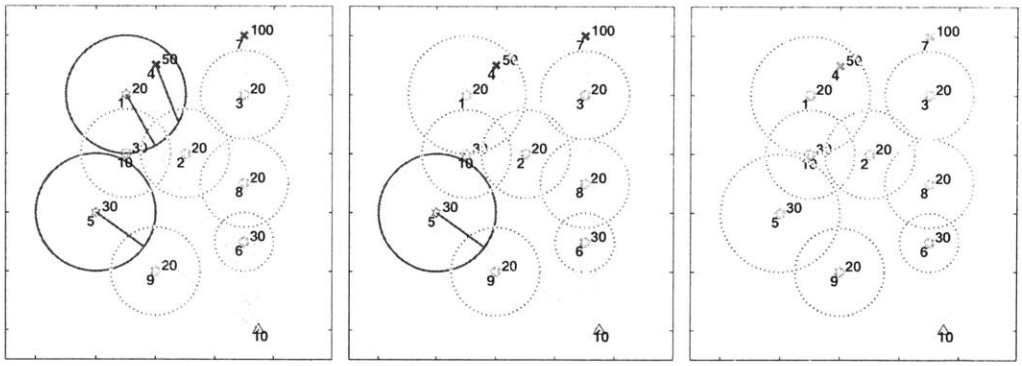
where $\tilde{J}_{t+1}$ is obtained on the basis of a one-step lookahead approximation

$$\tilde{J}_{t+1}(r_{t+1}, m_{t+1}) = \min_{u_{t+1}, |u_{t+1}| \leq m_{t+1}} \{S(u_{t+1}) + \lambda \bar{J}_{t+2}(r_{t+1} - u_{t+1}, m_{t+1} - |u_{t+1}|)\} \tag{4.21}$$

and $\bar{J}_{t+2}$ is an approximation of the true cost-to-go function J^*_{t+2}. The approximation discussed in Eq. 4.19 for the one-step lookahead is also used for the two-step



(a) Stage 0 (b) After stage 1 (c) After stage 2



(d) After stage 3 (e) After stage 4 (f) After stage 5

Fig. 4.8: One-step lookahead solution to a problem similar to Figure 4.6, with $p_s = 0.9$ and $\lambda = 0.9$. The mission is implemented in 5 stages and the expected value = 133.

Table 4.5: Comparing the result of the non-cooperative, DP, one-step lookahead and two-step lookahead solutions for the problem of 10 weapons and 10 targets.

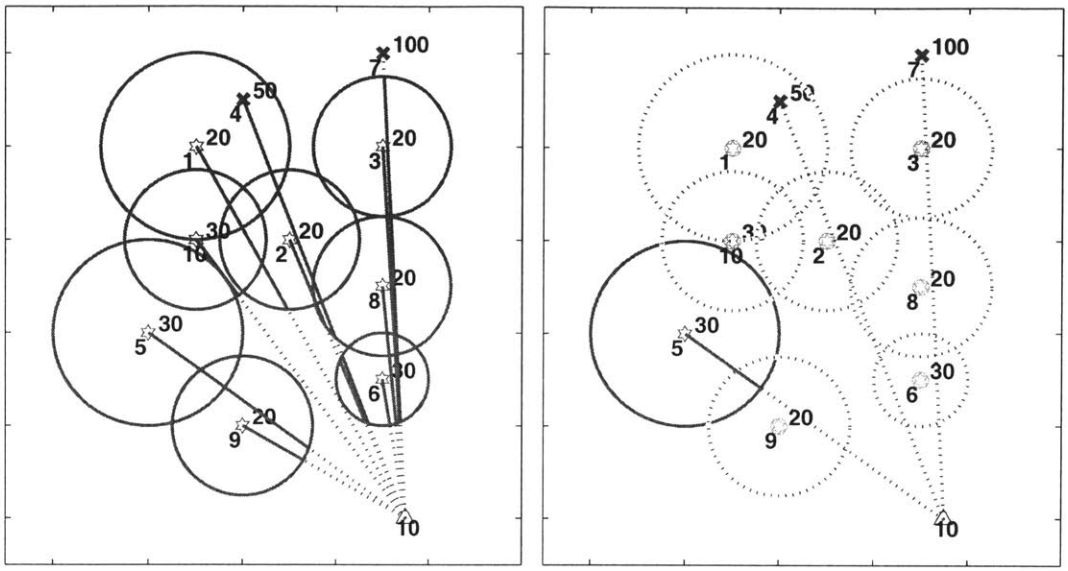
Algorithm	Expected Accumulated Value	Computation time	Number of stages
one-step lookahead	133.7	0.4 sec	6
two-step lookahead	160.0	13.4 sec	2
DP	160.0	56.2 sec	2
non-cooperative	21.9	0.1 sec	1

lookahead policy. This method is compared with the one-step lookahead and exact DP solutions using the problem of 10 targets and 10 weapons are solved. The result is shown in Figure 4.9 and is identical to the optimal solution. Table 4.5 compares the results of this method with previous methods. Computation time is substantially reduced compared to the exact DP case, but as expected, is higher than the one-step lookahead. On the other hand, the performance increases form the solution of the one-step lookahead, and in this case is identical to the optimal solution. To see if these results hold for other cases, the three algorithms (exact DP, one-step and two-step lookahead) were used to solve many randomly generated scenarios. In any set of these scenarios, the number of targets, N_t, number of weapons, N_w, time discount factor, λ, and survival probability, p, are kept constant and the position and value of targets and the range of SAM sites are randomly generated.

Figure 4.10 illustrates the results of these simulations. The horizontal axis in this graph shows the degree of sub-optimality percentage defined as

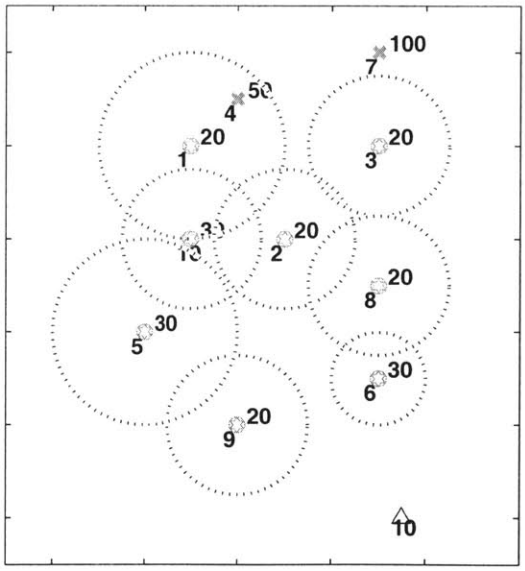
$$100 \times \frac{J_{\text{optimal}} - J_{\text{approximation}}}{J_{\text{optimal}}} \tag{4.22}$$

The vertical axis shows the cumulative percentage of the cases that are within the interval of sub-optimality indicated on the horizontal axis. These results clearly demonstrate that two-step lookahead policy outperforms the one-step lookahead policy, and that the performance of the two-step lookahead is very close to the optimal performance.



(a) Stage 0

(b) After stage 1



(c) After stage 2

Fig. 4.9: Two-step lookahead solution to a problem similar to Figure 4.6, with $p_s = 0.9$ and $\lambda = 0.9$. The mission is implemented in 2 stages and the expected value $= 160$.

