Coordination and Control of UAV Fleets using Mixed-Integer Linear Programming

by

John Saunders Bellingham

Bachelor of Applied Science
University of Waterloo, 2000

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

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2002

© John Saunders Bellingham, MMII. All rights reserved.

The author hereby grants to MIT permission to reproduce
and distribute publicly paper and electronic copies
of this thesis document in whole or in part.

Author . . . .
Department of Aeronautics and Astronautics
August 9, 2002

Certified by . . . .
Jonathan P. How
Associate Professor
Thesis Supervisor

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

Coordination and Control of UAV Fleets using Mixed-Integer Linear Programming

by

John Saunders Bellingham

Submitted to the Department of Aeronautics and Astronautics
on August 9, 2002, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

This thesis considers two important topics in the coordination and control of fleets of UAVs; the allocation and trajectory design problems. The first allocates waypoints to individual vehicles in the fleet, and includes constraints on vehicle capability, waypoint visitation, and visit timing. Formulations of the allocation problem are presented to find minimum mission completion time and maximum stochastic expectation of mission benefit. The trajectory design problem provides a minimum time reference trajectory to the goal, while avoiding obstacles in the environment and obeying a limited turning rate.

Mixed-Integer Linear Programming (MILP) is applied to both problems in order to integrate discrete and continuous decisions, making into one optimization program. The MILP allocation program's cost function is evaluated using estimated trajectory parameters, which come from approximated paths. This partially decouples the allocation and trajectory design problems, and detailed trajectories can later be designed for the selected waypoint sequences. This significantly reduces the allocation solution time, with negligible loss in performance. The stochastic formulation is shown to recover sophisticated attrition reduction strategies.

MILP is applied to the trajectory design problem within a receding horizon control framework. The optimization uses a novel terminal penalty which approximates the cost to go to the goal, and is cognizant of intervening obstacles. This approach provides trajectories that are within 3% of optimal with significantly less computational effort, while avoiding entrapment in concave obstacles. This trajectory designer is modified to guarantee its stability, and the resulting controller is capable of planning trajectories in highly constrained environments, without large increases in computation time.

The approaches presented here successfully solve the allocation and trajectory design problems, offering good performance and computational tractability.

Thesis Supervisor: Jonathan P. How
Title: Associate Professor

Acknowledgments

This research was funded in part under DARPA contract # N6601-01-C-8075 and Air Force grant # F49620-01-1-0453.

Contents

1 Introduction **15**

1.1 The UAV Trajectory Design Problem 16

1.1.1 Summary of Previous Work on UAV Trajectory Design 16

1.1.2 Outline of Trajectory Design Approach 18

1.2 The Allocation Problem 19

1.2.1 Summary Previous Work on the Allocation Problem 21

1.2.2 Outline of Allocation Problem Approach 23

2 Receding Horizon Control Trajectory Design **25**

2.1 Introduction 25

2.2 Fixed Horizon Minimum Time Controller 26

2.3 Simple Terminal Cost Formulation 28

2.4 Improved Receding Horizon Control Strategy 30

2.4.1 Control Architecture 30

2.4.2 Computation of Cost Map 32

2.4.3 Modified MILP Problem 35

2.5 Results 36

2.5.1 Avoidance of Entrapment 36

2.5.2 Performance 37

2.5.3 Computation Savings 38

2.5.4 Hardware Testing Results 40

2.5.5 Planning with Incomplete Information 41
2.6 Conclusions 45

3 Stable Receding Horizon Trajectory Design **47**
3.1 Review of Stability Analysis Techniques 48
3.2 Stable Receding Horizon Trajectory Designer 52
3.2.1 Terminal Constraint Set 53
3.2.2 The Tree of Trajectory Segments 54
3.2.3 Stability Analysis 57
3.3 MILP Formulation of Controller 58
3.3.1 Tangency Constraint 60
3.3.2 Selection of Unit Vectors in P 62
3.3.3 Arc Length Approximation 64
3.3.4 Linearized Terminal Penalty 66
3.4 Simulation Results 67
3.4.1 Operation of the Trajectory Designer 67
3.4.2 Trajectory Design in Highly Constrained Environments 68
3.4.3 Design of Long Trajectories 70
3.4.4 Rate of Decrease of Terminal Penalty 72
3.5 Conclusions 73

4 Minimum Time Allocation **75**
4.1 Introduction 75
4.2 Problem Formulation 76
4.3 The Allocation Algorithm 78
4.3.1 Overview 78
4.3.2 Finding Feasible Permutations and their Costs 79
4.3.3 Task Allocation 82
4.3.4 Modified Cost: Total Mission Time 83

4.3.5 Timing Constraints . . . 84
4.4 Simulations . . . 85
4.4.1 Simple Allocation Example . . . 85
4.4.2 Large-Scale Comparison to Fully Coupled Approach . . . 86
4.4.3 Complex Allocation Problem . . . 89
4.5 Conclusions . . . 90

5 Maximum Expected Score Allocation **93**
5.1 Introduction . . . 93
5.2 Optimization Program Formulations . . . 94
5.2.1 Purely Deterministic Formulation . . . 95
5.2.2 Deterministic Equivalent of Stochastic Formulation . . . 97
5.2.3 Stochastic Formulation . . . 98
5.3 Results . . . 103
5.3.1 Nominal Environment . . . 103
5.3.2 High Threat Environments . . . 104
5.3.3 Results on Larger Problem . . . 104

6 Conclusions **109**
6.1 Contributions to the Trajectory Design Problem . . . 109
6.2 Contributions to the Allocation Problem . . . 110
6.3 Future Work . . . 111

List of Figures

1-1 Schematic of a Typical Mission Scenario 20

2-1 Trajectory Planned using ϕ_2 as Terminal Penalty 30
2-2 Resolution Levels of the Planning Algorithm 31
2-3 Effect of Obstacles on Cost Values 34
2-4 Trajectories Designed with ϕ_3 as Terminal Penalty 37
2-5 Effects of Plan Length on Arrival Time and Computation Time 38
2-6 Cumulative Computation Time vs. Complexity 39
2-7 Sample Long Trajectory Designed using Receding Horizon Controller . . . 40
2-8 Planned and Executed Trajectories for Single Vehicle 41
2-9 Planned and Executed Trajectories for Two Vehicles 42
2-10 Trajectory Planning with Incomplete Information 43
2-11 Detection of Obstacle 43
2-12 Reaction to Obstacle 43
2-13 Resulting Trajectory Planned with Incomplete Information 43
2-14 Sample Long Trajectory Planned with Incomplete Information 44

3-1 Resolution Levels of the Stable Receding Horizon Trajectory Designer . . . 52
3-2 A Tree of Kinodynamically Feasible Trajectories to the Goal 55
3-3 The Trajectory Returned by Find-Connecting-Segment 57
3-4 The Vectors Involved in the Tangency Constraints 59
3-5 Trajectory between States Returned by Find-Connecting-Segment . . 63

3-6 Comparison of Upper Bounding Function $\|\hat{\mathbf{x}}(N) - \hat{\mathbf{x}}_{\mathrm{c}}\|_1$ to $|\theta|$. 64
3-7 Plans of the Stable Trajectory Designer 68
3-8 Trajectory Examples in Highly Constrained Environment 69
3-9 Tree of Trajectory Segments for Long Trajectory 71
3-10 Sample Long Trajectory Designed by Stable Controller 71
3-11 Rate of Decrease of Stable Controller's Terminal Penalty 72

4-1 Task Assignment and Trajectory Planning Algorithm 78
4-2 Distributed Task Assignment and Trajectory Planning Algorithm 80
4-3 Visibility Graph and Shortest Paths for the Allocation Problem 80
4-4 Simple Unconstrained Allocation Example 87
4-5 Simple Allocation Example with Added Capability Constraint 87
4-6 Simple Allocation Example with Added Timing Constraint 87
4-7 Simple Allocation Example with Added Obstacle 87
4-8 Comparison of Partially Decoupled and Fully Coupled Approaches 88
4-9 Large Allocation Problem 90

5-1 Example Purely Deterministic Allocation 96
5-2 Example Deterministic Equivalent Allocations 99
5-3 Piecewise Linear Approximation of the Exponential Function 101
5-4 Example Maximum Expected Score Allocation 102
5-5 Expected Score of Incumbent Solution vs. Time 106
5-6 Example Large Allocation Problem 107

List of Tables

3.1 Cumulative Solution Times for Long Trajectories 70

4.1 Computation Time for Small Allocation and Trajectory Design Problems . . 88
4.2 Computation Time for Random Allocation and Trajectory Design Problems 89

5.1 Results of Several Formulations in Probabilistic Environment 104
5.2 Expected Scores in More Threatening Environments 105
5.3 Probabilities of Reaching High Value Target in More Threatening Environments 105

Chapter 1

Introduction

This thesis examines coordination and control of fleets of Unmanned Aerial Vehicles (UAVs). UAVs offer advantages over conventional manned vehicles in many applications [3]. They can be used in situations too dangerous for manned vehicles, such as eliminating anti-aircraft defenses. Without being weighed down by systems required by a pilot, UAVs can also stay aloft longer on surveillance missions. While the roles and capabilities of UAVs are growing, current UAV control structures were conceived of with limited roles in mind for these vehicles, and operate separately from the command hierarchies of manned units. It is necessary to improve on this control structure in order to fully exploit the expanding capabilities of UAVs, and to enable them to act cohesively with manned and unmanned vehicles.

One approach to improving the UAV control structure is to provide planning and operational tools to UAV controllers for performing and task assignment for the fleet, and trajectory design to carry out these assignments. The control structure would then evaluate the overall fleet performance during operation, reacting quickly to changes in the environment and the fleet. These tools would allow operators to plan more complex missions functioning at a faster tempo. This thesis applies operations research and control system theory to the task assignment and trajectory optimization problems as a basis for such tools. These problems will now be described in greater detail.

1.1 The UAV Trajectory Design Problem

A significant aspect of the UAV control problem is the optimization of a minimum-time trajectory from the UAV's starting point to its goal. This trajectory is essentially planar, and is constrained by aircraft dynamics and obstacle avoidance. This optimization problem is difficult because it is non-convex due to the presence of obstacles, and because the space of possible control actions over a long trajectory is extremely large [23]. Simplifications that reduce its dimensionality while preserving feasibility and near-optimality are challenging.

1.1.1 Summary of Previous Work on UAV Trajectory Design

Two well-known methods that have been applied to this problem are Probabilistic Road Maps [14] (PRMs) and Rapidly-exploring Random Trees [17] (RRTs). PRMs are applied to reach a goal by adding small trajectory segments to larger, pre-computed routes. The RRT method extends a tree of trajectory segments from the starting point until the goal is reached. Each of the trajectory segments is found by selecting a point in the state space at random, and then connecting the closest point in the existing tree to this point. This technique has been further developed by applying quadratic optimization to trajectory segments before adding them to the RRT [13]. These methods all use some degree of randomness to sample the space of control actions, which makes tractable a category of problems with relatively high dimension. However, an accompanying effect is that the optimality of the resulting trajectories is limited by selecting the best from randomly sampled options. This may not be a disadvantage when feasibility is the important criterion, but if optimality is important, then randomized techniques may not be appropriate.

Another approach to trajectory design "smoothes" a path made up of straight line segments into a *flyable* trajectory that is dynamically feasible [21, 15]. This approach typically first constructs a Voronoi diagram of an environment in which anti-aircraft defenses are to be avoided. The Voronoi diagram is a network of connected edges that are positioned to maximize the distance from the two nearest anti-aircraft sites. Graph search techniques can be applied to find a path through the defenses. The combination of Voronoi diagrams and graph search provides a computationally tractable approach to planning minimum radar exposure paths.

These straight line segment paths through the Voronoi diagram can be smoothed by inserting fillets to construct the turning path of the vehicle at line segments intersections [21, 18, 28]. Smoothing can also be performed by replacing straight line segments with cubic splines [15]. These approaches do not account for obstacles in the environment, and cannot directly incorporate these constraints. While the fillet and spline construction technique are simple to apply and computationally tractable, they do not perform the smoothing optimally. These techniques can also cause difficulties when the straight line path constrains short edges joined by tight angles, making the straight line path difficult to smooth.

MILP has also been applied to trajectory design. MILP extends linear programming to include variables that are constrained to take on integer or binary values. These variables can be used to add logical constraints to the optimization problem [8, 33]. Obstacle avoidance can be enforced with logical constraints by specifying that the vehicle must be either above, below, left or right of an obstacle [29, 27, 26]. To improve the computational tractability of this approach, the MILP trajectory optimization is applied within a receding horizon control framework [29].

In general, receding horizon control (also called Model Predictive Control) designs an input trajectory that optimizes the plant's output over a period of time, called the *planning horizon*. The input trajectory is implemented over the shorter *execution horizon*, and the optimization is performed again starting from the state that is reached. This re-planning incorporates feedback to account for disturbances and plant modeling errors. In this problem setting, computation can be saved by applying MILP in a receding horizon framework to design a succession of short trajectories instead of one long trajectory, since the computation required to solve a MILP problem grows nonlinearly with its size.

One approach to ensuring that the successively planned short trajectories actually reach the goal is to minimize some estimate of the cost to go from each plan's end, or terminal point, to the goal. If this terminal penalty exactly evaluated the cost-to-go, then the receding horizon solution would be globally optimal. However, it is not obvious how to find an accurate estimate of the cost-to-go from an intermediate trajectory segment's terminal point without optimizing a detailed trajectory all the way to the goal, losing the benefits of receding horizon control. The approach suggested in [29] uses an estimate of the distance-to-go from the plan's endpoint to the goal that was not cognizant of obstacles in this interval, and led to the aircraft becoming trapped behind obstacles. Control Lyapunov Functions have been used successfully as terminal penalties in other problem settings [11], but these are also incompatible with the presence of obstacles in the environment.

1.1.2 Outline of Trajectory Design Approach

Chapter 2 of this thesis presents a receding horizon trajectory designer which avoids entrapment. It applies MILP to optimize the trajectory, and receding horizon control to rationally reduce the size of the decision space. A novel terminal penalty is presented which resolves many of the difficulties associated with previous terminal penalties for trajectory design in the presence of obstacles. This cost-to-go estimate is based on a path to the goal that avoids obstacles, but is made up of straight line segments. This estimate takes advantage of the fact that long range trajectories tend to resemble straight lines that connect the UAVs' starting position, the vertices of obstacle polygons, and the waypoints. The resulting trajectories are shown to be near-optimal, to require significantly less computational effort to design, and to avoid entrapment by improving the degree to which the terminal penalty reflects a flyable path to the goal. However, this straight line path contains heading discontinuities where straight line segments join, so the vehicle would not be able to follow this path exactly. Furthermore, it is possible that no dynamically feasible path exists around the straight line path. This can drive the system unstable by leading the vehicle down a path to the goal that is dynamically infeasible.

Chapter 3 presents a modified receding horizon controller and proves its stability. This modification ensures that cost-to-go estimates are evaluated only along kinodynamically feasible paths. This is guaranteed by adding appropriate constraints on admissible terminal states. This chapter applies the receding horizon stability analysis methods presented in Ref. [1] and generalized in Ref. [7]. This trajectory designer is demonstrated to be capable of planning trajectories in highly constrained environments with a moderate increase in computation time over the unstable receding horizon controller.