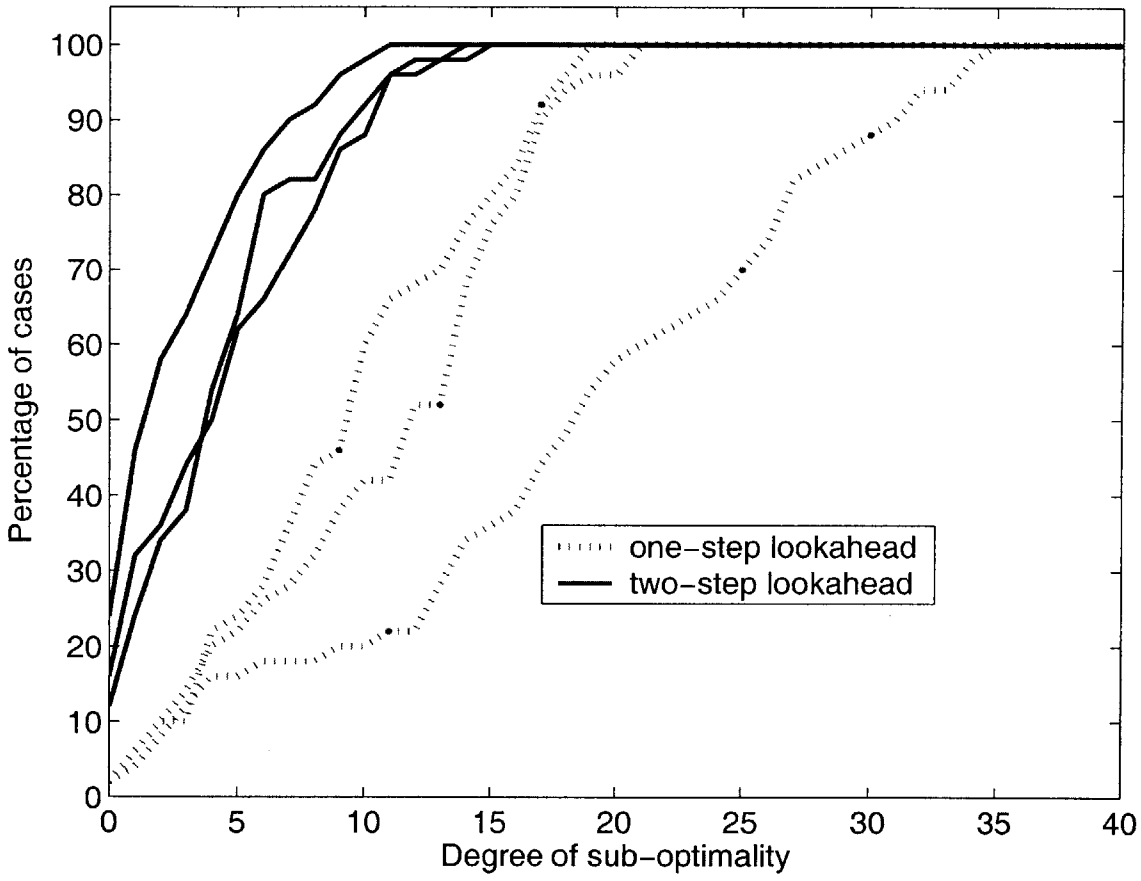


Figure 4.10: Illustrating the performance of the one-step lookahead and two-step lookahead policy against the optimal solution for different values of λ. Degree of sub-optimality is defined in 4.22.

4.5 Conclusions

This chapter discussed the problem of risk in the environment and a new stochastic formulation of UAV task assignment problem was presented. This formulation explicitly accounts for the interaction between the UAVs – displaying *cooperation* between the vehicles rather than just *coordination*. Cooperation entails coordinated task assignment with the additional knowledge of the future implications of a UAV's actions on improving the expected performance of the other UAVs. The key point is that the actions of one UAV can reduce the risk in the environment for the other UAVs; and the new formulation takes advantage of this fact to generate cooperative assignments to achieve better performance. We further extended the notion of cooperation to the Weapon Target Assignment (WTA) problem. The problem was formulated as a Dynamic Programming (DP) problem. A comparison with other approaches showed that including this cooperation lead to a significant increase in performance. Two DP approximation methods (the one-step and two-step lookahead) were also developed for large problems where curse of dimensionality in DP is prohibitive. Simulation results showed that the one-step lookahead can generate a cooperative solution very quickly, but the performance degrades considerably. The two-step lookahead policy generated plans which are very close to (and in most cases, identical to) the optimal solution.

Chapter 5

Experimental Results

This chapter presents several experimental results for Receding Horizon Task Assignment (RHTA) on multi-vehicle testbeds. These testbeds offer invaluable real world experiences such as the effect of unknown parameters, model uncertainty, environment uncertainty and noise. The first set of experiments uses two testbeds that have recently been developed at MIT to demonstrate the coordination and control of teams of multiple autonomous vehicles [52]. The first testbed is comprised of eight rovers and four blimps operated indoors to emulate a heterogeneous fleet of vehicles that can be used to perform search and rescue missions. The second testbed uses eight small aircraft that are flown autonomously using a commercially available autopilot. This combination of testbeds provides platforms for both advanced research and very realistic demonstrations. Typical results of the RHTA algorithm are presented for two representative experiments that examine different capabilities of RHTA (*i.e.*, replanning in real-time, and including timing/capability constraints). In the second set of experiments, the RHTA algorithm is implemented in Boeing's Open Experimental Platform (OEP) [29] for real-time planning in a dynamic environment.

5.1 Hardware Testbeds

The algorithm architecture shown in Figure 5.1 was designed to be very similar for the two testbeds. Each box represents an action taken in the algorithm and is briefly

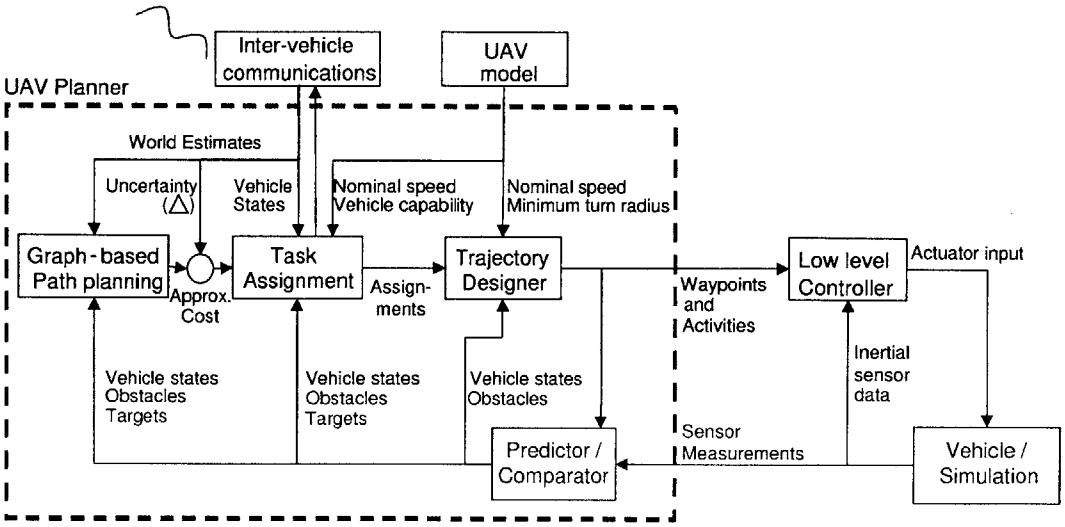


Fig. 5.1: Algorithm Architecture.

explained here (for a detailed explanation of each action see [53]).

- **Graph-based Path Planning**: The map of the environment is translated into a visible path graph and a distance matrix which then can be used in Task Assignment.
- **Task Assignment**: The distance matrix and current state of the environment are used to generate an ordered list of tasks for each UAV using the RHTA algorithm.
- **Trajectory Designer**: The trajectory designer uses the models of the UAVs and the list of tasks for each UAV to generate a detailed trajectory for each UAV. Receding Horizon Control (RHC) [46, 53] is used to generate the detailed trajectories.
- **Low Level Controller**: The low level controller enables each UAV to follow the waypoints generated by the "Trajectory Designer" in the presence of measurement noise and plant disturbances.
- **Predictor / Comparator**: In this part, the sensor output is translated to generate estimates of the environmental states. The data is then compared to propagated states and the results are reported to various parts of the architec-

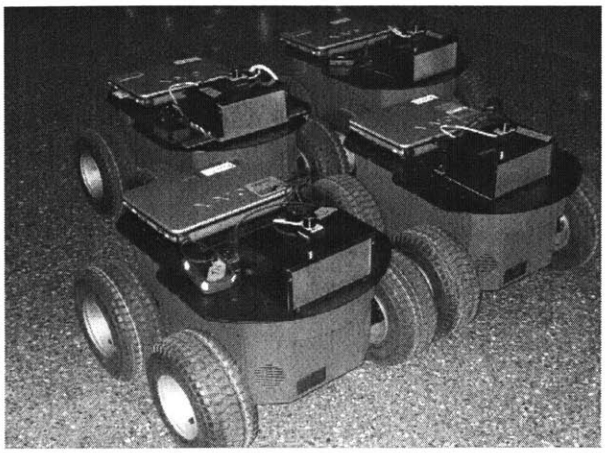


Fig. 5.2: 4 of 8 ActivMedia P3-AT Rovers.

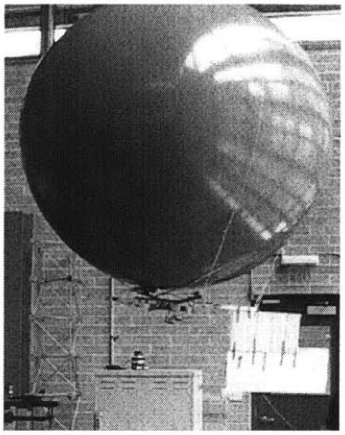


Fig. 5.3: 1 of 4 Blimps.

ture in order to initiate and aid in replanning.

- **Vehicle / Simulation**: This can be either an actual vehicle (*i.e.,* UAV, rover or blimp) or a vehicle simulator.

5.1.1 Rover/Blimp Testbed

The first testbed uses multiple rovers and blimps operated indoors to emulate a heterogeneous fleet of vehicles that can be used to perform Suppression of Enemy Air Defense (SEAD) type missions. The rovers in Figure 5.2 are ActivMedia's P3-AT's, which are operated with minimum speed and turn rate constraints to emulate the motion of an aircraft. A Sony VAIO mounted on the rover processes sensor data and performs the low-level control, while all high-level planning is done off-board using 2.4 GHz Dell laptops running MATLAB and CPLEX. A direct wireless Ethernet connection provides a fast and reliable network between the laptops, so this is equivalent to having all computation performed onboard. The ArcSecond Constellation 3D-i [54] indoor positioning system is used to measure the vehicle position indoors. This sensor has been verified to give ±4mm position accuracy at 20Hz. The 2.1m diameter blimps in Figure 5.3 were scaled to carry a Sony VAIO and have a similar control architecture and identical interface. The blimps were designed to perform re-

connaissance and classification tasks in conjunction with the rovers that act as strike vehicles. The blimps can also be used to map the environment for the rovers.

Indoor results using the rovers to perform a representative SEAD mission with dynamic tasking using RHTA is shown in Figures 5.4 and 5.5. The four rovers are tasked to execute a SEAD mission in an environment with one centrally located avoidance region and a removable SAM site (Figure 5.4). They are divided into two sub-teams of strike and bomb damage assessment (BDA) vehicles, and each is assigned two targets. For this scenario, BDA tasks are required to occur after each strike task, and tasks were assumed completed when the vehicle passes within a specified tolerance of the target point. The ability to dynamically reassign tasks is demonstrated when one of the BDA vehicles (rover 2) is stopped before completing all of its tasks and the second BDA-capable vehicle (rover 4) must finish the mission. A second dynamic aspect of this scenario is demonstrated by giving one strike vehicle (rover 3) the capability to eliminate the SAM site (represented by the dashed lines). Once that task was completed, rover 4 can pass through the region, and it takes advantage of this by cutting the corner to reach the first BDA more quickly. Rover 4 then completes the remaining BDA tasks by passing around the remaining obstacle in the field.

Figures 5.7 to 5.11 show the results for a second scenario with 4 rovers and 14 tasks. The environment is dynamic and RHTA with $m = 2$ is used for on-line reassignment. The time step for this problem was set at 2 seconds, which was sufficient time to communicate the vehicle state information, design new trajectories, and, if necessary, re-compute the task assignment. The mission starts with 4 rovers (Figure 5.7), but after only a few steps into this initial plan, rover 4 is lost (as shown in Figure 5.6). Figure 5.8 shows the new plan where the tasks of rover 4 have been reassigned to the other rovers. At a later point in the mission, the team realizes that two tasks (9 and 11) are located at positions that differ from those previously assumed, and a further round of replanning occurs (Figure 5.9). Figure 5.10 shows the results of the last re-plan which happens when two new tasks (15 and 16) are discovered and rover 3 is assigned to visit these locations. Figure 5.11 shows the waypoints generated by

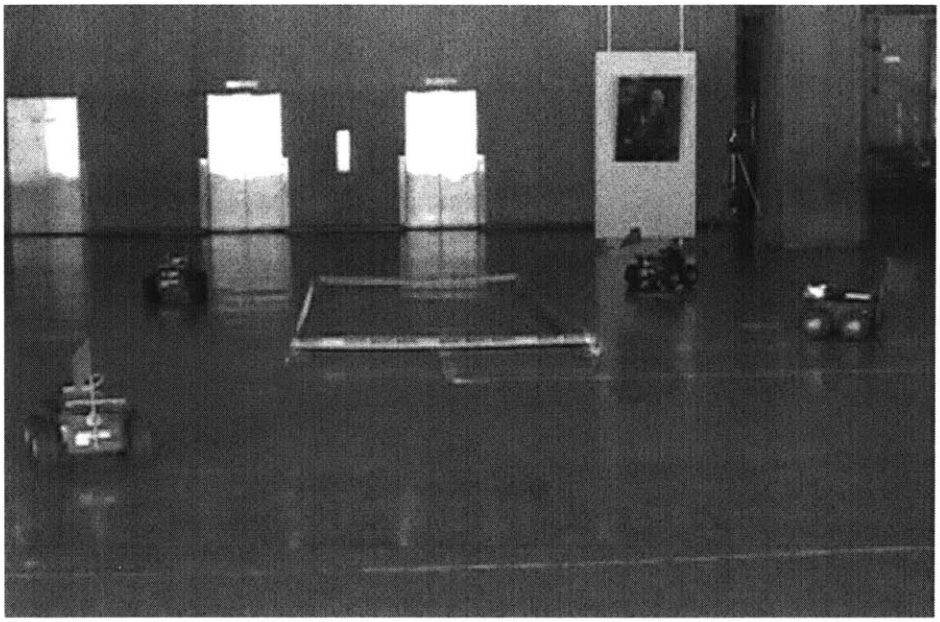


Fig. 5.4: 4 Rover experimental results associated with Figure 5.5.

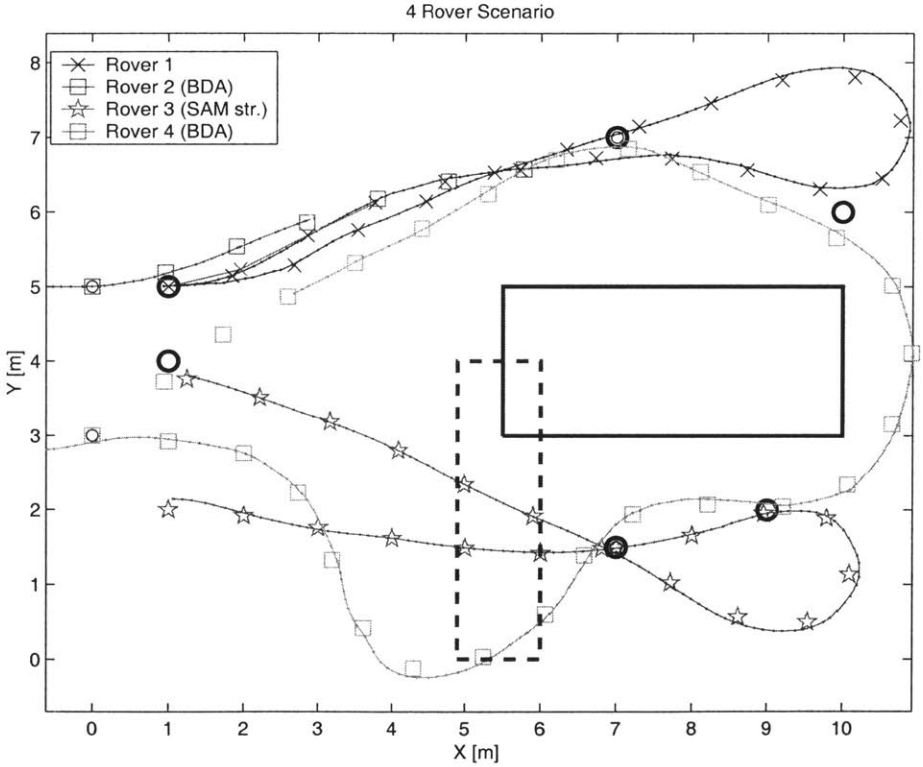


Fig. 5.5: 4 Rover experimental data from a typical SEAD-like mission.

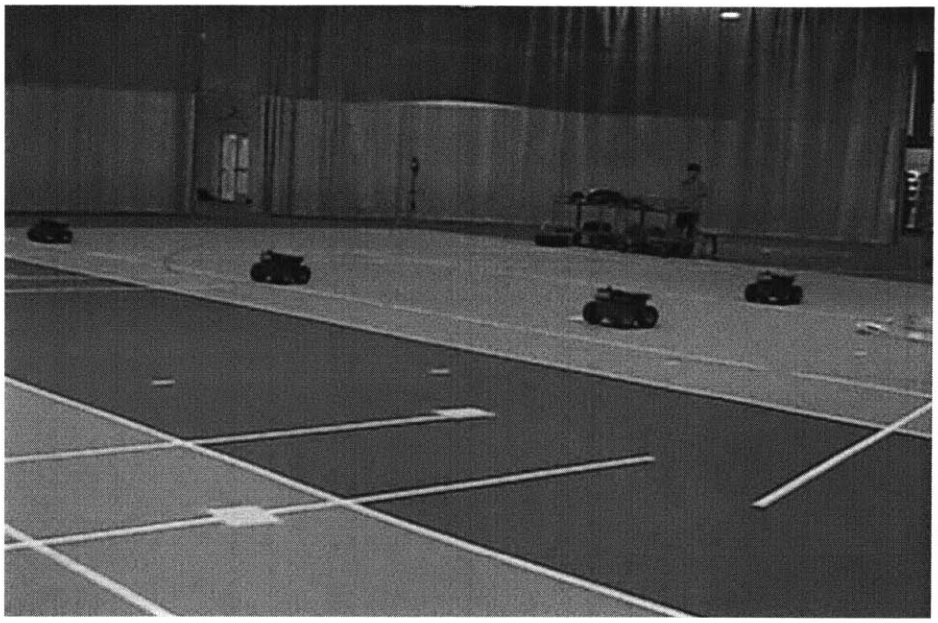


Fig. 5.6: 4 Rover experimental data associated with Figures 5.7 to 5.11.