The optimization problems presented in this thesis are formulated as MILPs. MILP is an ideal optimization framework for the UAV coordination and control problem, because it integrates the discrete and continuous decisions required for this application. The MILP problems are solved using AMPL [9] and CPLEX [10]. AMPL is a powerful language for expressing the general form of the constraints and cost function of an optimization program. MATLAB is used to provide the parameters of a problem instance to AMPL, and to invoke CPLEX to solve it. Applying commercial software to perform the optimization allows concentration on the formulation, rather than the search procedure to solve it, and allows advances in solving MILPs to be applied as they become available.

1.2 The Allocation Problem

Before trajectories can be designed for the UAVs in the fleet, the control architecture solves an allocation problem to determine a sequence of waypoints for each vehicle to visit. An example of a fleet coordination scenario is shown in Fig. 1-1. In the simplest form of the waypoint allocation problem, every waypoint must be visited while avoiding obstacles. Additional constraints can be added to this problem to capture different types of waypoints representing sites at which to collect sensor information, high value targets which must be destroyed, or anti-aircraft defenses whose destruction might increase the probability of mission success. Only a subset of the fleet might be capable of visiting each waypoint. Timing constraints can be added to enforce simultaneous, delayed or ordered arrival at waypoints. Designing a detailed overall coordination plan for this mission can be viewed as two coupled decisions. Tasks that achieve the fleet goals must be assigned to each team member, and a path must be

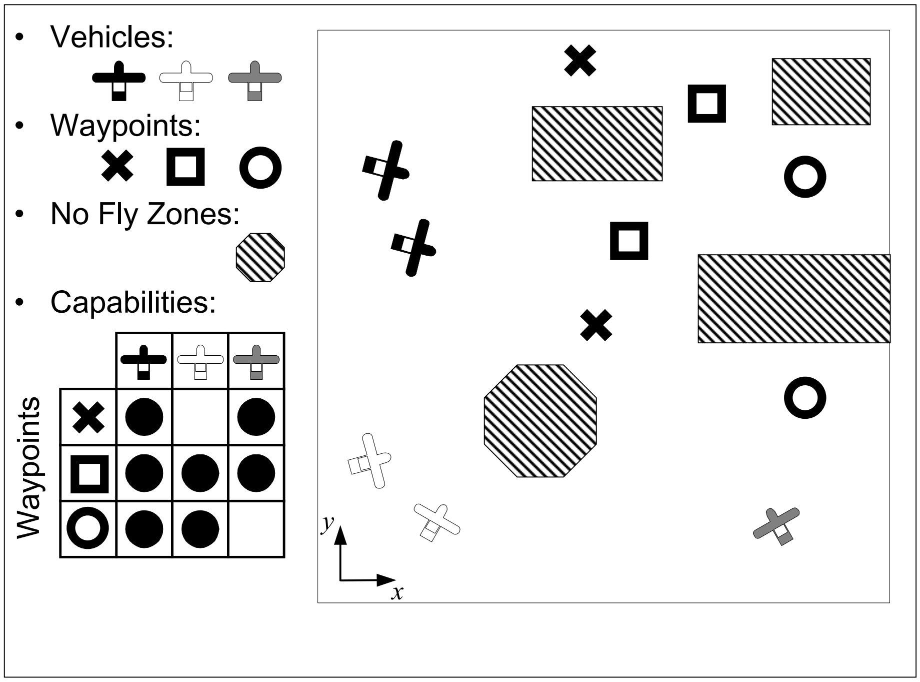


Fig. 1-1: Schematic of a typical mission scenario for a UAV fleet with numerous waypoints, No Fly Zones and capabilities

designed for each team member that achieves their tasks while adhering to spatial constraints, timing constraints, and the dynamic capabilities of the aircraft. The coordination plan is designed to minimize some mission cost, such as completion time or probability of failure.

Consideration of each of the fleet coordination decisions in isolation shows that they are computationally demanding tasks. For even moderately sized problems, the number of combinations of task allocations and waypoint orderings that must be considered for the team formation and task assignment decisions is very large. For the relatively simple coordination problem shown in Fig. 1-1, there are 1296 feasible allocations, and even more possible ordered arrival permutations. Coupling exists between the cost of visiting a particular waypoint and the other waypoints visited by the same vehicle, since the length of the trajectory between them depends on their order. There is also considerable uncertainty in this problem; there is a probability that a UAV will be destroyed by anti-aircraft defenses during its mission, and the waypoint location information may be uncertain and incomplete. The problem of planning kinematically and dynamically constrained optimal paths, even for one aircraft, is also a very high dimension nonlinear optimization problem due to its size and non-convexity, as described in Section 1.1.1. Furthermore, each UAV trajectory is coupled to all other UAV trajectories by collision avoidance constraints.

As difficult as each of these decisions is in isolation, they are in fact strongly coupled to each other. The optimality of the overall coordination plan can be strongly limited by incompatible team partitioning or unsuitable task allocation. However, the cost to be minimized by these decisions is a function of the resulting detailed trajectories. While it is not clear how to partition teams and allocated tasks without full cost information, evaluating the cost of all options would require designing detailed trajectories for an exponential number of possible choices.

1.2.1 Summary Previous Work on the Allocation Problem

This coupling has been handled in one approach [24] by forming a large optimization problem that simultaneously assigns the tasks to vehicles and plans corresponding detailed trajectories. This method is computationally intensive, but it is guaranteed to find the globally-optimal solution to the problem and can be used as a benchmark against which approximate techniques can be compared.

Recent research has examined several aspects of the UAV allocation problem. Tabu search has been successfully applied to find near optimal coordination plans for many UAVs and many waypoints which minimize total flight time [16] and the expectation of waypoint visitation value [12]. These approaches include fixed time windows for visiting individual waypoints, but do not include constraints on relative timing of waypoint visits and do not capture the ability of one UAV to decrease risk to another UAV by destroying a threatening SAM site. A stochastic formulation of allocating weapons to targets has also been studied [22]. In this problem formulation, a set of weapons can be fired at a set of available targets, but some targets may be discovered in the future. To maximize the expectation of destroyed target value over the entire engagement, a stochastic integer program is solved to balance the value of firing weapons at the detected targets against the value of holding weapons to fire at undetected targets. This formulation does not involve timing constraints, and each weapon may be fired against only one target.

Recent research that has focused on the structure of UAV coordination systems gives special attention to the Low Cost Autonomous Attack System (LOCAAS). Researchers have suggested a hierarchical decomposition of the problem into levels that include team forming, intra-team task allocation, and individual vehicle control [5]. Recent research has proposed methods for decision making at each of these levels [4]. Minimum spanning trees are found to group together the tasks for each team. The intra-team assignment is then performed using an iterative network flow method. At each iteration, this method temporarily assigns all remaining tasks to vehicles, and fixes the assignment that would be performed first in time. This is repeated until all the tasks are assigned. An approach to the allocation problem has also been suggested that involves a network minimum cost flow formulation [31]. The formulations can be solved rapidly, and model the value of searching for additional targets.

There are disadvantages associated with both the iterative and the minimum cost flow formulations. The iterative network formulation has problems with robustness, feasibility and occasional poor performance. These are related to the inclusion of fixed time windows for certain tasks to be performed. The iterative and minimum cost flow approaches cannot incorporate these constraints directly, and incomplete approaches to adding these constraints are used. Furthermore, the minimum cost flow formulation requires penalties for modifying the allocation so that it does not make frequent reassignments as the mission is performed. This formulation permits only one task to be assigned to each vehicle at a time, resulting in suboptimal results. Market-based allocation methods have been considered for the LOCAAS problem [30] and for the general UAV allocation problem [32]. The types of coupling that these problem formulations address is relatively simple; these control systems realize benefits through de-conflicting UAV missions and information sharing, and are not capable of using more subtle cooperation between UAVs such as opening corridors through anti-aircraft defenses.

Approaches to the allocation problem which emphasize timing constraints have also been proposed [21, 18, 28]. In this approach, 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. This is performed through the use of coordination functions; each vehicle finds its own minimum arrival time as a function of radar exposure, and communicates their coordination function to the rest of the fleet. Each member of the fleet then solves the same optimization problem to determine the arrival time which minimizes radar exposure and allows all members to arrive simultaneously.

Research has also applied rollout algorithms to UAV control problems [34]. These algorithms are an approach to solving stochastic scheduling problems [2]. Rollout algorithms repeatedly optimize the next scheduling decision to be made. This scheduling decision is selected to minimize the expectation of the sum its cost and the cost-to-go of applying a base scheduling policy from the resulting state to completion. Since the problem is stochastic, some form of simulation is performed to find the expectation of cost, given the first scheduling decision. The UAV control problem is represented in this framework by the aggregation of finite state automata representing aircraft and targets, and a greedy heuristic is used as the base policy. It has been reported [34] that the rollout algorithm is able to learn strategies such as opening attack corridors to decrease attrition.

1.2.2 Outline of Allocation Problem Approach

Chapter 4 of this thesis presents an approach to the combined resource allocation and trajectory optimization aspects of the fleet coordination problem. This approach calculates and communicates the key information that couples the two problems. This algorithm estimates the cost of various trajectory options using distributed platforms and then solves a centralized assignment problem to minimize the mission completion time. It performs this estimation by using the same straight line path approximation examined in Chapter 2 for evaluating the receding horizon controller's terminal penalty. The allocation problem is solved as a MILP, and can include sophisticated timing constraints such as "Waypoint A must be visited t minutes before Waypoint B". This approach also permits the cost estimation step and detailed trajectory planning for this assignment to be distributed between parallel processing platforms for faster solution.

Chapter 5 considers a stochastic MILP formulation of the allocation problem, which maximizes the expectation of the mission's score. This formulation addresses one of the most important forms of coupling in the allocation 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. [12] assumes a fixed risk for visiting each of the waypoints, the ability to reduce this threat is not addressed directly by any approach in the allocation literature. The formulation presented in Chapter 5 associates not only a score but also a threat with each waypoint. A waypoint's threat captures the probability that an anti-aircraft defense at that waypoint destroys a nearby UAV during its mission. A waypoint poses no threat if the waypoint does not represent an anti-aircraft defense, or if a UAV has already destroyed it. The solution 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 2

Receding Horizon Control Trajectory Design

2.1 Introduction

This chapters presents an approach to minimum time trajectory optimization for autonomous fixed-wing aircraft performing large scale maneuvers. These trajectories are essentially planar, and are constrained by no-fly zones and the vehicle's maximum speed and turning rate. MILP is used for the optimization, and is well suited to trajectory optimization because it can incorporate logical constraints, such as no-fly zone avoidance, and continuous constraints, such as aircraft dynamics. MILP is applied over a receding planning horizon to reduce the computational effort of the planner and to incorporate feedback. In this approach, MILP is used to plan short trajectories that extend towards the goal, but do not necessarily reach it. The cost function accounts for decisions beyond the planning horizon by estimating the time to reach the goal from the plan's end point. This time is estimated by searching a graph representation of the environment. This approach is shown to avoid entrapment behind obstacles, to yield near-optimal performance when comparison with the minimum arrival time found using a fixed horizon controller is possible, to work on a large trajectory optimization problem that is intractable for the fixed horizon controller, and to plan trajectories that can be followed by vehicles in a hardware testbed.

This chapter will first present a fixed horizon version of the trajectory planner and a receding horizon controller with a simple terminal penalty for comparison. The control architecture of the improved trajectory planner is presented, including the cost map preparation algorithm and the constraints required to evaluate the new terminal penalty. Finally, simulation and hardware testing results are shown.

2.2 Fixed Horizon Minimum Time Controller

A minimum arrival time controller using MILP over a fixed planning horizon was presented in Ref. [26]. It designs a series of control inputs $\{\mathbf{u}(i) \in \mathcal{R}^2 : i = 0, 1, \ldots, N-1\}$, that give the trajectory $\{\mathbf{x}(i) \in \mathcal{R}^2 : i = 1, 2, \ldots, N\}$. Constraints are added to specify that one of the N trajectory points $\mathbf{x}(i) = [x_{\mathrm{k+i,1}} \;\; x_{\mathrm{k+i,2}}]^T$ must equal the goal $\mathbf{x}_{\mathrm{goal}}$. The optimization minimizes the time along this trajectory at which the goal is reached, using N binary decision variables $\mathbf{b}_{\mathrm{goal}} \in \{0, 1\}$ as

$$\min_{\mathbf{u}(\cdot)} \;\; \phi_1(\mathbf{b}_{\mathrm{goal}}, \mathbf{t}) = \sum_{i=1}^{N} b_{\mathrm{goal,i}} t_{\mathrm{i}} \tag{2.1}$$

subject to

$$\begin{aligned}
x_{\mathrm{k+i,1}} - x_{\mathrm{goal,1}} &\leq \;\; M(1 - b_{\mathrm{goal,i}}) \\
x_{\mathrm{k+i,1}} - x_{\mathrm{goal,1}} &\geq -M(1 - b_{\mathrm{goal,i}}) \\
x_{\mathrm{k+i,2}} - x_{\mathrm{goal,2}} &\leq \;\; M(1 - b_{\mathrm{goal,i}}) \\
x_{\mathrm{k+i,2}} - x_{\mathrm{goal,2}} &\geq -M(1 - b_{\mathrm{goal,i}}) \qquad (2.2) \\
\sum_{i=1}^{N} b_{\mathrm{goal,i}} &= 1 \qquad (2.3)
\end{aligned}$$

where M is a large positive number, and t_{i} is the time at which the trajectory point $\mathbf{x}(i)$ is reached. When the binary variable $b_{\mathrm{goal,i}}$ is 0, it relaxes the arrival constraint in Eqn. 2.2. Eqn. 2.3 ensures that the arrival constraint is enforced once.

To include collision avoidance in the optimization, constraints are added to ensure that none of the N trajectory points penetrate any obstacles. Rectangular obstacles are used in this formulation, and are described by their lower left corner $[u_{\mathrm{low}}\ v_{\mathrm{low}}]^T$ and upper right corner $[u_{\mathrm{high}}\ v_{\mathrm{high}}]^T$. To avoid collisions, the following constraints must be satisfied by each trajectory point

$$\begin{aligned} x_{k+i,1} &\leq u_{\mathrm{low}} + M\ b_{\mathrm{obst},1} \\ x_{k+i,1} &\geq u_{\mathrm{high}} - M\ b_{\mathrm{obst},2} \\ x_{k+i,2} &\leq v_{\mathrm{low}} + M\ b_{\mathrm{obst},3} \\ x_{k+i,2} &\geq v_{\mathrm{high}} - M\ b_{\mathrm{obst},4} \qquad (2.4) \\ \sum_{\mathrm{j}=1}^{4} b_{\mathrm{obst,j}} &\leq 3 \qquad (2.5) \end{aligned}$$

The j^{th} constraint is relaxed if $b_{\mathrm{obst,j}} = 1$, and enforced if $b_{\mathrm{obst,j}} = 0$. Eqn. 2.5 ensures that at least one constraint in Eqn. 2.4 is active for the trajectory point. These constraints are applied to all trajectory points $\{\mathbf{x}(i) : i = 1, 2, \ldots, N\}$. Note that the obstacle avoidance constraints are not applied between the trajectory points for this discrete-time system, so small incursions into obstacles are possible. As a result, the obstacle regions in the optimization must be slightly larger that the real obstacles to allow for this margin.