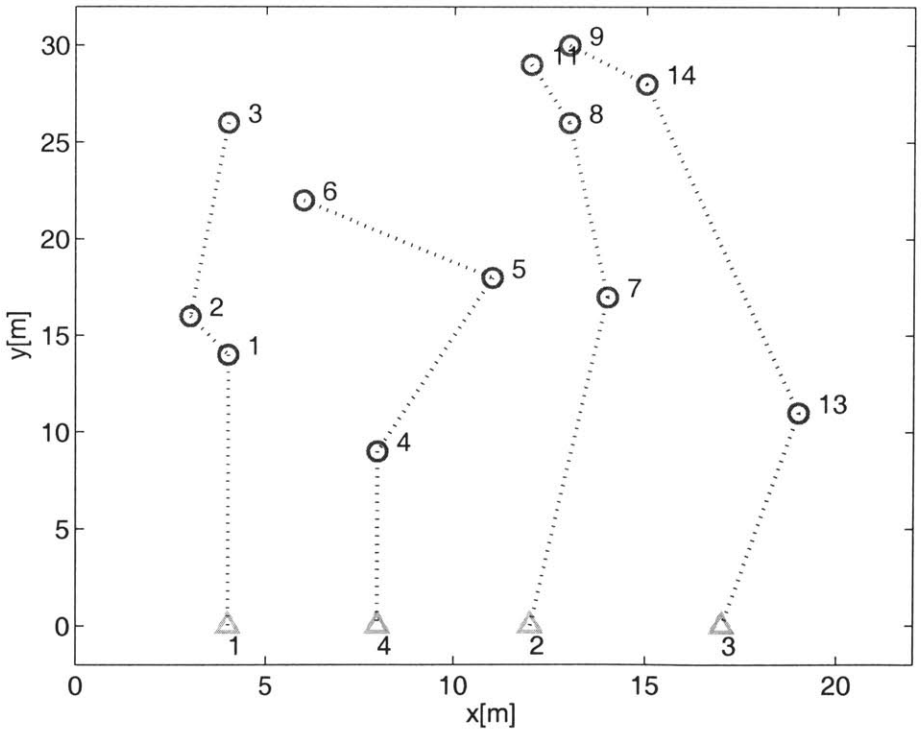


Fig. 5.7: Initial Plan

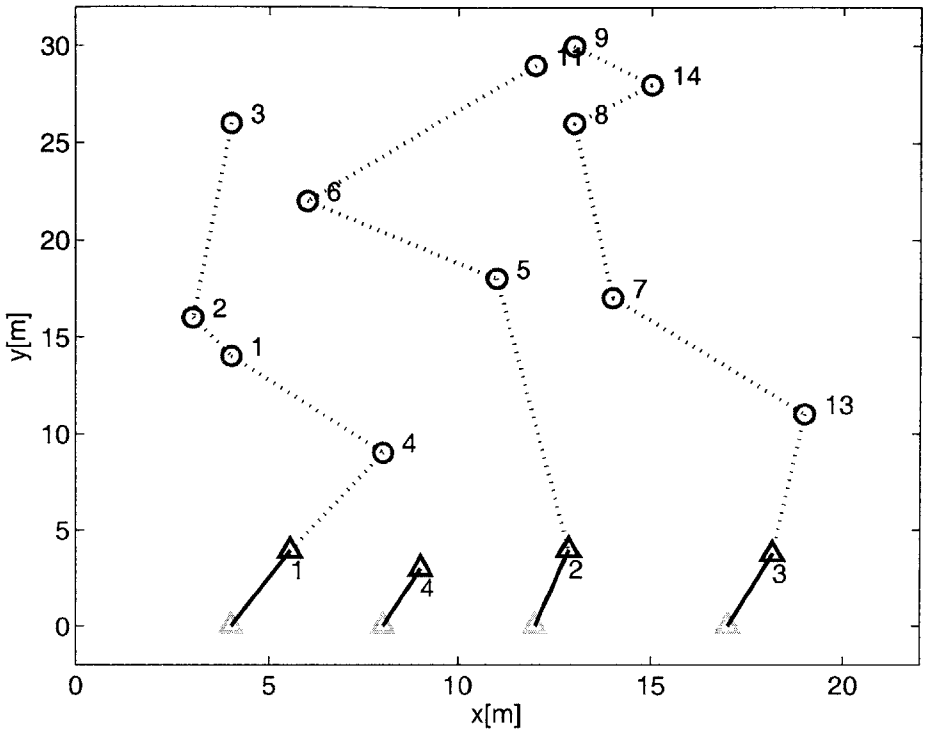


Fig. 5.8: Plan after first re-assignment (rover 1 is lost).

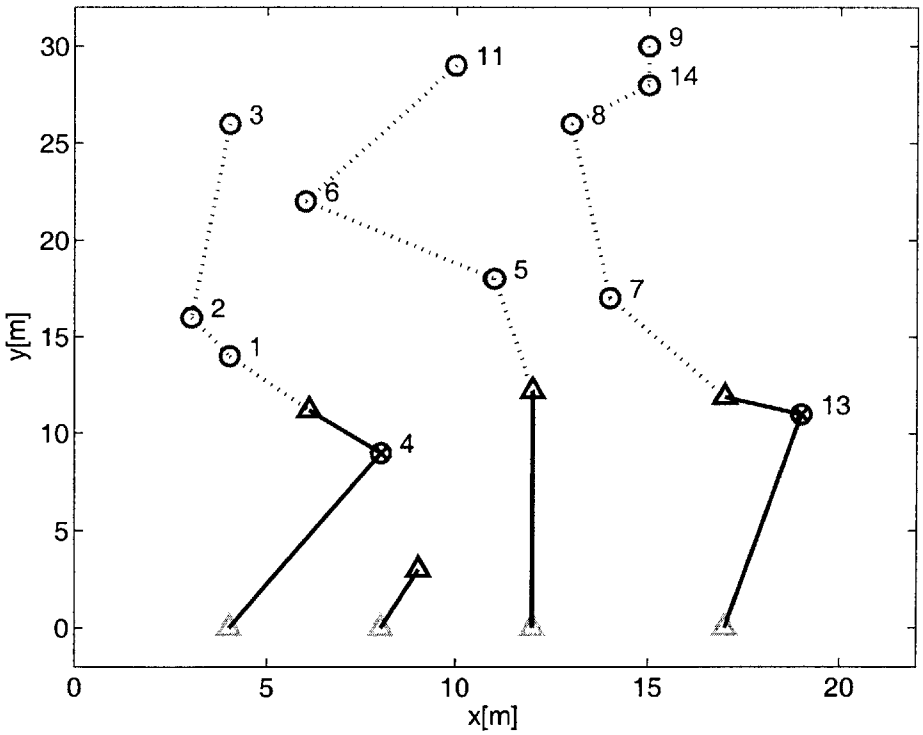


Fig. 5.9: Assignment after change in the location of targets 9 and 11

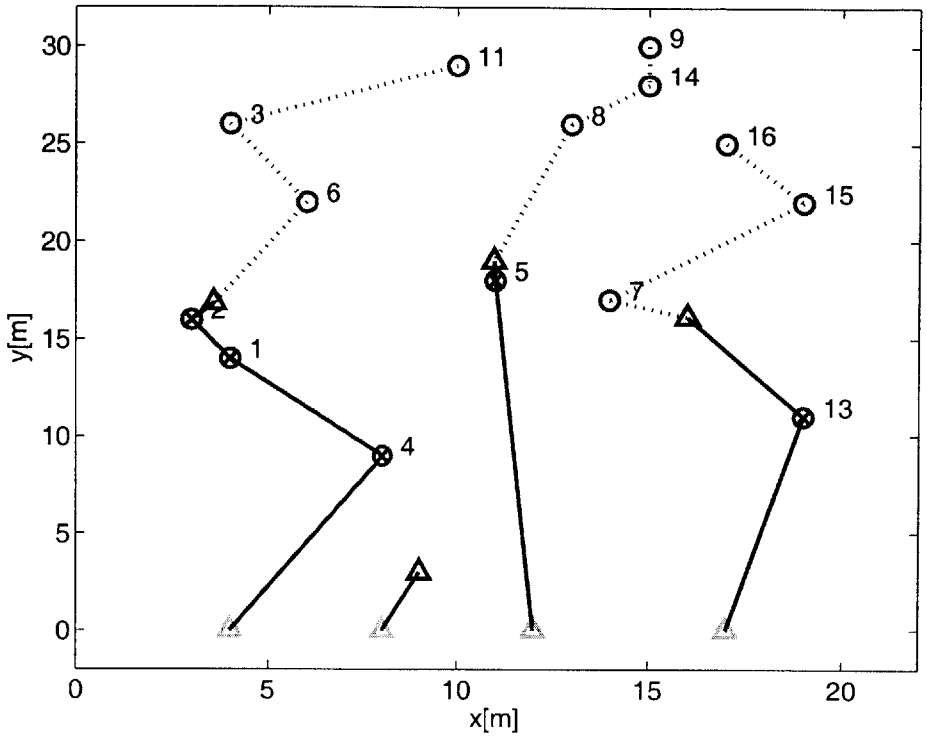


Fig. 5.10: Last assignment (new tasks at wpts 15 and 16 appear).