The trajectory is also constrained by discretized dynamics that model a fixed-wing aircraft as a point of mass m [26]

$$\begin{bmatrix} \dot{x}_{\mathrm{i}+1,1} \\ \dot{x}_{\mathrm{i}+1,2} \\ \ddot{x}_{\mathrm{i}+1,1} \\ \ddot{x}_{\mathrm{i}+1,2} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{\mathrm{i}+1,1} \\ x_{\mathrm{i}+1,2} \\ \dot{x}_{\mathrm{i}+1,1} \\ \dot{x}_{\mathrm{i}+1,2} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{bmatrix} \begin{bmatrix} u_{\mathrm{i},1} \\ u_{\mathrm{i},2} \end{bmatrix} \qquad (2.6)$$

The model also includes a limited speed and turning rate. The latter is represented by a limit on the magnitude of the turning force $\mathbf{u}(i)$ that can be applied

$$L_2(\dot{\mathbf{x}}(i)) \leq v_{\mathrm{max}} \qquad (2.7)$$

$$L_2(\mathbf{u}(i)) \;\leq\; u_{\max} \tag{2.8}$$

The constraints of Eqns. 2.7 and 2.8 make use of a linear approximation $L_2(\mathbf{r})$ of the 2-norm of a vector $\mathbf{r} = (r_1, r_2)$

$$\forall\; [p_1\; p_2]^T \;\in P \;:\; L_2(\mathbf{r}) \geq r_1 p_1 + r_2 p_2, \tag{2.9}$$

where P is a finite set of unit vectors whose directions are distributed from $0° - 360°$. The projection of $\mathbf{r}$ onto these unit vectors gives the length of the component of $\mathbf{r}$ in the direction of each unit vector. When a sufficient number of unit vectors is used in this test, the resulting maximum projection is close to the length of $\mathbf{r}$. The set of unit vectors P is provided to the MILP problem as a parameter.

Note that in this implementation of the problem, there is only an upper bound on the speed. It is feasible for the speed to fall below $v_{\max}$, allowing tighter turns. However, for the minimum time solution, it is favorable to remain at maximum speed [26].

This formulation finds the minimum arrival time trajectory. Experience has shown that the computational effort required to solve this optimization problem can grow quickly and unevenly with the product of the length of the trajectory to be planned and the number of obstacles to be avoided [29, 26]. However, as discussed in the following sections, a receding horizon approach can be used to design large-scale trajectories.

2.3 Simple Terminal Cost Formulation

In order to reduce the computational effort required and incorporate feedback, MILP has been applied within a receding horizon framework. To enable a more direct comparison of the effects of the terminal penalty, the following provides a brief outline of the receding horizon approach suggested in Ref. [29]. The MILP trajectory optimization is repeatedly applied over a moving time-window of length N. The result is a series of control inputs $\{\mathbf{u}(i) \in \mathbf{R}^2 : i = 0, 1, \ldots, N-1\}$, that give the trajectory $\{\mathbf{x}(i) \in \mathbf{R}^2 : i = 1, 2, \ldots, N\}$. The first part of this input trajectory, of length $N_e \leq N$, is executed before a new trajectory is planned. The cost function of this optimization is the terminal penalty $\phi_2(\mathbf{x}(N))$, which finds the 1-norm of the distance between the trajectory's end point and the goal. The formulation is piecewise-linear and can be included in a MILP using slack variables as

$$\min_{\mathbf{u}(\cdot)} \ \phi_2(\mathbf{x}(N)) = L_1(\mathbf{x}_{\text{goal}} - \mathbf{x}(N)) \tag{2.10}$$

where $L_1(\mathbf{r})$ evaluates the 1-norm of $\mathbf{r}$ as the sum of the absolute values of the components of $\mathbf{r}$. Slack variables s_u and s_v are used in the piecewise linear relationships

$$\begin{aligned} L_1(\mathbf{r}) &= s_u + s_v \\ s_u &\geq u \\ s_u &\geq -u \\ s_v &\geq v \\ s_v &\geq -v \end{aligned} \tag{2.11}$$

Obstacle avoidance and dynamics constraints are also added. This formulation is equivalent to the fixed horizon controller when the horizon length is just long enough to reach the goal. However, when the horizon length does not reach the goal, the optimization minimizes the approximate distance between the trajectory's terminal point and the goal.

This choice of terminal penalty can prevent the aircraft from reaching the goal when the approximation does not reflect the length of a *flyable path*. This occurs if the line connecting $\mathbf{x}(N)$ and the goal penetrates obstacles. This problem is especially apparent when the path encounters a concave obstacle, as shown in Fig. 2-1. When the first trajectory segment is designed, the terminal point that minimizes the 1-norm distance to the goal is within the concavity behind the obstacle, so the controller plans a trajectory into the concavity. Because the path out of the concavity would require a temporary increase in the 1-norm distance to the goal, the aircraft becomes trapped behind the obstacle. This is comparable to the entrapment in local minima that is possible using potential field methods.

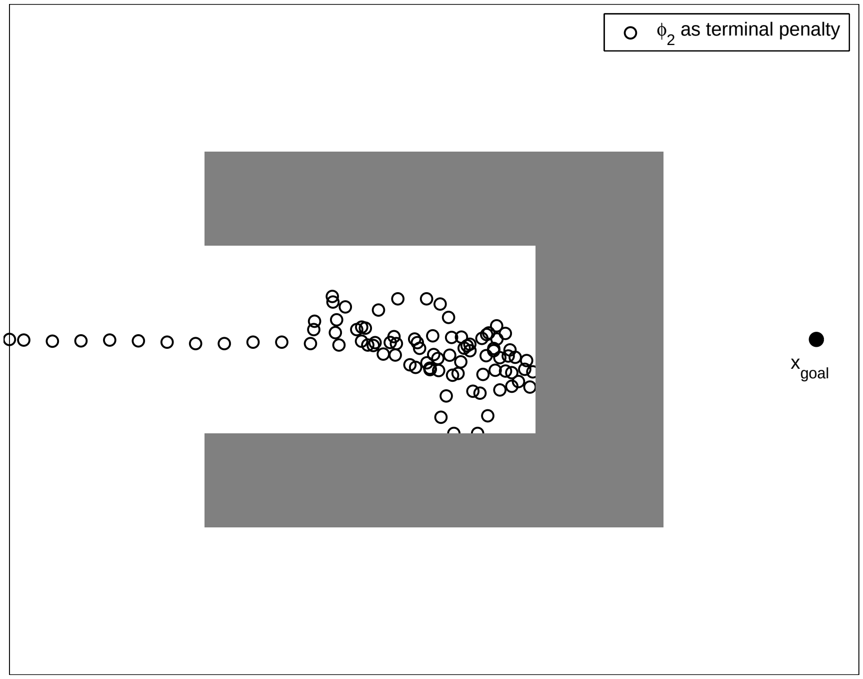


Fig. 2-1: Trajectory Planned using 1-Norm as Terminal Penalty. Starting point at left, goal at right. Circles show trajectory points. Receding horizon controller using simple terminal penalty ϕ_2 and $N = 12$ becomes entrapped and fails to reach the goal.

2.4 Improved Receding Horizon Control Strategy

This section presents a novel method for approximating the time-to-go along a path to the goal which avoids obstacles in order to avoid entrapment. This terminal penalty is implemented in a MILP program, using only linear and binary variables.

2.4.1 Control Architecture

The control strategy is comprised of a cost estimation phase and a trajectory design phase. The cost estimation phase computes a compact "cost map" of the approximate minimum distance to go from a limited set of points to the goal. The cost estimation phase is performed once for a given obstacle field and position of the goal, and would be repeated if the environment changes.

The trajectory designer uses this cost map information to evaluate the terminal penalty

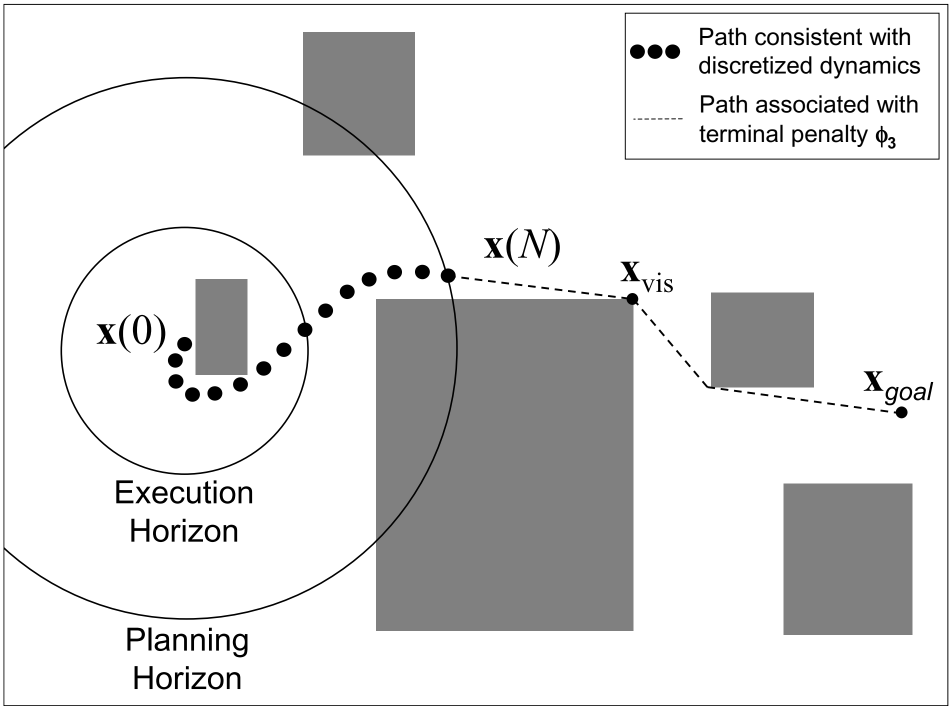

●●● Path consistent with discretized dynamics
Path associated with terminal penalty ϕ_3
$\mathbf{x}(N)$
$\mathbf{x}_{\text{vis}}$
$\mathbf{x}(0)$
$\mathbf{x}_{goal}$
Execution Horizon
Planning Horizon

Fig. 2-2: Resolution Levels of the Planning Algorithm

of the receding horizon optimization. This division of computation between the cost estimation and trajectory design phases enables the trajectory optimization to use only linear relationships. This approach avoids the difficulties associated with nonlinear programming, such as choosing a suitable initial guess for the optimization.

An example of a result that would be expected from the trajectory design phase is shown schematically in Fig. 2-2. In this phase, a trajectory consistent with the discretized aircraft dynamics is designed from $\mathbf{x}(0)$ over a fine resolution planning horizon of N steps. The trajectory is optimized using MILP to minimize the terminal penalty. This cost estimates the distance to the goal from this point as the distance from $\mathbf{x}(N)$ to a visible point $\mathbf{x}_{\text{vis}}$, whose cost-to-go was estimated in the previous phase, plus the cost-to-go estimate c_{vis} for $\mathbf{x}_{\text{vis}}$. As described in Section 2.4.2, c_{vis} is estimated using a coarser model of the aircraft dynamics that can be evaluated very quickly. Only the first N_e steps are executed before a new plan is formed starting from the state reached the end of the execution horizon.

The use of two sets of path constraints with different levels of resolution exploits the trajectory planning problem's structure. On a long time-scale, a successful controller need only decide which combination of obstacle gaps to pass through in order to take the shortest dynamically feasible path. However, on a short time-scale, a successful controller must plan the dynamically feasible time-optimal route around the nearby obstacles to pass through the chosen gaps. The different resolution levels of the receding horizon controller described above allow it to make decisions on these two levels, without performing additional computation to "over plan" the trajectory segment to an unnecessary level of detail.

The cost estimation is performed in MATLAB. It produces a data file containing the cost map in the AMPL [9] language, and an AMPL model file specifies the form of the cost function and constraints. The CPLEX [10] optimization program is used to solve the MILP problem and outputs the resulting input and position trajectory. MATLAB is used to simulate the execution of this trajectory up to $\mathbf{x}(N_e)$, which leads to a new trajectory optimization problem with an updated starting point.

2.4.2 Computation of Cost Map

The shortest path around a set of polygonal obstacles to a goal, without regard for dynamics, is a series of joined line segments that connect the starting point, possibly obstacle vertices, and the goal. To find this path, a visibility graph can be formed whose nodes represent these points. Edges are added between pairs of nodes if the points they represent can be connected by a line that does not penetrate any obstacles. The visibility graph is searched using Dijkstra's Single Source Shortest Path Algorithm [6], starting at the goal, to find the shortest path from the each node of the graph to the goal, and the corresponding distances.

Dijkstra's Algorithm takes advantage of the fact that if the shortest path from $\mathbf{x}_\mathrm{i}$ to the goal passes through $\mathbf{x}_\mathrm{j}$, then the portion of this path from $\mathbf{x}_\mathrm{j}$ to the goal is also $\mathbf{x}_\mathrm{j}$'s shortest path. The algorithm maintains two sets of nodes: $\mathcal{N}$, whose path to the goal has been fixed, and $\bar{\mathcal{N}}$, whose path to the goal could still be improved. $\mathcal{N}$ is initially empty, and $\bar{\mathcal{N}}$ initially

```
algorithm $(\mathbf{c}, \mathbf{s}) = \text{Dijkstra}(\mathbf{x}_{\text{goal}}, \mathcal{X}_{\text{obst}}, \mathbf{x}(0), \mathbf{d})$
    Place $\mathbf{x}_{\text{goal}}$, all obstacle vertices, and $\mathbf{x}(0)$ in $\bar{\mathcal{N}}$;
    $c_1 := 0, s_1 := 1$;  \\ Node 1 is goal
    Set all other costs in $\mathbf{c}$ to $\infty$;
    while $\bar{\mathcal{N}} \neq \emptyset$ do
        Choose the node $\mathbf{x}_j$ in $\bar{\mathcal{N}}$ with minimum $c_j$;
        Move node $\mathbf{x}_j$ from $\bar{\mathcal{N}}$ to $\mathcal{N}$;
        $\text{Relax}(j, \bar{\mathcal{N}}, \mathbf{c}, \mathbf{s}, \mathbf{d})$;
    end while

procedure $\text{Relax}(j, \bar{\mathcal{N}}, \mathbf{c}, \mathbf{s}, \mathbf{d})$
    for all nodes $\mathbf{x}_i \in \bar{\mathcal{N}}$ that are connected to $\mathbf{x}_j$ do
        if $d_{ij} + c_j < c_i$ then
            $c_i := d_{ij} + c_j$;  \\ Shorten by going through j
            $s_i := j$;
        end if
    end for
```

Algorithm 1: Dijkstra's Algorithm. This algorithm provides the basis for Alg. 2

contains all obstacle vertices, the start node, and the goal node. Since optimal trajectories tend to head towards obstacle vertices, these points are a good choice of points at which to find the cost-to-go. Additional points can be added to $\bar{\mathcal{N}}$.

At each iteration, the algorithm chooses a point of known cost $\mathbf{x}_j$ to move from $\bar{\mathcal{N}}$ to $\mathcal{N}$, effectively fixing its path to the goal. In the procedure Relax, all of the nodes $\mathbf{x}_i$ that are both connected to $\mathbf{x}_j$ and in $\bar{\mathcal{N}}$ are then examined. If the current route from $\mathbf{x}_i$ to the goal is longer than the route from $\mathbf{x}_i$ through $\mathbf{x}_j$ to the goal, then $\mathbf{x}_i$'s current minimum distance c_i is updated to this lower value, and j is recorded as s_i, the successor of node i on the path to the goal.

After the distances are updated for all the connected nodes, the minimum of the distance values corresponding to nodes in $\bar{\mathcal{N}}$ is now known with certainty. The node with this minimum distance is moved to $\mathcal{N}$. This process continues until the shortest path from all nodes to the goal has been found.

In order to illustrate how the resulting cost map accounts for obstacles, their contribution to the cost is isolated in Fig. 2-3. To produce this graph, cost values were found over a fine

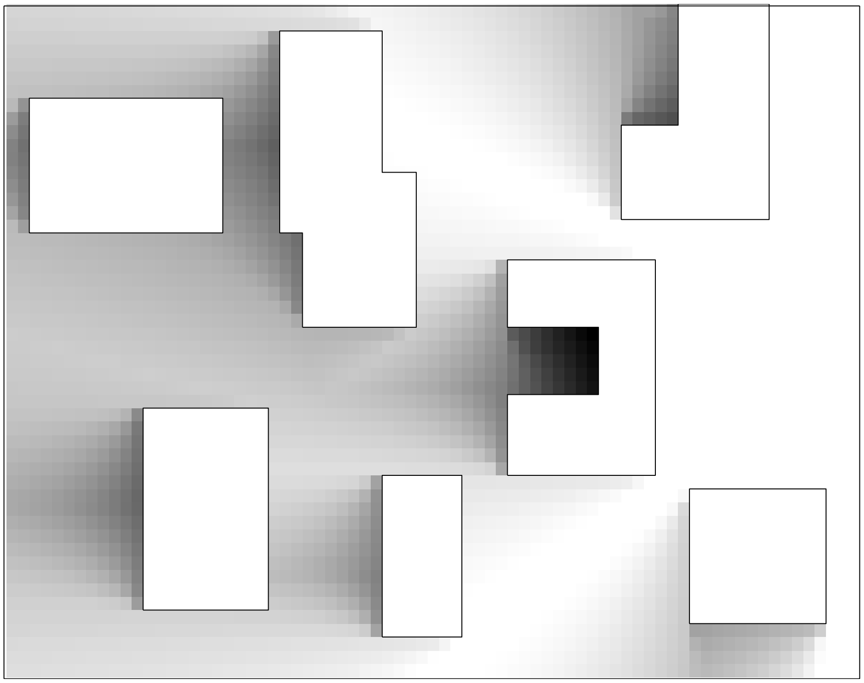


Fig. 2-3: Effect of Obstacles on Cost Values. Difference between actual cost at various points in an obstacle field and cost in same region with no obstacles, larger differences shown with darker shading. This shows the effects of obstacles on cost values. Goal is at center right.

grid of points in two fields of equal size, one with obstacles, and one without[1]. The cost values found in the obstacle-free field were subtracted from the cost values found in the obstacle field to remove the contribution of straight line distance to costs. Areas of larger difference are shown in Fig. 2-3 by darker shading. Note that the cost is increasing into the concave obstacle. This increase is crucial to avoiding the entrapment shown in Fig. 2-1.

The major computational activities in the cost estimation phase are determining whether lines intersect with obstacles, and searching through the visibility graph for shortest paths. Computationally efficient ways of doing each are readily available [6], and the entire cost estimation portion of the computation can be performed in a fraction of the time required to form one plan.

---

[1] Cost values need not be found over such a fine grid to plan trajectories successfully. Since optimal large-scale trajectories tend to connect the starting point, obstacle vertices, and the goal, costs need only be found at these points. Many extra grid points are added here to more clearly demonstrate the trend in cost values.

2.4.3 Modified MILP Problem

The results of the cost estimation phase are provided to the trajectory design phase as pairs of a position where the approximate cost-to-go is known and the cost at that point $(\mathbf{x}_{\text{cost,j}}, c_{\text{j}})$. This new formulation includes a significantly different terminal cost that is a function of $\mathbf{x}(N)$, and $(\mathbf{x}_{\text{vis}}, c_{\text{vis}})$, a pair from the cost estimation phase. The optimization seeks to minimize the distance that must be covered from $\mathbf{x}(N)$ to reach the goal by choosing $\mathbf{x}(N)$ and the pair $(\mathbf{x}_{\text{vis}}, c_{\text{vis}})$ that minimize the distance from $\mathbf{x}(N)$ to $\mathbf{x}_{\text{vis}}$, plus the estimated distance c_{vis} to fly from $\mathbf{x}_{\text{vis}}$ to $\mathbf{x}_{\text{goal}}$.

$$\min_{\mathbf{u}(\cdot)} \ \phi_3(\mathbf{x}(N)) = L_2(\mathbf{x}_{\text{vis}} - \mathbf{x}(N)) + c_{\text{vis}} \tag{2.12}$$

A key element in the algorithm is that the optimization is not free to choose $\mathbf{x}(N)$ and $\mathbf{x}_{\text{vis}}$ independently. Instead, $\mathbf{x}_{\text{vis}}$ is constrained to be visible from $\mathbf{x}(N)$. Note that visibility constraints are, in general, nonlinear because they involve checking whether every point along a line is outside of all obstacles. Because these nonlinear constraints cannot be included in a MILP problem, they are approximated by constraining a discrete set of interpolating points between $\mathbf{x}(N)$ and $\mathbf{x}_{\text{vis}}$ to lie outside of all obstacles. These interpolating points are a portion τ of the distance along the line-of-sight between $\mathbf{x}(N)$ and $\mathbf{x}_{\text{vis}}$

$$\forall\, \tau \in \mathcal{T} : [\mathbf{x}(N) + \tau \cdot (\mathbf{x}_{\text{vis}} - \mathbf{x}(N))] \notin \mathcal{X}_{\text{obst}} \tag{2.13}$$

where $\mathcal{T} \subset [0, 1]$ is a discrete set of interpolation distances and $\mathcal{X}_{\text{obst}}$ is the obstacle space.

The visibility constraint ensures that the length of the line between $\mathbf{x}(N)$ and $\mathbf{x}_{\text{vis}}$ is a good estimate of the length of a path between them which avoids obstacles. The interpolating points are constrained to lie outside obstacles in the same way that the trajectory points are constrained to lie outside obstacles in the previous formulations (see Eqns. 2.4 and 2.5), so it is possible that portions of the line-of-sight between interpolating points penetrate obstacles. However, the number of interpolating points can be chosen as a function of the distance to the goal and the narrowest obstacle dimension to guarantee that the line-of-sight will only be able to "cut corners" of the obstacles. In this case, the extra distance required to fly around the corner is small, and the accuracy of the terminal penalty is not seriously affected.

The values of the position $\mathbf{x}_{\mathrm{vis}}$ and cost c_{vis} are evaluated using the binary variables $\mathbf{b}_{\mathrm{cost}}$ and the n points on the cost map as

$$\mathbf{x}_{\mathrm{vis}} = \sum_{j=1}^{n} b_{\mathrm{cost,j}} \mathbf{x}_{\mathrm{cost,j}} \tag{2.14}$$

$$C_{\mathrm{vis}} = \sum_{j=1}^{n} b_{\mathrm{cost,j}} C_{\mathrm{j}} \tag{2.15}$$

$$\sum_{j=1}^{n} b_{\mathrm{cost,j}} = 1 \tag{2.16}$$

Obstacle avoidance constraints (Eqns. 2.4 and 2.5) are enforced without modification at $\{\mathbf{x}(i) : i = 1, 2, \ldots, N\}$. The dynamics model (Eqn. 2.6), the velocity limit (Eqn. 2.7), and the control force limit (Eqn. 2.8) are also enforced in this formulation. This provides a completely linear receding horizon formulation of the trajectory design problem.

2.5 Results

The following examples demonstrate that the new receding horizon control strategy provides trajectories that are close to time-optimal and avoid entrapment, while maintaining computational tractability.

2.5.1 Avoidance of Entrapment

In order to test the performance of the improved cost penalty around concave obstacles, the improved terminal penalty ϕ_3 was applied to the obstacle field shown in Fig 2-1, and the resulting trajectories are shown in Fig. 2-4. The new cost function captures the difference between the distance to the goal and the length of a path to the goal that avoids obstacles, allowing the receding horizon controller to plan trajectories that reach the goal.

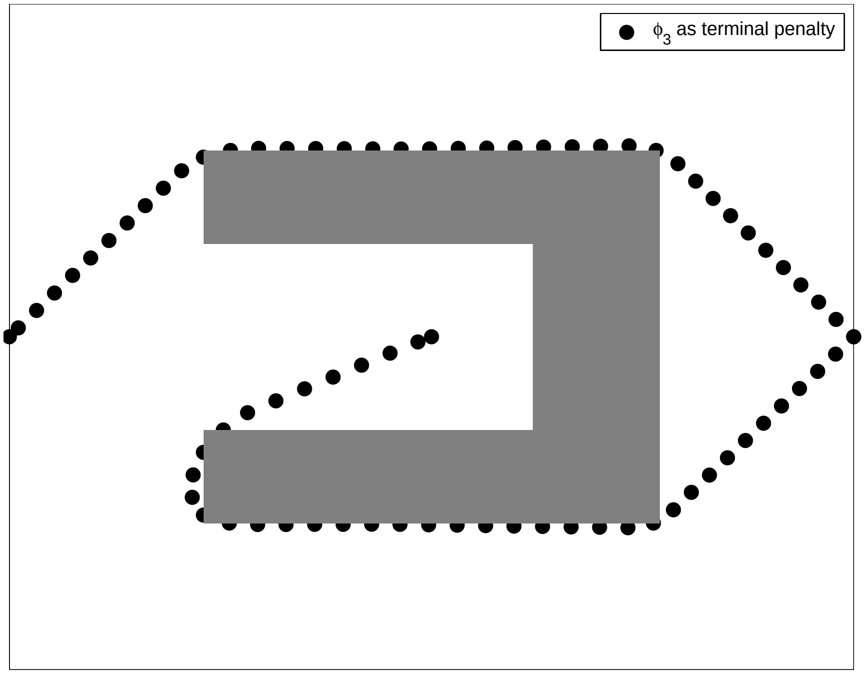


Fig. 2-4: Trajectories designed using receding horizon controller with ϕ_3 terminal penalty avoid entrapment. Trajectories start at left and at center, goal is at right. Circles show trajectory points. $N = 12$.

2.5.2 Performance

The computational effort required by the receding horizon control strategy is significantly less than that of the fixed horizon controller because its planning horizon is much shorter. However, for the same reason, global optimality is not guaranteed. To examine this trade-off, a set of random obstacle fields was created, and a trajectory to the goal was planned using both the receding and fixed horizon controllers. The receding horizon controller was applied several times to each problem, each time with a longer planning horizon. The results are shown in Fig. 2-5. The extra number of time steps in the receding horizon controller's trajectory is plotted as a percentage of the minimum number of steps found using the fixed horizon controller, averaged over several obstacle fields. The plot shows that, on average, the receding horizon controller is within 3% of the optimum for a planning horizon longer than 7 time steps. The average total computation time for the receding horizon controller is also plotted, showing that the increase in computation time is roughly linear with plan

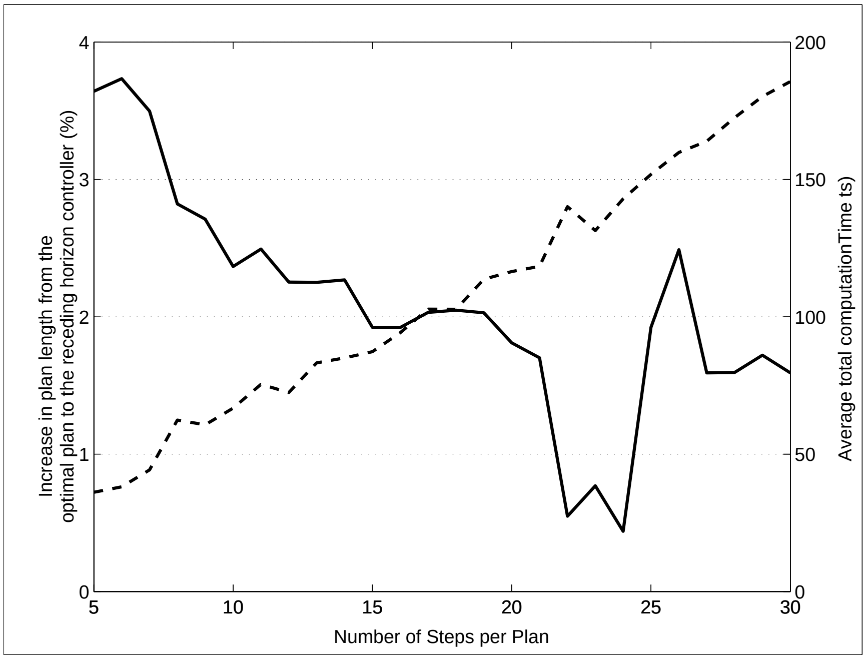


Fig. 2-5: The Effects of Plan Length. Increase in arrival time from optimal to that found by receding horizon controller is plotted with a solid line. Average total computation time is plotted with a dashed line.

length.

2.5.3 Computation Savings

The effects of problem complexity on computation time were also examined by timing the fixed and receding horizon controllers' computation on a 1 GHz PIII computer. The complexity of a MILP problem is related to its number of binary variables. For the fixed horizon trajectory designer, the number of binary variables required for obstacle avoidance dominates the total number of binary variables as the problem size grows. Four binary variables are required for every obstacle at every time step, so the product of the number of time steps required to reach the goal and the number of obstacles was chosen as a metric of complexity. A series of obstacle fields was created with increasing values of this metric, and a trajectory through each obstacle field was planned using both the receding and fixed horizon controllers. Unlike the previous test, the receding horizon controller was applied here with

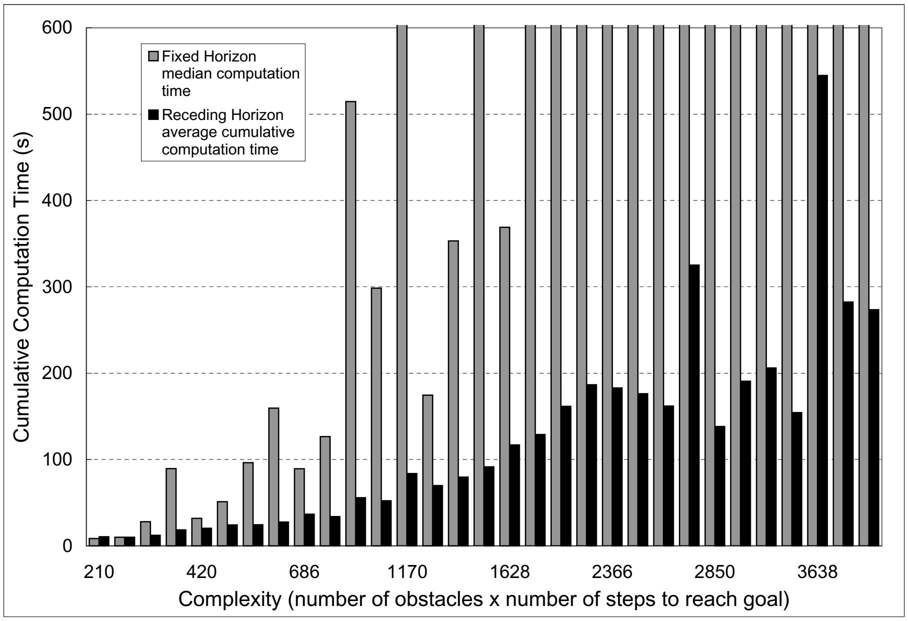


Fig. 2-6: Cumulative Computation Time vs. Complexity

a fixed-length planning horizon. The optimization of each plan was aborted if the optimum was not found in 600 seconds.