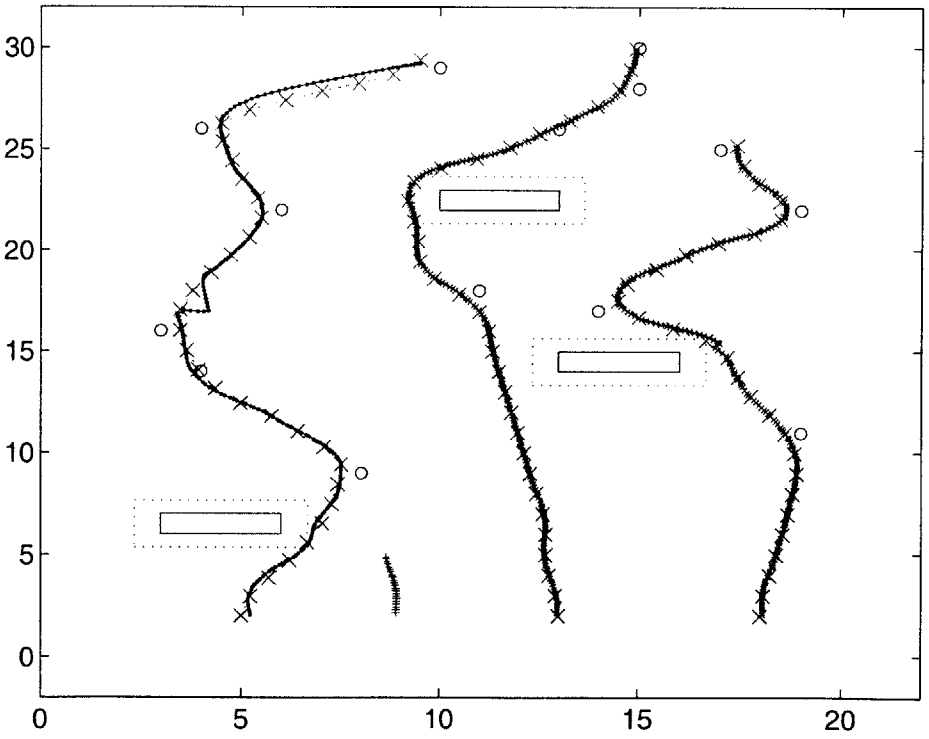


Fig. 5.11: Rover trajectories as measured during the experiment.

Fig. 5.12: 6 of 8 UAVs.

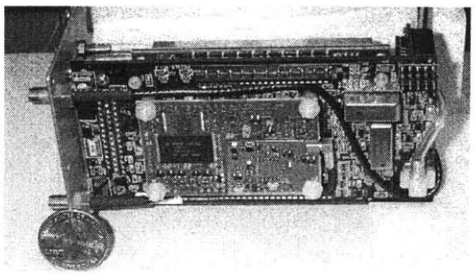


Fig. 5.13: Piccolo^{TM} autopilot from Cloud Cap Tech.

the planner and the actual trajectories of the rovers around the obstacles. The same scenario was also implemented using the greedy ($m = 1$) algorithm, and the time-discounted score accumulated by the rovers was 567. This score is approximately 10% lower than the 625 score achieved using RHTA with $m = 2$. These results confirm that RHTA with $m = 2$ can be implemented on a real-time testbed and yields better performance than an iterated greedy approach ($m = 1$).

5.1.2 UAV Testbed

The UAV testbed is a fleet of 8 UAVs (Figure 5.12) that are flown autonomously using the Cloud Cap commercial autopilot interfaced directly with the planning and task assignment algorithms. Figure 5.13 shows the 7.5oz Piccolo autopilot from Cloud Cap Technologies. Small aircraft (PT-60 sized trainers) were purposefully chosen to reduce operational complexity while still providing a high degree of flexibility in the missions that can be performed. The large trainer wing and Saito-91 four-stroke engine allow an additional two pounds of payload for sensor or communications upgrades. Twenty minute flights are easily achievable in the current configuration, and further extensions are possible. The UAV testbed has been operated autonomously on numerous occasions, and the flight results demonstrated the ability to track waypoints

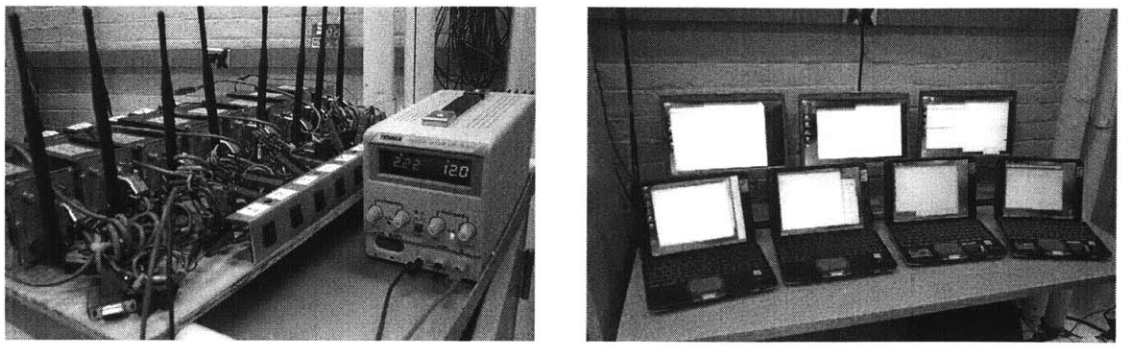
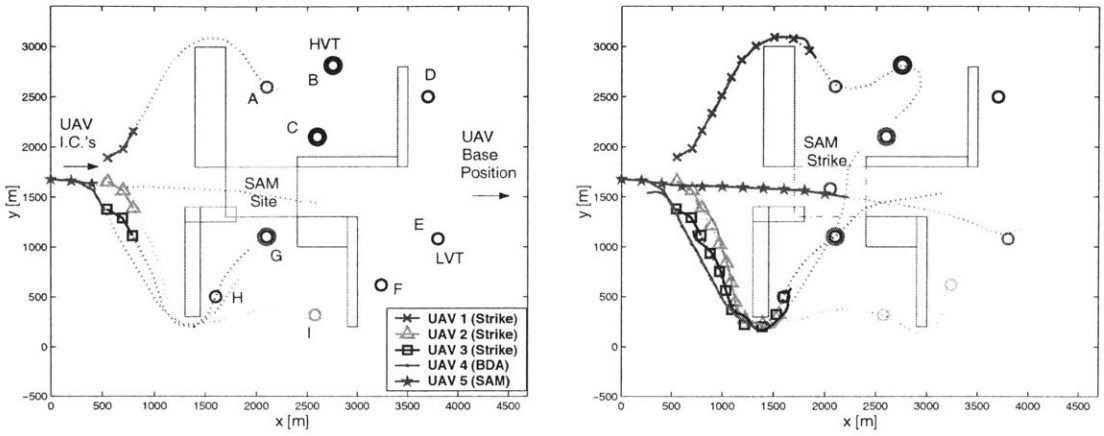


Fig. 5.14: Hardware-in-the-loop UAV testbed.

and maintain altitude in the presence of moderate wind disturbances.

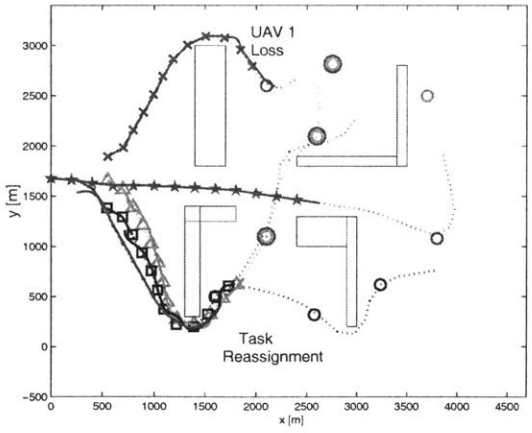
The autopilots also have an extensive hardware-in-the-loop (Figure 5.14) simulation capability, which can be used to demonstrate a full suite of coordination and control algorithms. The entire task assignment and trajectory design algorithms have been run off-board and were uplinked to the UAV autopilots in real-time, exactly as they will be during flight. Figure 5.15 shows experimental results from one such hardware-in-the-loop simulation involving 5 UAVs, and a mixture of both high and low value targets in a complex environment. For this scenario, high value targets (HVT) $\{B,C,G\}$ require both a strike and subsequent BDA, while the remaining low value targets (LVT) require only a strike task by a single UAV. UAV 5 is also given the capability to remove the centrally located SAM site, which the other UAVs are not permitted to enter.

The initial assignments send strike UAVs 1, 2 and 3 around the obstacles to their targets, while UAV 4 is assigned the first BDA task (Figure 5.15(a)). Note that a typical mission *timing constraint* is also shown by requiring that strike task D be completed only after the BDA for target B has been accomplished. UAV 5 is assigned to take out the SAM site which would then permit the strike vehicles to pass through the central region (Figure 5.15(b)). In this scenario, UAV 1 suddenly fails after reaching target A, and the remaining tasks are re-assigned to the rest of the team using the RHTA algorithm (Figure 5.15(c)). Figure 5.15(d) shows the completed mission after all tasks have been completed.