The results of this test are shown in Fig. 2-6, which gives the average cumulative computation time for the receding horizon controller, and the median computation time for the fixed horizon controller. The median computation time for the fixed horizon controller was over 600 seconds for all complexity levels over 1628. At several complexity levels for which its median computation time was below 600 seconds, the fixed horizon controller also failed to complete plans. The cumulative time required by the receding horizon controller to design all the trajectory segments to the goal was less than this time limit for every problem. All but the first of its plans can be computed during execution of the previous, so the aircraft can begin moving towards the goal much sooner.

Next, an extremely large problem, with a complexity of 6636, was attempted with the

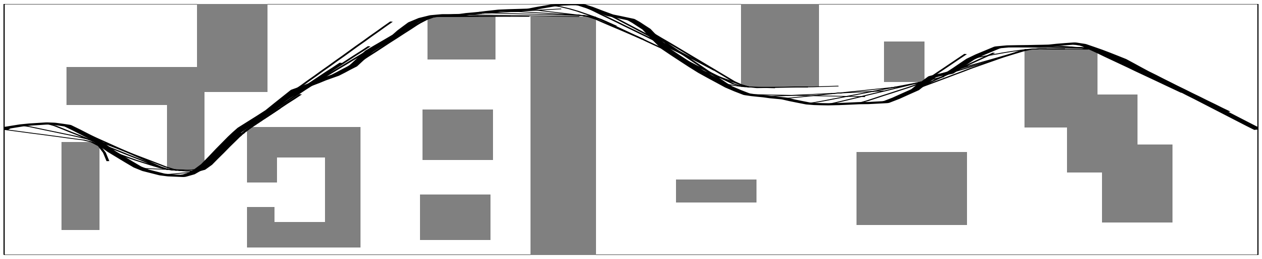


Fig. 2-7: Sample Long Trajectory Designed using Receding Horizon Controller. Executed trajectory (plan plus velocity disturbance) shown with thick line, planned trajectory segments shown with thin lines.

receding horizon controller. It successfully designed a trajectory that reached the goal in 316 time steps, in a cumulative computation time of 313.2 seconds. This controller took 2.97 seconds on average to design one trajectory segment. The fixed horizon controller could not solve this problem in 1200 seconds of computation time. A trajectory through the same obstacle field was also planned in the presence of velocity disturbances, causing the followed trajectory to differ significantly from each of the planned trajectory segments. By designing each trajectory segment from the state that is actually reached, the receding horizon controller compensates for the disturbance. The executed trajectory and planned trajectory segments are shown in Fig. 2-7.

2.5.4 Hardware Testing Results

The receding horizon controller was also used to plan a trajectory for a hardware testbed system. This system was made up of radio-controlled trucks equipped with GPS for state estimation [20]. This trajectory was provided as a reference to a controller on board each truck, and is plotted along with the trajectory that the vehicles actually followed in Fig. 2-8. The truck successfully avoided the obstacles and reached its goal. Due mainly to a time delay between state estimation and steering actuation, the reference trajectory is not followed exactly. However, the reference's selected minimum radius of curvature is clearly larger than the vehicle's turning radius, indicating that the model of the vehicle dynamics restricts the formulation to planning maneuvers that are compatible with the vehicle's turning radius.

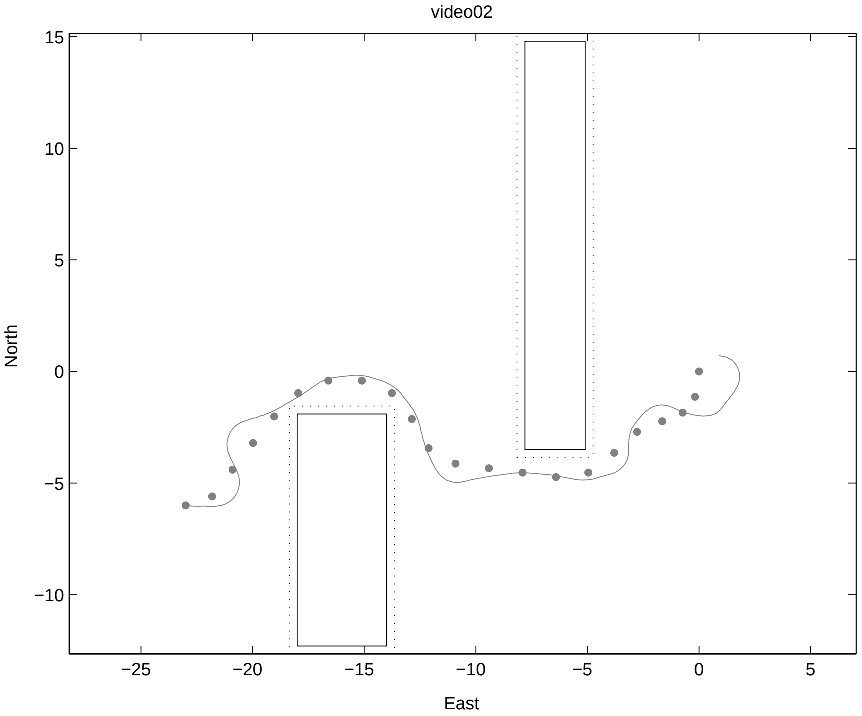


Fig. 2-8: Planned and Executed Trajectories for Single Vehicle. Time steps making up the planned trajectory are shown with circles. Positions recorded during execution are shown with solid line. Vehicle starts at right and finishes at left. Note that due to the discretization of obstacle avoidance, the obstacles were increased to guarantee that the original obstacles were not penetrated between time steps.

Next, trajectories were planned in a combined optimization for two trucks starting at opposite ends of an obstacle field. Inter-vehicle collision avoidance was enforced similarly to obstacle avoidance, but with a prohibited square of size 5m around each vehicle's time-varying position, instead of the fixed obstacle rectangle. The executed trajectories are plotted together with the planning trajectories in Fig. 2-9. This indicates that inter-vehicle collision avoidance can be enforced by a receding horizon planner which simultaneously plans trajectories for both vehicles.

2.5.5 Planning with Incomplete Information

The examples presented so far have assumed that the position of all obstacles was known at the beginning of the trajectory design and execution process. To examine the effects of

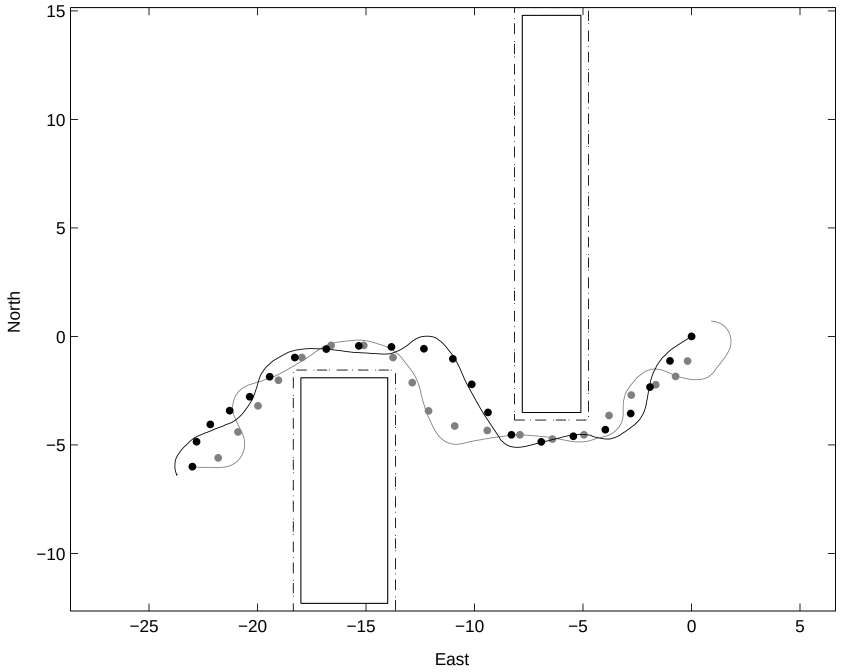


Fig. 2-9: Planned and Executed Trajectories for Two Vehicles. Time steps making up the planned trajectory are shown with circles. Positions recorded during execution are shown with solid line. Grey vehicle starts at left, black vehicle starts at right and vehicles exchange sides. Note inter-vehicle collision avoidance at center.

imperfect obstacle position information, simulations were performed that model a vehicle which begins with no obstacle information, but detects obstacles when they come within some range of it. It is assumed that the obstacles do not move. The detection range was chosen to be longer than the distance that the vehicle could cover over the execution horizon, to avoid collisions.

A simulation was performed in a relatively simple environment, and the vehicle's position and obstacle information is shown in Figs. 2-10 – 2-13. In this simulation, the vehicle takes an inefficient route, but still reaches the goal because its cost map is updated when new information is received. This update is performed rapidly, requiring a fraction of the time required to optimize one trajectory segment.

The long trajectory planning problem of Fig. 2-7 was also attempted with imperfect obstacle information, and the resulting trajectory to the goal is shown in Fig. 2-14. In this

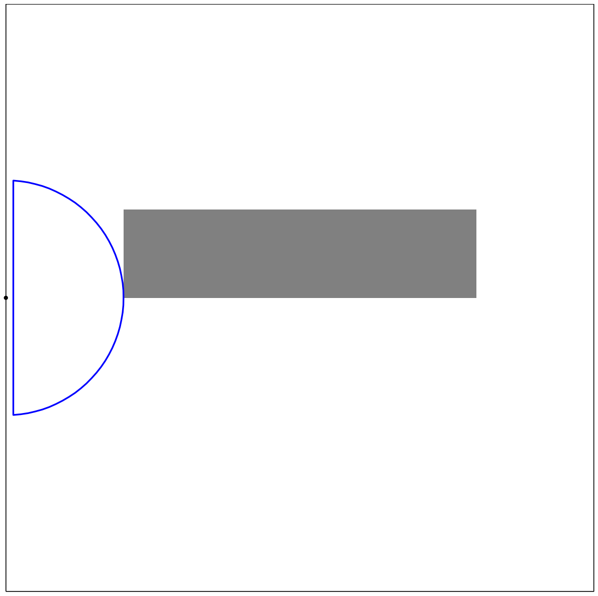


Fig. 2-10: Trajectory Planning with Incomplete Information. At start, location of one obstacle is known. Detection range is shown with circle. Goal is at right.

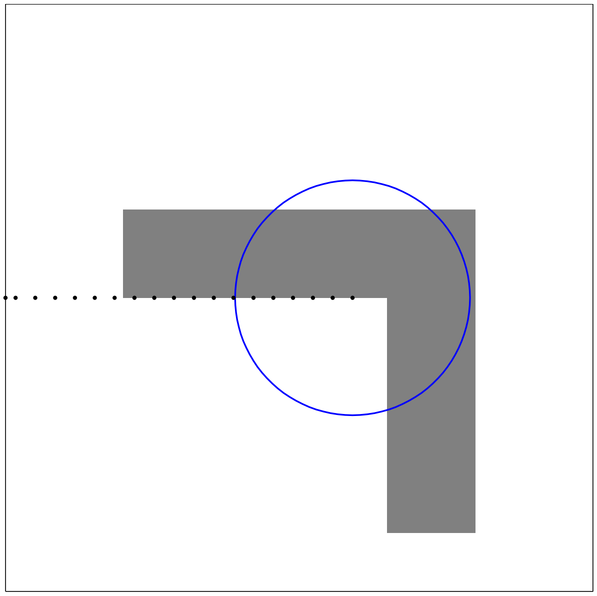


Fig. 2-11: Another obstacle is detected when it enters the detection range, and the cost map is updated rapidly.

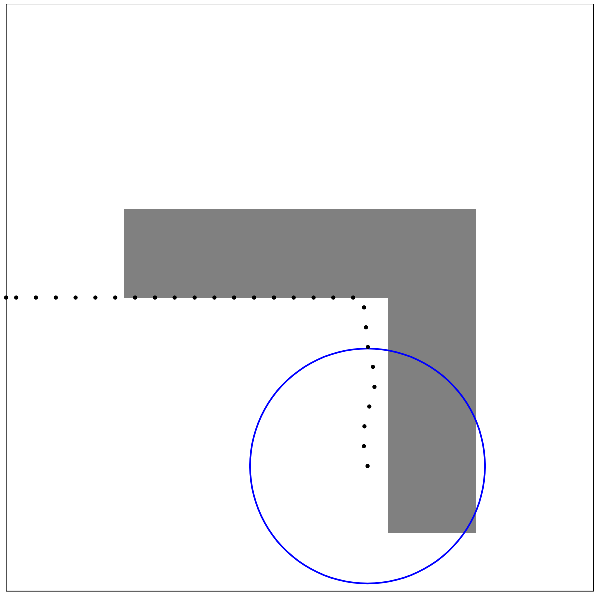


Fig. 2-12: With cost values updated, the lower left corner of the intervening obstacle is selected as $\mathbf{x}_{\text{vis}}$.

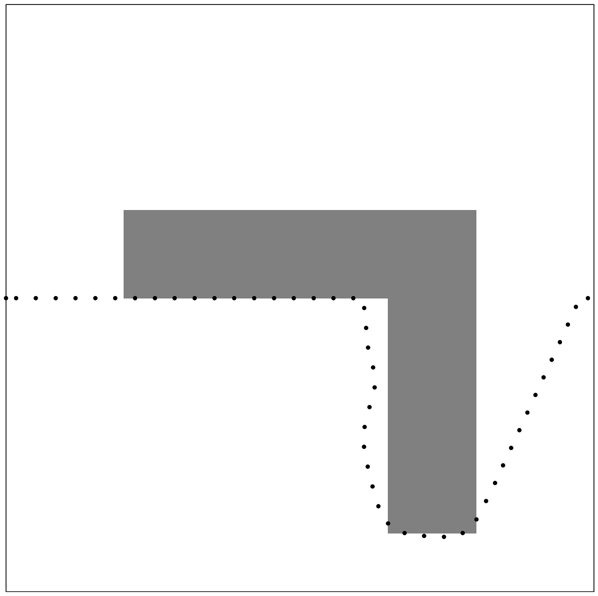


Fig. 2-13: The resulting trajectory to the goal.