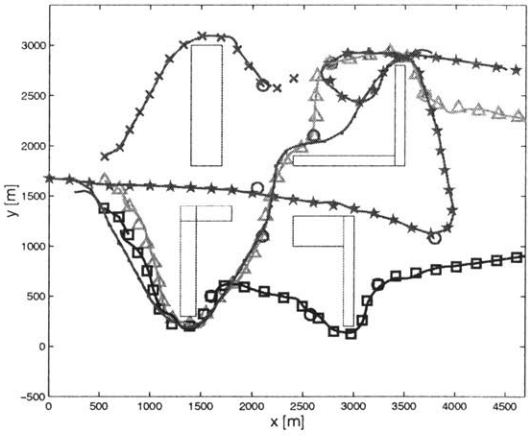


(a) Initial conditions and task assignments. Solid blue lines are obstacles

(b) UAV5 strikes SAM site, allows other UAVs to pass



(c) UAV1 lost, requires task re-assignment



(d) Completed mission with all tasks achieved

Fig. 5.15: Five UAV mission with dynamic task assignment using RHTA. Paths show both the optimal planned trajectories and the hardware-in-the-loop autopilot responses (with simulated wind turbulence).

5.2 MICA OEP Experiment

The Open Experimental Platform (OEP) [29] was developed by the Boeing Company for evaluation of hierarchical planning and coordinated control methodologies as part of the DARPA MICA program [5]. The OEP incorporates modeling and simulation capabilities to analyze and test planning algorithms. Figure 5.16 gives a schematic of the hierarchy of the MIT planning and control algorithms that were implemented.

MIT's Cooperative Path Planning (CPP) contains three modules: "Coarse Router" (finding shortest path around the obstacles using straight line approximation), "MILP Task Assignment" (receding horizon task assignment algorithm), and "RH MILP Path Planning" (trajectory designer), which are integrated into the overall planning hierarchy as seen in Figure 5.16. In this setup, our algorithms used the output of JobShop, a coarse scheduling algorithm. We then performed a detailed scheduling (assigned a specific vehicle to each task in the rank ordering given) and designed the detailed UAV trajectories. The list of waypoints (with events and event timing) were then executed within the OEP. At each iteration, information about the environment was obtained and compared with predictions to identify changes (removal of a target, identification of new targets, loss of UAVs, etc.). The planner then returned a detailed waypoint list for the UAVs, including the paths to follow and tasks to perform. A schematic of the controller used is shown in Figure 5.17.

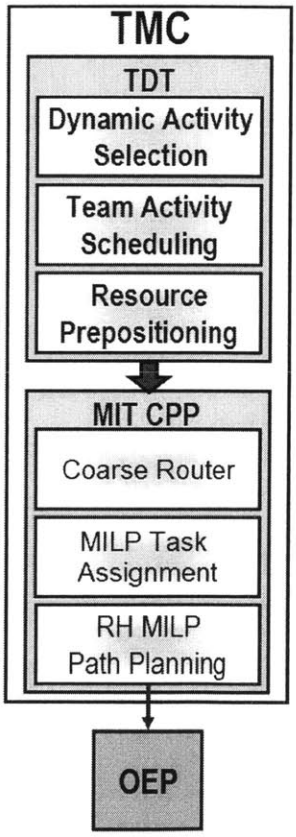


Fig. 5.16: MIT CPP algorithms inside in the planning hierarchy.

Results from a typical scenario are shown in Figures 5.18 to 5.20. The scenario has 6 UAVs of three types, classification, strike and BDA [29]. There are approximately 25 targets that have a wide variation in value (ranging from 10 to 300). There are a total of 10 SAM

sites (both medium and long range), two of which are unknown and possibly mobile. The initial goal of the mission is to use the UAV team to clear out the SAM sites in this area of operation. However, two previously unknown, very high value targets appear almost immediately after the start of the mission and the Operator provides an input that there is a strong desire to strike these new targets. The results show (Figure 5.18) some of the UAVs being diverted to hit the new HVTs with the rest of the team taking over new roles. The medium SAM sites are then targeted and several strike missions are executed. BDA tasks are assigned to occur after the strike missions are performed.

Figures 5.19 and 5.20 show that the team successfully attacks most of the medium SAM sites, but several of the UAVs were shot down as they attacked the medium SAM sites protected by long SAMs. For UAVs to attack these medium SAMs it is necessary to go inside the range of long SAMs which is a high risk environment. The key point was that the receding horizon task assignment algorithm was very efficient at "reacting" to changes in the environment to improve the paths of the UAVs to the targets when a SAM is removed. However, it was not very "proactive" in creating changes that would benefit the rest of the team. This motivated the design of the cooperative task assignment algorithm discussed in Chapter 4. The proactive nature of that assignment algorithm generates more cooperative behavior as members of the team choose to hit some targets just to reduce the path length/risk for other UAVs. For instance, a cooperative plan for the example mentioned above would first attack the long-range SAMs to reduce the risk of the paths for the UAVs assigned to medium SAMs.

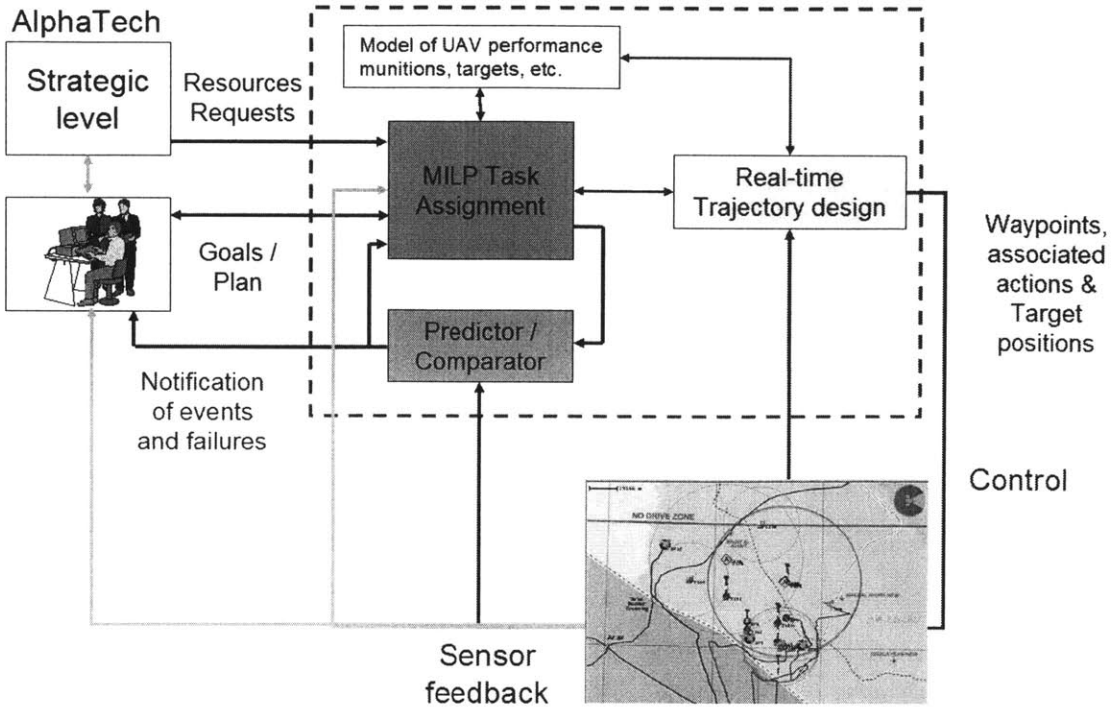


Fig. 5.17: Figure shows the overall MIT controller implemented on the OEP. The dark solid lines show loops that were closed in these experiments.

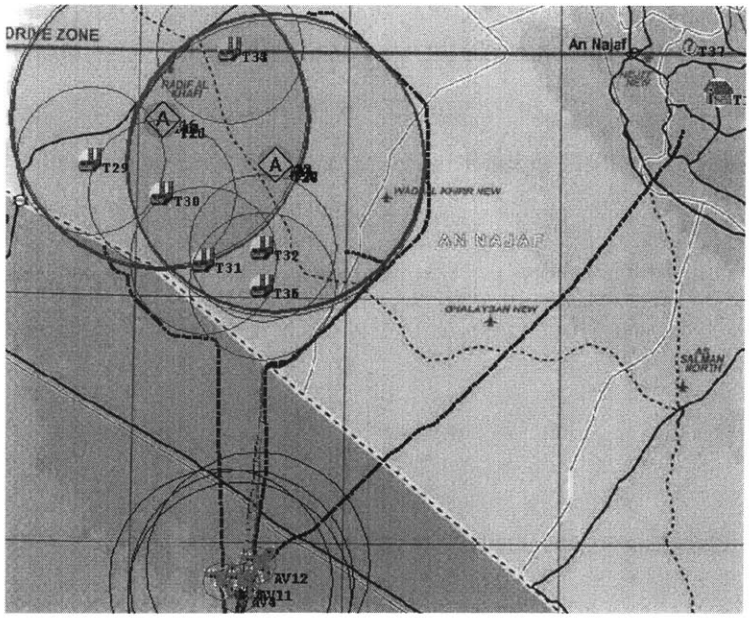


Fig. 5.18: Planned trajectories during the second loop closure. Note the UAV tasked to hit the HVT target top-right and the UAV trajectories skirting the SAM site to avoid detection.