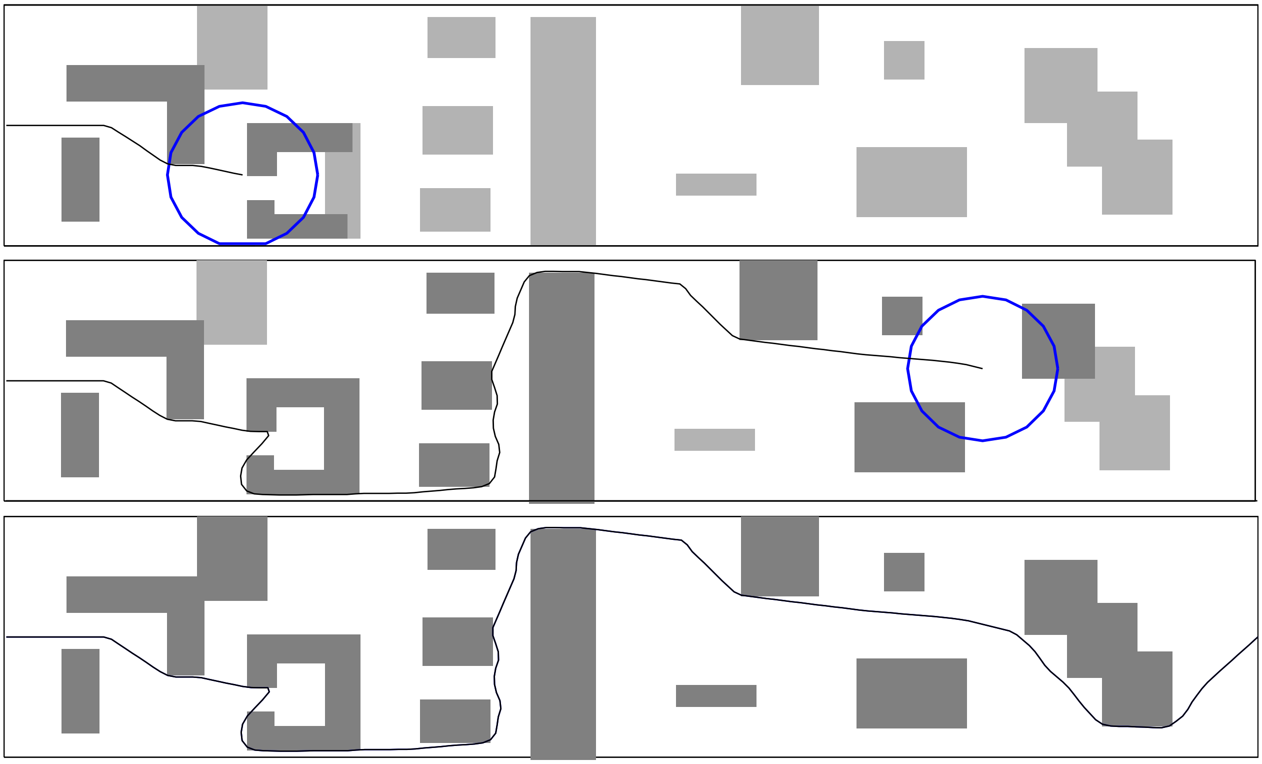


Fig. 2-14: Sample Long Trajectory in Uncertain Environment. The undetected obstacles are shown with lighter shading, with the concave obstacle modeled as a set of overlapping rectangular obstacles. In top figure, the vehicle is about to enter the concavity because it has not yet detected the obstacle forming the concavity's far end. When it does so, the cost map is updated to reflect the fact that points inside the obstacle are further along a flyable path from the goal than points at its opening. In the middle figure, the vehicle takes an inefficient route around obstacles near the goal by attempting to pass below an obstacle, without knowledge of an additional blocking obstacle below it. The final trajectory and all obstacles are shown in the bottom figure.

example, the vehicle temporarily enters the concavity behind an obstacle, but leaves it when it detects that the concavity is closed at its far end.

These examples indicate that the trajectory planner can take a longer route than necessary when it has incomplete obstacle information. However, the cost map is updated to capture new obstacle information, and the vehicle eventually reaches the goal.

2.6 Conclusions

This chapter presents a new algorithm for designing long-range kinodynamically constrained trajectories for fixed-wing UAVs. It is based on MILP optimization within a receding horizon control framework. A novel terminal penalty for the receding horizon optimization is computed by finding a cost map for the environment, and connecting the aircraft trajectory over the planning horizon to the cost map. The resulting MILP problem can be solved with commercially available optimization software. Simulation results show that the receding horizon controller plans trajectories whose arrival times are within 3% of optimal, and that the controller can successfully solve complex trajectory planning problems in practical computation times. This controller avoids avoids entrapment in concavities, even when incomplete obstacle location information is available. The receding horizon trajectory designer has been test on a hardware testbed, and planned trajectories that were compatible with the dynamics of the trucks.

Chapter 3

Stable Receding Horizon Trajectory Design

In general, the cost function of a receding horizon controller's optimization problem estimates the cost-to-go from the selected terminal state to the goal. While the receding horizon controller reoptimizes the trajectory before the system reaches the end of the plan, it is possible for some properties of the path associated with the cost-to-go estimate to appear in the executed trajectory. If the associated path does not avoid obstacles, the vehicle can become entrapped behind concave obstacles [29]. The trajectory designer presented in Chapter 2 prevented entrapment by using the length of a path to the goal made up of straight line segments as its cost-to-go. While this planner provides good results in practice, it is not infallible. The vehicle cannot follow the heading discontinuities where the line segments join, so the path associated with the cost-to-go estimate is not dynamically feasible. Normally, replanning is able to find a dynamically feasible path around the line segment path which the vehicle can follow to the goal. However, the trajectory design problem can become infeasible when the positioning of nearby obstacles leaves no dynamically feasible trajectory from the state that is reached. In this case, the receding horizon controller finds no path for the vehicle to follow, and collision with obstacles is unavoidable without violating the aircraft dynamics.

This chapter modifies the trajectory planner to guarantee that it is stable, and reaches the goal in bounded time. It does so by evaluating the cost-to-go estimate along a kinodynamically feasible path to the goal. This trajectory designer is shown to be computationally tractable, to be capable of designing trajectories in highly constrained environments where the unstable formulation presented in Chapter 2 is incapable of reaching the goal, and to result in minimal increase in path length over the trajectories found by that planner. The relevant receding horizon control stability analysis techniques are summarized in Section 3.1, then a basic stability proof for a modified trajectory designer is given in Section 3.2. A method for computing a suitable cost map is described in Section 3.2.2, and a MILP form of the trajectory designer's optimization problem is shown in Section 3.3. While this optimization problem is involved, it is necessary to completely guarantee the intuitive conditions for stability presented in Section 3.1. Finally, trajectory design examples are shown in Section 3.4.

3.1 Review of Stability Analysis Techniques

In addition to Ref. [1], the survey paper Ref. [7] gives a summary of techniques for analyzing the stability of discrete-time receding horizon controllers. The receding horizon controllers studied in this paper all solve optimization problems with the goal of driving the system to the origin. The optimization problems constrain the system to reach a desired final region at the end of the planning horizon.

Two benefits are gained through this approach: i) computational effort is saved by forming plans that do not necessarily reach the goal, and can therefore be shorter; and ii) the cost of control is reduced by allowing successive optimization solutions to delay reaching the goal until the end of the receding planning horizon. By proving that the controller reaches the origin in bounded time, it is guaranteed that the controller does not delay completion forever, and that the origin itself is eventually reached, rather than just the final region.

The common elements that have been useful for proving the stability of many different receding horizon controllers are a positive definite state and control penalty $\ell(\mathbf{x}(i), \mathbf{u}(i))$, a positive definite terminal penalty $F(\mathbf{x}(N))$, and a terminal constraint set $\mathcal{X}_f$. The optimization problem is

$$\begin{aligned} \min_{\mathbf{u}} V_N(\mathbf{x}) &= \sum_{i=0}^{N-1} \ell(\mathbf{x}(i), \mathbf{u}(i)) + F(\mathbf{x}(N)) \\ \mathbf{x}^+ &\equiv \mathbf{x}(i+1) = f(\mathbf{x}(i), \mathbf{u}(i)) \\ \mathbf{u}(i) &\in \mathcal{U} \\ \mathbf{x}(i) &\in \mathcal{X} \\ \mathbf{x}(N) &\in \mathcal{X}_f \subset \mathcal{X} \end{aligned} \tag{3.1}$$

Where $\mathbf{x}(i+1) = f(\mathbf{x}(i), \mathbf{u}(i))$ corresponds to the system dynamics, $\mathcal{U}$ represents the set of admissible control inputs, $\mathcal{X}$ represents the set of admissible states, and $N \geq 1$. The optimal solution of this program at state $\mathbf{x}$ results in a state trajectory $X^{\mathrm{o}}(\mathbf{x})$ and control trajectory $U^{\mathrm{o}}(\mathbf{x})$ with associated minimum cost $V_N^O(\mathbf{x})$. Let $\mathbf{x}^{\mathrm{o}}(i; \mathbf{x})$ be the i^{th} state reached by applying $U^{\mathrm{o}}(\mathbf{x})$ starting from $\mathbf{x}$, and let $\mathbf{u}^{\mathrm{o}}(i; \mathbf{x})$ be the i^{th} control input applied.

References [1] and [7] assume that the first control input $\mathbf{u}^{\mathrm{o}}(0; \mathbf{x})$ from U^{o} is executed before the optimization is performed again. The notation $\phi^*(\mathbf{x}, \mathbf{u})$ represents the change in some function ϕ between state $\mathbf{x}(i)$ and $\mathbf{x}(i+1)$,

$$\phi^*(\mathbf{x}(i), \mathbf{u}(i)) = \phi(f(\mathbf{x}(i), \mathbf{u}(i))) - \phi(\mathbf{x}(i))$$

The sufficient condition for stability is

$$V_N^{O*} + \ell(\mathbf{x}, \mathbf{u}^{\mathrm{o}}(0; \mathbf{x})) \leq 0 \tag{3.2}$$

This condition states that as the optimization is repeatedly solved over a receding horizon, the optimal value of the cost function must decrease by at least the first state and control penalty $\ell(\mathbf{x}, \mathbf{u}^{\mathrm{o}}(0; \mathbf{x}))$. If this condition is satisfied, then the system reaches the origin eventually. This is because the optimal cost V_N^O is lower-bounded by 0, and decreasing by at least

$\ell(\mathbf{x}, \mathbf{u}^o(0; \mathbf{x}))$. Therefore, V_N^O must converge to some value. When it converges, $V_N^{O*} \to 0$, so $\ell(\mathbf{x}, \mathbf{u}^o(0; \mathbf{x})) \to 0$, which is satisfied only if $\mathbf{x} \to 0$.

Ref. [7] provides four conditions that guarantee that Eqn. 3.2 holds. While Eqn. 3.2 is a powerful general statement, these conditions are easier to evaluate in the context of a particular controller. The conditions in Eqns 3.4 – 3.6 refer to a terminal control law $\mathbf{u} = \kappa_f(\mathbf{x})$, which can be used when the system is within $\mathcal{X}_f$. The conditions are

$$\mathcal{X}_f \subset \mathcal{X},\ \mathcal{X}_f \text{ is closed, } 0 \in \mathcal{X}_f \tag{3.3}$$

$$\kappa_f(\mathbf{x}) \in \mathcal{U}, \forall\ \mathbf{x} \in \mathcal{X}_f \tag{3.4}$$

$$f(\mathbf{x}, \kappa_f(\mathbf{x})) \in \mathcal{X}_f, \forall\ \mathbf{x} \in \mathcal{X}_f \tag{3.5}$$

$$[F^* + \ell](\mathbf{x}, \kappa_f(\mathbf{x})) \leq 0, \forall\ \mathbf{x} \in \mathcal{X}_f \tag{3.6}$$

Eqn. 3.3 specifies that the origin must be in the desired terminal region, and that this region must satisfy the state constraint. Eqn. 3.4 specifies that κ_f must produce admissible control inputs, and Eqn. 3.5 specifies that once the region $\mathcal{X}_f$ is entered, κ_f must keep the system within it. Eqn. 3.6 specifies that the terminal controller must make the terminal penalty decrease by at least the state and control penalty associated with the terminal point $\mathbf{x}$ between $\mathbf{x}$ and $\mathbf{x}^+$. While κ_f could be used within $\mathcal{X}_f$, its only purpose is to assist in the proof, and it is never actually used to control the system.

The connection between these conditions and Eqn. 3.2 will be shown now. Assume that the optimization of Eqn. 3.1 is solved at $\mathbf{x}$, resulting in a trajectory with terminal state $\mathbf{x}_f^o = \mathbf{x}^o(N; \mathbf{x})$ and optimal cost $V_N^O(\mathbf{x})$. If the first resulting control input is applied so that state $\mathbf{x}^+$ is reached, and the conditions of Eqn. 3.3 – 3.6 are met, then a feasible control trajectory of length N can be constructed with $\mathbf{x}^+$ as its starting point using $\kappa_f(\mathbf{x}_f^o)$ as

$$\tilde{U}(\mathbf{x}) = \{\mathbf{u}^o(1; \mathbf{x}), \ldots, \mathbf{u}^o(N-1; \mathbf{x}), \kappa_f(\mathbf{x}_f^o)\}$$

The control trajectory $\tilde{U}(\mathbf{x})$ would result in a terminal state of $f(\mathbf{x}_f^o, \kappa_f(\mathbf{x}_f^o))$. The cost associated with implementing $\tilde{U}(\mathbf{x})$ can be constructed from the previous cost $V_N^O(\mathbf{x})$ by dropping the first state and control penalty $\ell(\mathbf{x}, u^{\mathrm{o}}(0; \mathbf{x}))$ and terminal penalty $F(\mathbf{x}_f^{\mathrm{o}})$, and adding the state and control penalty $\ell(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}}))$ and terminal penalty $F(f(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}})))$ associated with using $\kappa_f(\mathbf{x}_f^{\mathrm{o}})$.

$$\begin{aligned} V_N(\mathbf{x}^+, \tilde{U}(\mathbf{x})) &= V_N^O(\mathbf{x}) - \ell(\mathbf{x}, u^{\mathrm{o}}(0; \mathbf{x})) - F(\mathbf{x}_f^{\mathrm{o}}) + \ell(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}})) + F(f(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}}))) \\ V_N(\mathbf{x}^+, \tilde{U}(\mathbf{x})) - V_N^O(\mathbf{x}) + \ell(\mathbf{x}, u^{\mathrm{o}}(0; \mathbf{x})) &= F(f(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}}))) - F(\mathbf{x}_f^{\mathrm{o}}) + \ell(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}})) \quad (3.7) \end{aligned}$$

By the constraints of Eqns. 3.4 and 3.5, the trajectory associated with $\tilde{U}$ is a feasible solution. Thus, its associated cost $V_N(\mathbf{x}^+, \tilde{U}(\mathbf{x}))$ is an upper bound on the optimal cost $V_N^O(\mathbf{x}^+)$. The following inequality can be constructed with the left-hand side of Eqn. 3.7

$$\begin{aligned} V_N^O(\mathbf{x}^+) - V_N^O(\mathbf{x}) + \ell(\mathbf{x}, \mathbf{u}^{\mathrm{o}}(0; \mathbf{x})) &\leq V_N(\mathbf{x}^+, \tilde{\mathbf{u}}(\mathbf{x})) - V_N^O(\mathbf{x}) + \ell(\mathbf{x}, u^{\mathrm{o}}(0; \mathbf{x})) \\ V_N^{O*} + \ell(\mathbf{x}, \mathbf{u}^{\mathrm{o}}(0; \mathbf{x})) &\leq V_N(\mathbf{x}^+, \tilde{\mathbf{u}}(\mathbf{x})) - V_N^O(\mathbf{x}) + \ell(\mathbf{x}, u^{\mathrm{o}}(0; \mathbf{x})) \quad (3.8) \end{aligned}$$

The condition of Eqn. 3.6 implies that the right-hand side of Eqn. 3.7 is less than zero

$$F(f(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}}))) - F(\mathbf{x}_f^{\mathrm{o}}) + \ell(\mathbf{x}_f^{\mathrm{o}}, \kappa_f(\mathbf{x}_f^{\mathrm{o}})) \equiv [F^* + \ell](\mathbf{x}, \kappa_f(\mathbf{x})) \leq 0 \qquad (3.9)$$

Substituting Eqns. 3.8 and 3.9 into Eqn. 3.7 gives the sufficient condition for stability of Eqn. 3.2

$$V_N^{O*} + \ell(\mathbf{x}, \mathbf{u}^{\mathrm{o}}(0; \mathbf{x})) \leq 0$$

This shows that if the conditions given in Eqns. 3.3 to 3.6 are satisfied, then the condition of Eqn. 3.2 is met, and the receding horizon controller is stable.