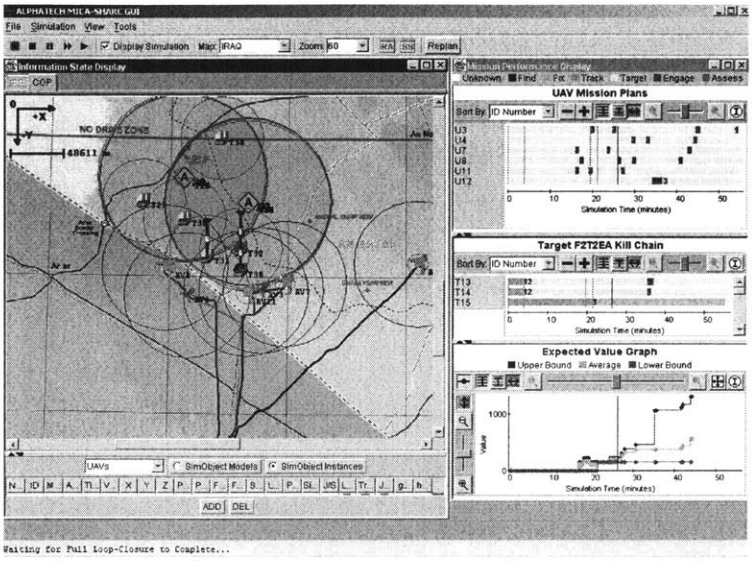


Fig. 5.19: Shows diagnostics available to evaluate progress of the controller. Results shown after the seventh loop closure. Most medium SAM sites have been hit and/or targeted.

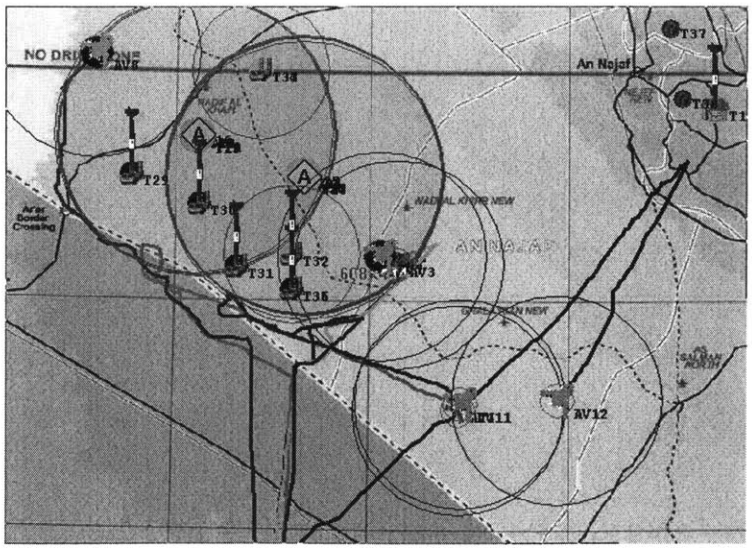


Fig. 5.20: Results after the eleventh loop closure. Most of the medium SAMs have been removed, but UAVs were not as successful against the long SAMs.

5.3 Conclusions

This chapter presented both hardware demonstrations of the receding horizon task assignment on the rover and UAV testbeds and simulations on the OEP. The multi-vehicle testbeds provide unique platforms to evaluate various distributed coordination and control strategies in real-time with real vehicles and environments. The results illustrated that RHTA can be implemented for real-time replanning in dynamic environments. In the second part, a set of experiments on Boeing's OEP was presented. In these experiments RHTA was used for real-time task assignment in a risky and dynamic environment. The result show that RHTA performs well in reacting to changes in the environment but the generated plans were rather aggressive. These result motivated the idea of cooperative assignment that was discussed in Chapter 4 to generate more cooperative, risk averse assignment. The hardware implementation of the algorithms and the results of these experiments allowed us to validate the theory that was discussed in previous chapters.

Chapter 6

Conclusions

This thesis has addressed several issues in the UAV task assignment problem for dynamic and uncertain environments. The following describes the contribution of this thesis in each of these areas.

Chapter 2 presented a receding horizon formulation of the UAV task assignment algorithm. This formulation uses the idea of "model predictive control" (MPC) in the assignment problem and solves the problem in an iterative fashion. In each iteration a limited horizon assignment problem is formed using "petal" formulation. Each of these problems is a small Mixed-Integer Linear Programming (MILP) problem that can be solved very rapidly using commercially available software such as CPLEX. Timing constraints, which are examples of *hard* side constraints, typically make the assignment problems much more difficult to solve, but they can be implemented in the receding horizon task assignment (RHTA) algorithm without changing the complexity of the problem. This is achieved by implementing the constraints outside the MILP optimization. The RHTA formulation is quite general, and many other constraints can be included, such as the capacity constraints (*i.e.*, munition constraints). These types of constraints were shown to cause other, more myopic, iterative methods (*i.e.*, iterative greedy) to become infeasible or yield degraded performance. Violation of these constraints is included in RHTA as a penalty term in the cost function, which helps to ensure the feasibility of the future plans. These penalty terms can be regarded as a very simple cost-to-go function that provides a coarse estimate of the score of

the remainder of the mission. This approach does not try to enumerate all (or many) of the possible future plans to determine a "good" cost estimate, but instead it easily identifies the future implications of "bad" short-term choices. Thus RHTA offers a good trade between performance and computational effort.

Simulation results using RHTA for several scenarios were presented. The results illustrated that RHTA can be implemented in real-time for replanning. They also showed that RHTA is capable of solving large problems in reasonable computation times. Comparing the results of RHTA to the greedy methods, which are known to be fast to calculate, confirmed that the RHTA algorithm creates far better solutions than the greedy methods without a substantial increase in the complexity of the problem (or computation time). The results of the RHTA algorithm for different values of petal size (m) were also compared and the results showed that there is a tradeoff between performance (degree of sub-optimality) and computation time, and that this can be tuned by the choice of the petal size. The examples showed that $m = 2$ gives the best result in which the computation time is small enough for real-time implementation and the performance is close to optimal.

Chapter 3 discussed the issues associated with a fast replanning rate. It was argued that fast replanning, which is crucial in a dynamic environment, can cause *instability* and/or *churning* when the data is noisy or the environment is uncertain. The frequency domain interpretation as originally formulated in Ref. [31] was extended in Chapter 3 using correlation as a metric. A new filtering methodology was introduced to attenuate the effect of noise in the environment in the assignment problem. The assignment algorithm was then reformulated to include this filtering scheme, and simulation results were presented to show the effectiveness of this approach. The results showed that this formulation can eliminate churning and instability by tracking the signal and rejecting the noise.

Chapter 4 discussed the problem of risk in the environment and a new stochastic formulation of UAV task assignment problem was presented. This formulation explicitly accounts for the interaction between the UAVs – displaying *cooperation* between the vehicles rather than just *coordination*. Cooperation entails coordinated task as-

signment with the additional knowledge of the future implications of a UAV's actions on improving the expected performance of the other UAVs. The key point is that the actions of one UAV can reduce the risk in the environment for the other UAVs; and the new formulation takes advantage of this fact to generate cooperative assignments to achieve better performance. Chapter 4 further extends the notion of cooperation to the weapon target assignment (WTA) problem. The problem was formulated as a dynamic programming (DP) problem. A comparison with other approaches showed that including this cooperation lead to a significant increase in performance. Two DP approximation methods (the one-step and two-step lookahead) were also developed for the large problem where curse of dimensionality in DP is prohibitive. Simulation results showed that the one-step lookahead can generate a cooperative solution very quickly, but the performance degrades considerably. The two-step lookahead policy generated plans which are very close to (and in most cases, identical to) the optimal solution.

Chapter 5 presented the hardware and simulation results for the RHTA algorithm. Two hardware testbeds were used to implement the RHTA in real-time and the results illustrated that RHTA is capable of reacting to the changes in the environment and replanning in real-time. The RHTA algorithm was also implemented in the Boeing's OEP demo which resembles a real-world environment. The results showed that RHTA does a good job in reacting to the changes in the environments. However since the environment is risky (due to existence of SAM sites), the performance was relatively poor because this risk was neglected in the basic assignment. This further confirms the need for taking into account the risk of environment and generating a risk averse, cooperative plan.

The algorithms developed in this thesis have demonstrated the importance of incorporating risk and uncertainty in high level planning algorithms. These algorithms were demonstrated to work on hardware testbeds that simulate real-world UAV operations.

Bibliography

[1] U. S. A. Board, *UAV Technologies and combat operations,* Tech. Rep. Tech. Tep. SAB-TR-96-01, November 1996.

[2] A. Kott, *Advanced Technology Concepts for Command and Control,* Xlibris Corporation.