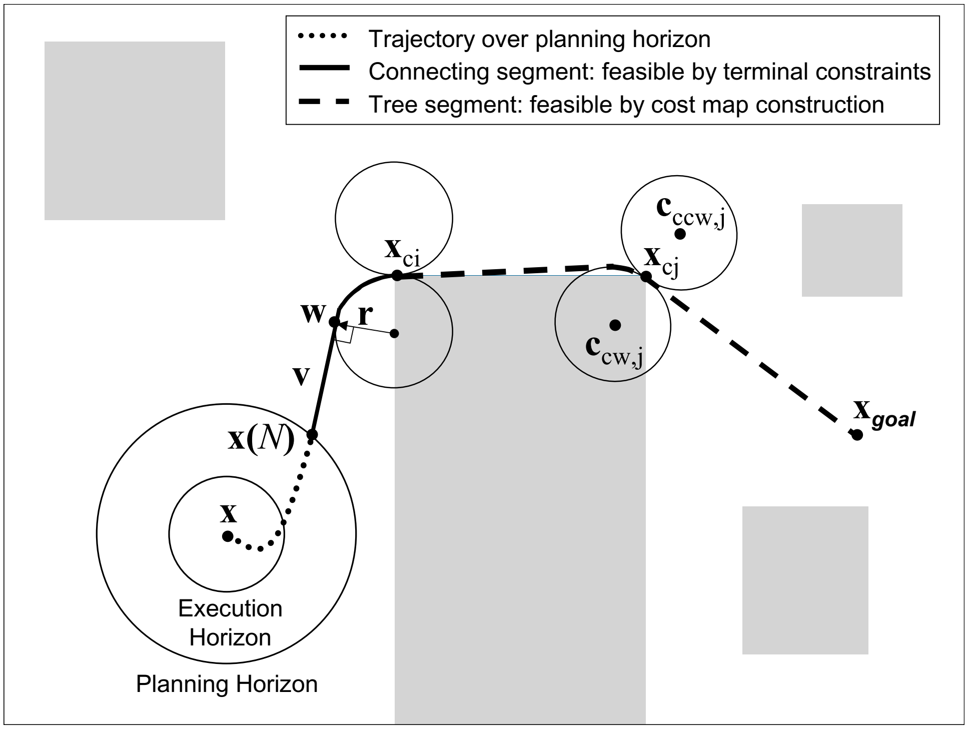

Trajectory over planning horizon
Connecting segment: feasible by terminal constraints
Tree segment: feasible by cost map construction
$\mathbf{x}_{ci}$
$\mathbf{c}_{ccw,j}$
$\mathbf{x}_{cj}$
$\mathbf{w}$
$\mathbf{r}$
$\mathbf{c}_{cw,j}$
$\mathbf{v}$
$\mathbf{x}_{goal}$
$\mathbf{x}(N)$
$\mathbf{x}$
Execution Horizon
Planning Horizon

Fig. 3-1: Resolution Levels of the Stable Receding Horizon Trajectory Designer. The connecting segment and tree segment make up the path that $\kappa_f(\mathbf{x})$ would follow and guarantee the feasibility of the trajectory optimization past the planning horizon to the goal.

3.2 Stable Receding Horizon Trajectory Designer

The receding horizon controller stability analysis techniques described above will be applied to prove that a modified version of the receding horizon controller is stable and reaches the goal in bounded time. A mixed-integer linear program form of this controller's optimization problem has been implemented in AMPL and is presented in Section 3.3.

The operation of the stable receding horizon trajectory planner is shown schematically in Fig. 3-1. Before the trajectory optimization is performed, a tree of kinodynamically feasible paths and their lengths is found from a discrete set of N_C states $[X_c^T \ \dot{X}_c^T]^T$ to the goal. In the trajectory design phase, a detailed trajectory segment over the planning horizon is optimized using a discrete-time model of the vehicle dynamics. The set $\mathcal{X}_f$ of allowed terminal states is chosen as those states from which κ_f can follow a simple "connecting segment" from $[\mathbf{x}(N)^T \ \dot{\mathbf{x}}(N)^T]^T$ to join with the tree of feasible paths. The connecting segment connects with constant heading from $\mathbf{x}(N)$ to $\mathbf{w}$ along $\mathbf{v}$, then turns at the vehicle's maximum rate over a path from $\mathbf{w}$ to $[\mathbf{x}_{\text{ci}}^T \ \dot{\mathbf{x}}_{\text{ci}}^T]^T \in [X_c^T \ \dot{X}_c^T]^T$. The vehicle can then follow a "tree segment" from the pre-computed set to the goal.

The connecting segment, from $\mathbf{x}(N)$ to a state of known cost, is discussed in Section 3.2.1. The selection of the states of known cost, and the construction of the tree of trajectories, is discussed in Section 3.2.2. The stability proof of the overall controller is given in Section 3.2.3.

3.2.1 Terminal Constraint Set

The definition of the terminal constraint set makes use of two circles centered at $\mathbf{c}_{\text{cw,i}}$ and $\mathbf{c}_{\text{ccw,i}}$ for every one of the N_C states of known cost $[\mathbf{x}_{\text{ci}}^T \ \dot{\mathbf{x}}_{\text{ci}}^T]^T$, as shown in Fig 3-1. These circles model the clockwise and counter-clockwise turning path of the vehicle up to $[\mathbf{x}_{\text{ci}}^T \ \dot{\mathbf{x}}_{\text{ci}}^T]^T$. The circles have a radius of ρ, the vehicle's turning radius at maximum speed v_{max}. The circle centers $\mathbf{c}_{\text{cw,i}}$ and $\mathbf{c}_{\text{ccw,i}}$ are positioned so that $\mathbf{x}_{\text{ci}}$ is on their circumference, and oriented so that $\dot{\mathbf{x}}_{\text{ci}}$ is tangent to both.

$$\forall\, i \in \mathcal{C}:$$

$$\mathbf{c}_{\text{cw,i}} = \mathbf{x}_{\text{ci}} + \frac{\rho}{v_{\text{max}}} \begin{bmatrix} \dot{x}_{\text{ci2}} \\ -\dot{x}_{\text{ci1}} \end{bmatrix} \tag{3.10}$$

$$\mathbf{c}_{\text{ccw,i}} = \mathbf{x}_{\text{ci}} + \frac{\rho}{v_{\text{max}}} \begin{bmatrix} -\dot{x}_{\text{ci2}} \\ \dot{x}_{\text{ci1}} \end{bmatrix} \tag{3.11}$$

where $C = C_{\text{cw}} \cup C_{\text{ccw}}$ are the circle centers, and $\mathcal{C} = \{1 \ldots N_C\}$. The set $\mathcal{X}_f$ is those which have a velocity tangent to one of the turning circles. This property guarantees that a connecting segment exists from all states in $\mathcal{X}_f$ to a state in $[X_c^T \ \dot{X}_c^T]^T$, and is expressed through constraints on the straight portion $\mathbf{v}$ of the connecting segment. The connecting segment joins the terminal state $[\mathbf{x}(N)^T \ \dot{\mathbf{x}}(N)^T]^T$ to the point $\mathbf{w} = \mathbf{c}_i + \mathbf{r}$ on the turning circle as shown in Fig. 3-1. The vector $\mathbf{v}$ is constrained to be in the direction $\dot{\mathbf{x}}(N)/\|\dot{\mathbf{x}}(N)\|$ of the terminal state's velocity and perpendicular to $\mathbf{r}$.

$$\forall(\mathbf{x}, \dot{\mathbf{x}}) \in \mathcal{X}_f, \exists\ \mathbf{c}_i \in C, \mathbf{r}, \mathbf{v} :$$

$$\begin{aligned} \mathbf{x} + \mathbf{v} &= \mathbf{c}_i + \mathbf{r} \\ \mathbf{v} &= V \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} && (3.12) \\ \mathbf{v} \cdot \mathbf{r} &= 0 && (3.13) \\ \|\mathbf{r}\| &= \rho \end{aligned}$$

where V is a scalar. All points on turning circles, including the states of known cost $\mathbf{x}_{\mathrm{ci}}$, are in $\mathcal{X}_f$ and therefore satisfy the invariance constraint of Eqn. 3.5 over the turning circles. Guaranteeing obstacle avoidance over the connecting segment will be discussed in section 3.3.

3.2.2 The Tree of Trajectory Segments

As described above, the receding horizon controller's optimization problem makes use of a tree of kinodynamically feasible trajectories from a limited set of states to the goal. An example of a such a tree is shown in Fig. 3-2. This section presents an algorithm to find the tree.

As discussed in Section 2.4.2, minimum distance trajectories tend to connect the vehicle's starting position, vertices of obstacles, and the goal. These points are chosen as the positions X_c of the states of known cost. The velocities $\dot{X}_c$ of the states of known cost have magnitudes equal to the vehicle's maximum speed $v_{\max}$. Their direction is chosen, as the tree is being constructed, to follow the next segment in the tree, so that all states on the trajectory tree satisfy $\mathcal{X}_f$. Once the velocity $\dot{\mathbf{x}}_{\mathrm{ci}}$ of a state of known cost is chosen, the circle centers $\mathbf{c}_{\mathrm{cw,i}}$ and $\mathbf{c}_{\mathrm{ccw,i}}$ are set as specified in Eqns. 3.10 and 3.11. The distance d_{i} from the i^{th} state of known cost $\mathbf{x}_{\mathrm{ci}}$ to the goal is also given by this algorithm.

Algorithm 2 is applied to grow the tree of kinodynamically feasible paths from all states

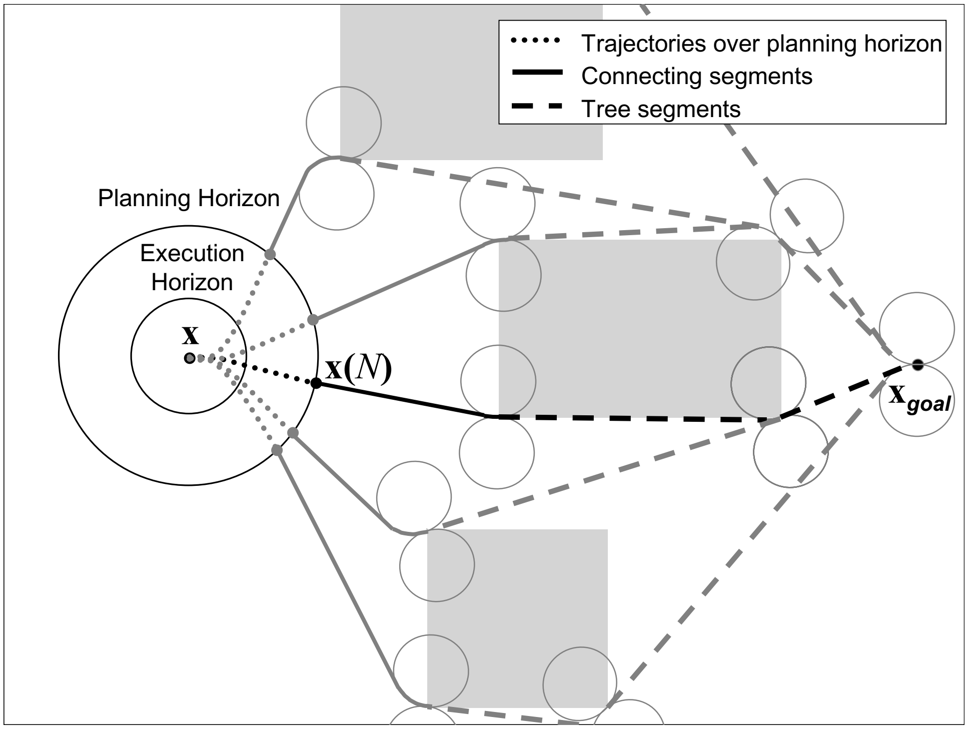


Fig. 3-2: An example of a tree of kinodynamically feasible trajectories to the goal. For each leaf of the tree, a trajectory over the planning horizon and a connecting segment is also shown in grey. The optimal position for $\mathbf{x}(N)$ minimizes the length of the associated connecting and tree segments, and is shown in black.

in $[X_c^T \ \dot{X}_c^T]^T$ to the goal $\mathbf{x}_{\text{goal}}$. When MAKE-PATH-TREE is called, the desired positions X_c for the states of known cost are passed in. MAKE-PATH-TREE divides X_c into two sets; $\mathcal{N}$, whose path to the goal has been fixed, and $\bar{\mathcal{N}}$, whose path to the goal could still be improved. At each iteration, the algorithm chooses a state of known cost $[\mathbf{x}_{\text{ci}}^T \ \dot{\mathbf{x}}_{\text{ci}}^T]^T$ to move from $\bar{\mathcal{N}}$ to $\mathcal{N}$, effectively fixing its path and adding it as a branch to the current tree of paths. Each position $\mathbf{x}_{\text{ci}} \in \bar{\mathcal{N}}$ is then examined.