[3] O. of the Secretary of Defense, *Unmanned Aerial Vehicles Roadmap,* technical report, December 2002.

[4] P. R. Chandler, S. Rasmussen, "UAV Cooperative Path-Planning," In *Proceedings of the AIAA Guidance, Navigation and Control Conference,* 2000. AIAA-2000-4370.

[5] S. A. Heise, *DARPA Industry Day Briefing,* 2001.

[6] P. R. Chandler, M. Pachter, D. Swaroop, J.M. Fowler, J. K. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard, "Complexity in UAV cooperative control," In *Proceedings of the American Control Conference*, Anchorage AK, May 2002.

[7] C. Schumacher, P. R. Chandler and S. Rasmussen, "Task Allocation for Wide Area Search Munitions via Network Flow Optimization," In *Proceedings of the American Control Conference,* IEEE, May 2002.

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

[9] A. G. Richards, J. S. Bellingham, M. Tillerson and J. .P How, "Co-ordination and Control of Multiple UAVs," In *Proceedings of the AIAA Guidance, Navigation and Control Conference,* 2002.

[10] J. S. Bellingham, A. G. Richards and J. P. How, "Receding Horizon Control of Autonomous Aerial Vehicles", In *Proceedings of the American Control Conference*, Anchorage AK, May 2002.

[11] J. S. Bellingham, Y. Kuwata, and J. How, "Stable Receding Horizon Trajectory Control for Complex Environments," In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Austin, TX), Aug 2003.

[12] 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.

[13] M. Alighanbari, Y. Kuwata, and J. P. How, "Coordination and Control of Multiple UAVs with Timing Constraints and Loitering," In *Proceedings of the American Control Conference*, June 2003.

[14] J. Wohletz, "Cooperative, Dynamic Mission Control for Uncertain, Multi-Vehicle Autonomous Systems," special session presented at the IEEE *Conference on Decision and Control*, Dec. 2002.

[15] T. McLain, P. Chandler, S. Rasmussen, and M. Pachter. "Cooperative control of UAV rendezvous," In *Proceedings of the American Control Conference*, pages 2309–2314, Arlington, VA, June 2001.

[16] M. Goodrich R. W. Beard, T. W. McLain, "Coordinated target assignment and intercept for unmanned air vehicles," In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC, May 2002.

[17] K. Passino, M. Polycarpou, D. Jacques, M. Pachter, Y. Liu, Y. Yang, M. Flint, and M. Baum, "Cooperative Control for Autonomous Air Vehicles," Chapter

12 in R. Murphy and P. Pardalos, eds., *Cooperative Control and Optimization,* Kluwer Academic Publishers, Dordrecht, Netherlands, 2002.

[18] T. Schouwenaars, B. De Moor, E. Feron, and J. How. "Mixed integer programming for multi-vehicle path planning," In *Proceedings of the European Control Conference*, Porto, Portugal, September 2001.

[19] T. W. McLain and R. W. Beard, "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," In *Proceedings of the American Control Conference*, June 2003.

[20] R. Olfati-Saber, W. B. Dunbar, and R. M. Murray "Cooperative Control of Multi-Vehicle Systems Using Graphs and Optimization," In *Proceedings of the American Control Conference*, June 2003.

[21] W. Kang and A. Sparks "Task Assignment in the Cooperative Control of Multiple UAVs," In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, (Austin, TX), Aug 2003.

[22] Y. Jin, M. M. Polycarpou, and A. Minai, "Cooperative Real-Time Task Allocation Among Groups of UAVs," *Recent Development in Cooperative Control and Optimization*, S. Butenko, R. Murphey, and P. Pardalos eds., Kluwer Academic Publishers, 2004.

[23] P. Toth and D. Vigo, *The Vehicle Routing Problem,* SIAM, Philadelphia, 2002.

[24] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms,* MIT Press/McGraw-Hill, 1990.

[25] C. A. Floudas, *Nonlinear and Mixed-Integer Programming – Fundamentals and Applications,* Oxford University Press, 1995.

[26] M. Gendreau, G. Laporte and J. Y. Ptvin, "Metaheuristics for the Capacitated VRP," in *The Vehicle Routing Problem* edited by P. Toth and D. Vigo, SIAM, Philadelphia, 2002.

[27] K. P. O'Rourke, T. G. Bailey, R. Hill and W. B. Carlton, "Dynamic Routing of Unmanned Aerial Vehicles using Reactive Tabu Search," In *Proceedings of the 67th MORS Symposium,* Nov. 1999.

[28] J. L. Ryan, T. G. Bailey, and J. T. Moore, "Reactive tabu search in unmanned aerial reconnaissance simulations," In D. J. Medeiros et al., editor, *Proceedings of the 1998 Winter Simulation Conference,* 1998. e", IEEE, May 2002.

[29] *MICA OEP user Guide,* Boeing Company, Revision 10.

[30] R. Kastner, C. Hsieh, M. Potkonjak, and M. Sarrafzadeh, "On the Sensitivity of Incremental Algorithms for Combinatorial Auctions," WECWIS 2002: Newport Beach, California, USA.

[31] J. Tierno and A. Khalak, "Frequency Domain Control Synthesis For Time-Critical Planning," In *Proceedings of the IEE European Control Conference,* Sept. 2003.

[32] A. G. Richards, J. P. How, "Aircraft Trajectory Planning with Collision Avoidance using Mixed Integer Linear Programming," In *Proceedings of the American Control Conference,* IEEE, May 2002.

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

[34] ILOG, *ILOG CPLEX User's guide,* 1999.

[35] J.M. Maciejowski, *Predictive Control with Constraints,* Prentice Hall, England, 2002.

[36] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. M. Scokaert, "Constrained Model Predictive Control: Stability and Optimality," *Automatica,* 36(2000), Pergamon Press, UK, p. 789–814.

[37] J. W. Curtis, "Churning: Repeated Optimization and Cooperative Stability," *Recent Development in Cooperative Control and Optimization*, S. Butenko, R. Murphey, and P. Pardalos eds., Kluwer Academic Publishers, 2004.

[38] J. M. Mulvey, R. J. Vanderbei, S. A. Zenios, "Robust optimization of large-scale systems," *Oper. Res.* Vol. 43 (1995) pp. 264–281.

[39] M. Sim and D. Bertsimas, "The Price of Robustness," *MIT Sloan Internal Tech. Report,* 2002.

[40] L. F. Bertuccelli, *Robust Planning for Heterogeneous UAVs in Uncertain Environments,* SM Thesis, MIT Department of Aeronautics and Astronautics, May 2004.

[41] J. W. Curtis, and R. Murphey, "Simultaneous Area Search and Task Assignment for a Team of Cooperative Agents," In *Proceedings of the AIAA Guidance, Navigation and Control Conference,* (Austin, TX), Aug 2003.

[42] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Cooperative Path Planning for Multiple UAVs in Dynamic and Uncertain Environment," In *Proceeding of IEEE Conference on Decision and Control,* Dec. 2002.

[43] Personal communication with Jorge Tierno, Sept. 2003.

[44] Percival and Walden, *Spectral Analysis for Physical Applications*, Cambridge University Press, 1993.

[45] R. A. Murphey, "An approximate algorithm for a weapon target assignment stochastic program," In *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems.* Kluwer Academic Publishers, 1999.

[46] J. S. Bellingham, *Coordination and Control of UAV Fleets using Mixed-Integer Linear Programming,* SM Thesis, MIT Department of Aeronautics and Astronautics, Aug. 2002.

[47] P. Hosein, and M. Athans, "The Dynamic Weapon Target Assignment Problem." *Proc. of Symposium on C^2 Research*, Washington, D.C. 1989.

[48] R. A. Murphey, "Target-Based Weapon Target Assignment Problems," in *Nonlinear Assignment Problems: Algorithms and Applications*, P. M. Pardalos and L. S. Pitsoulis eds., Kluwer Academic Publishers, 2000.

[49] R K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin. "Exact and heuristic algorithms for the weapon-target assignment problem". Submitted to Operations Research, 2003.

[50] P. Hossein, and M. Athans, "An Asymptotic Result for the Multi-Stage Weapon-Target Allocation Problem," In *Proceeding of IEEE Conference on Decision and Control,* Dec. 1990.

[51] D. P. Bertsekas, *Dynamic Programming and Optimal Control,* Athena Scientific, Belmont, Massachusetts, 2000.

[52] E. T. King, M. Alighanbari, Y. Kuwata, and J. P. How, "Coordination and Control Experiments on a Multi-vehicle Testbed," *Proceedings of the American Control Conference*, IEEE, June 2004.

[53] Y. Kuwata, *Real-time Trajectory Design for Unmanned Aerial Vehicles using Receding Horizon Control,* SM Thesis, MIT Department of Aeronautics and Astronautics, June 2003.

[54] ArcSecond, *Constellation 3D-i Error Budget and Specifications,* June, 2002. *http://www.arcsecond.com*

[55] ActivMedia Robotics, *Pioneer Operations Manual,* January 2003. *http://robots.activmedia.com*