The routine FIND-CONNECTING-SEGMENT is used to find a trajectory $[X^T \ \dot{X}^T]^T$ from $\mathbf{x}_{\text{ci}}$ to state $[\mathbf{x}_{\text{cj}}^T \ \dot{\mathbf{x}}_{\text{cj}}^T]$. As shown in Fig. 3-3, this trajectory has the properties of a connecting segment: it maintains a speed of $v_{\max}$ while following a tangent to the turning circle centered at $\mathbf{c}_{\text{cw,j}}$ or $\mathbf{c}_{\text{ccw,j}}$, then turns around this circle to reach $[\mathbf{x}_{\text{cj}} \ \dot{\mathbf{x}}_{\text{cj}}]$. If this path does not penetrate

```
algorithm $(\mathrm{C}_{\mathrm{cw}}, \mathrm{C}_{\mathrm{ccw}}, \dot{\mathrm{X}}_{\mathrm{c}}, \mathbf{d}) = \textsc{Make-Path-Tree}(X_c, \mathcal{X}_{\mathrm{obst}}, \mathbf{x}_{\mathrm{goal}}, \dot{\mathbf{x}}_{\mathrm{goal}})$
    $d_1 := 0$;  \\ initialize data for goal
    $\mathbf{x}_{\mathrm{c},1} := \mathbf{x}_{\mathrm{goal}}$; $\dot{\mathbf{x}}_{\mathrm{c},1} := \dot{\mathbf{x}}_{\mathrm{goal}}$;
    $\mathbf{c}_{\mathrm{cw},1} := \mathbf{x}_{\mathrm{c},1} + \rho[\dot{x}_{\mathrm{goal},2}, -\dot{x}_{\mathrm{goal},1}]^T / v_{\mathrm{max}}$; $\mathbf{c}_{\mathrm{ccw},1} := \mathbf{x}_{\mathrm{goal},1} - \rho[\dot{x}_{\mathrm{goal},2}, -\dot{x}_{\mathrm{goal},1}]^T / v_{\mathrm{max}}$;
    Set all other costs in $\mathbf{d}$ to $\infty$;
    $\bar{\mathcal{N}} := X_c; \mathcal{N} := \emptyset$;
    while $\bar{\mathcal{N}} \neq \emptyset$ do
        Choose the node $\mathbf{x}_{\mathrm{j}}$ in $\bar{\mathcal{N}}$ with minimum $d_{\mathrm{j}}$
        Move $\mathbf{x}_{\mathrm{cj}}$ from $\bar{\mathcal{N}}$ to $\mathcal{N}$;
        $\textsc{Relax}(\mathrm{j}, \bar{\mathcal{N}}, \mathbf{d}, \dot{\mathrm{X}}_{\mathrm{c}}, \mathcal{X}_{\mathrm{obst}}, \mathrm{C}_{\mathrm{cw}}, \mathrm{C}_{\mathrm{ccw}}, \rho,\ v_{\mathrm{max}})$
    end while

procedure $\textsc{Relax}(\mathrm{j}, \bar{\mathcal{N}}, \mathbf{d}, \dot{\mathrm{X}}_{\mathrm{c}}, \mathcal{X}_{\mathrm{obst}}, \mathrm{C}_{\mathrm{cw}}, \mathrm{C}_{\mathrm{ccw}}, \rho,\ v_{\mathrm{max}})$
    for all $\mathbf{x}_{\mathrm{ci}} \in \bar{\mathcal{N}}$ do
        $(X, \dot{X}) := \textsc{Find-Connecting-Segment}(\mathbf{x}_{\mathrm{ci}}, \mathbf{c}_{\mathrm{cw,j}}, \mathbf{c}_{\mathrm{cw,j}})$;
        $\Delta d_{\mathrm{ij}} := \textsc{Path-Length}(X)$;
        if $\Delta d_{\mathrm{ij}} + d_{\mathrm{j}} < d_{\mathrm{i}}$ and $\forall\ \mathbf{x} \in X : \mathbf{x} \notin \mathcal{X}_{\mathrm{obst}}$ then
            $d_{\mathrm{i}} := \Delta d_{\mathrm{ij}} + d_{\mathrm{j}}$;  \\ shorten path by going through j
            $\dot{\mathbf{x}}_{\mathrm{ci}} := \dot{\mathbf{x}}_0$;  \\ direct velocity to head down next segment in path
            $\mathbf{c}_{\mathrm{cw,i}} := \mathbf{x}_{\mathrm{ci}} + \rho[\dot{x}_{12}, -\dot{x}_{11}]^T / v_{\mathrm{max}}$; $\mathbf{c}_{\mathrm{ccw,i}} := \mathbf{x}_{\mathrm{ci}} + \rho[-\dot{x}_{12}, \dot{x}_{11}]^T / v_{\mathrm{max}}$;
        end if
    end for
```

Algorithm 2: Constructing the Feasible Path Tree: This is a modified form of Dijkstra's Algorithm (see Alg. 1), in which costs are found at the same time as paths are grown to the goal.

an obstacle, and passing through $\mathbf{x}_{\mathrm{cj}}$ shortens the path from $\mathbf{x}_{\mathrm{ci}}$ to the goal, then the distance, velocity, and circles associated with $\mathbf{x}_{\mathrm{ci}}$ are updated. The velocity $\dot{\mathbf{x}}_{\mathrm{ci}}$ is always chosen as the first velocity $\dot{\mathbf{x}}_0$ from the connecting segment.

By this construction, all states on the tree of trajectory segments are also in $\mathcal{X}_f$, because they either have a velocity tangent to a turning circle or are on a turning circle. Also, by this construction the end of any connecting segment is continuous in position and velocity with the start of a tree segment. These properties will be exploited in the next section to prove the stability of the controller.

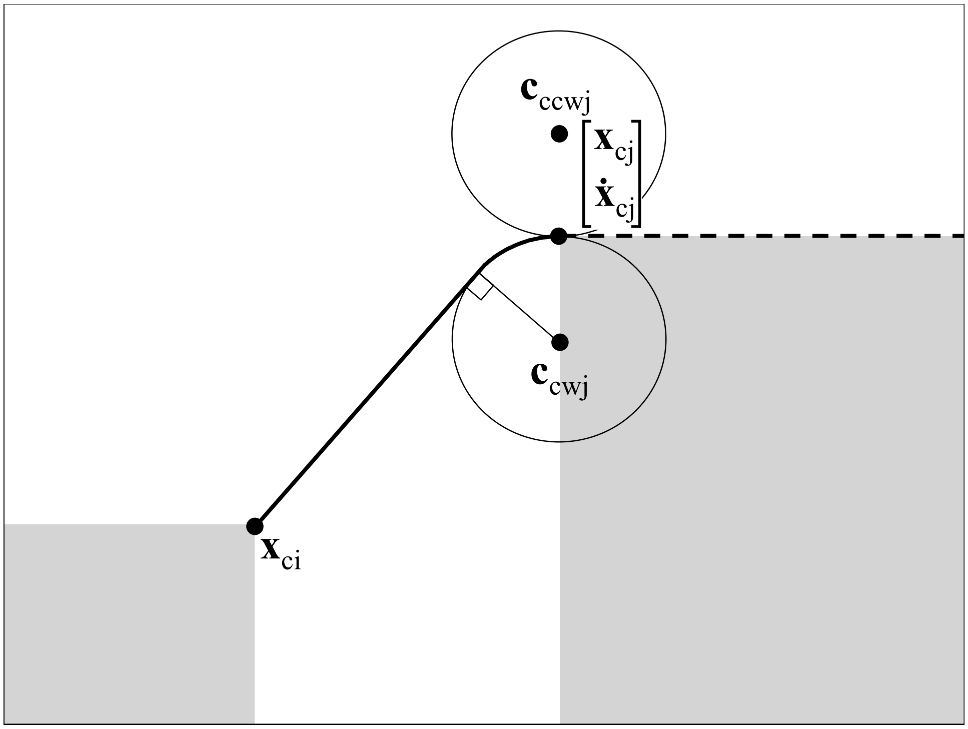

$\mathbf{c}_{\mathrm{ccwj}}$
$\begin{bmatrix}\mathbf{x}_{\mathrm{cj}} \\ \dot{\mathbf{x}}_{\mathrm{cj}}\end{bmatrix}$
$\mathbf{c}_{\mathrm{cwj}}$
$\mathbf{x}_{\mathrm{ci}}$

Fig. 3-3: The Trajectory Returned by Find-Connecting-Segment. This routine connects the $\mathbf{x}_{\mathrm{ci}}$ to the state $[\mathbf{x}_{\mathrm{cj}}^T \dot{\mathbf{x}}_{\mathrm{cj}}^T]^T$. First, it finds the shortest line through $\mathbf{x}_{\mathrm{ci}}$ that is tangent to the turning circle at $\mathbf{c}_{\mathrm{cw,i}}$ or $\mathbf{c}_{\mathrm{ccw,i}}$ and compatible with the turning direction of the circle. The turning path from the tangent to $[\mathbf{x}_{\mathrm{j}}^T \dot{\mathbf{x}}_{\mathrm{j}}^T]^T$ is then added.

3.2.3 Stability Analysis

The stable receding horizon controller solves the optimization problem given in Eqn. 3.1, with the coordinate frame chosen with $\mathbf{x}_{\mathrm{goal}}$ at the origin. The tree of trajectory segments is rooted at the goal, satisfying the constraint of Eqn. 3.3. If the trajectory optimization is feasible the first time it is performed, the constraints on $\mathbf{x}(N)$ guarantee that the connecting segment and tree segment together provide a trajectory to the goal. This trajectory is made up of straight segments connecting arcs around turning circles. The turning circles have radius $||\mathbf{r}|| = \rho$, the vehicle's minimum turning radius, so the trajectory is dynamically feasible. The terminal controller κ_f is chosen to follow this trajectory, satisfying the control admissibility constraint of Eqn 3.4.

The tree of trajectories is itself made up of connecting segments which fly straight, then turn onto a state of known cost closer to the goal. Therefore, all points along tree segments satisfy the terminal constraint set $\mathcal{X}_f$. This satisfies the invariance constraint of Eqn 3.5, and the next trajectory optimization is guaranteed to have this implicitly constructed trajectory as a feasible solution.

The terminal penalty $F(\mathbf{x})$ is chosen to be the distance that κ_f would travel to reach the goal from $\mathbf{x}$

$$F(\mathbf{x}) = V + \rho|\theta| + d_{\mathrm{i}} \tag{3.14}$$

where V is defined in Eqn. 3.12 as the length of the tangent from $\mathbf{x}(N)$ to the point $\mathbf{w}$ at which it intersects the circle, $|\theta|$ is the magnitude of the angle that the vehicle turns between $\mathbf{w}$ and $\mathbf{x}_{\mathrm{ci}}$, and d_{i} is the length of the kinodynamically feasible path found from $\mathbf{x}_{\mathrm{ci}}$ to the goal.

The state and control penalty $\ell(\mathbf{x}(i), \mathbf{u}(i))$ is chosen to be the distance that will be covered over the following time-step, starting at $\mathbf{x}(i)$ and applying $\mathbf{u}(i)$, so that

$$\ell(\mathbf{x}(i), \mathbf{u}(i)) = F(\mathbf{x}) - F(\mathbf{x}^+) = F^*(\mathbf{x}) \tag{3.15}$$

With this definition, $[F^* + \ell](\mathbf{x}, \kappa_f(\mathbf{x})) = 0$ so the rate of decrease constraint of Eqn. 3.6 is satisfied as an equality. Since the entire path associated with $F(\mathbf{x})$ is feasible, it is an upper bound on the length of the path that the optimizing controller will actually follow and $F(\mathbf{x})$ will decrease by $\ell(\mathbf{x}(i), \mathbf{u}(i))$ at every time-step. This proves this controller is stable, and will reach the goal in bounded time.

3.3 MILP Formulation of Controller

While the formulation of the receding horizon controller presented in Section 3.2 is provably stable, it is nonlinear. The optimization problem presented in Section 3.2 includes the

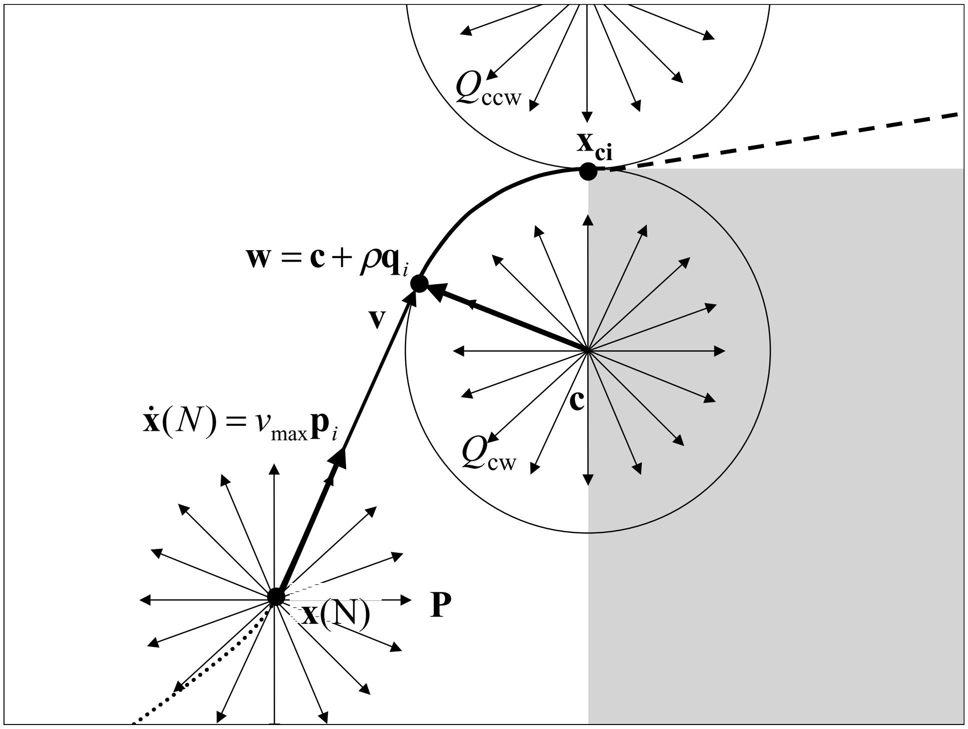

Q_{ccw}
$\mathbf{x}_{\mathbf{ci}}$
$\mathbf{w} = \mathbf{c} + \rho\mathbf{q}_i$
$\mathbf{v}$
$\dot{\mathbf{x}}(N) = v_{\max}\mathbf{p}_i$
$\mathbf{c}$
Q_{cw}
$\mathbf{x}(\mathrm{N})$
$\mathbf{P}$

Fig. 3-4: The vectors involved in the tangency constraints. The controller simultaneously chooses the direction for $\dot{\mathbf{x}}(N)$ and a perpendicular direction for $\mathbf{w}-\mathbf{c}$ from two finite sets of mutually perpendicular unit vectors. The vector $\mathbf{v}$ is also constrained to be in the direction of $\dot{\mathbf{x}}(N)$.

requirement that the velocity at the end of the planning horizon $\dot{\mathbf{x}}(N)$ be tangent to one of the turning circles. As given in Eqn. 3.13, this constraint can be written using a dot product of the decision variables $\dot{\mathbf{x}}$ and $\mathbf{r}$, showing that this constraint is quadratic. Furthermore, the angle through which the vehicle turns is used in the terminal penalty given in Eqn. 3.14, but is not available directly.

Because nonlinear optimization is typically far more difficult than linear optimization, it is desirable to reformulate this controller as a linear optimization problem. This section presents a version of the receding horizon controller that uses a discretization in order to formulate the controller as a MILP. This discretized version is also stable, and its computational performance will be discussed in Section 3.4.

3.3.1 Tangency Constraint

In section 3.2.1, Eqn. 3.13 used a quadratic term to constrain the terminal velocity to be tangent to a turning circle. The MILP formulation of the trajectory designer converts this constraint into a discrete, linear form. The vectors involved in the new constraints are shown in Fig. 3-4. The controller's choice of one circle center $\mathbf{c}$ from the set C is encoded by the vectors of binary decision variables $\mathbf{b}_{\mathrm{cw}}$ and $\mathbf{b}_{\mathrm{ccw}}$.

$$\begin{aligned} \mathbf{c} &= \sum_{\mathrm{i}\in\mathcal{C}} (b_{\mathrm{cw,i}}\mathbf{c}_{\mathrm{cw,i}} + b_{\mathrm{ccw,i}}\mathbf{c}_{\mathrm{ccw,i}}) \\ \sum_{\mathrm{i}\in\mathcal{C}} (b_{\mathrm{cw,i}} + b_{\mathrm{ccw,i}}) &= 1 \\ \mathbf{b}_{\mathrm{cw}} \in \{0,1\}^{\mathrm{N_C}} &\quad , \quad \mathbf{b}_{\mathrm{ccw}} \in \{0,1\}^{\mathrm{N_C}} \end{aligned}$$

The choice of a direction for $\dot{\mathbf{x}}(N)$ from a set P of N_P possible unit vectors is encoded with the vectors of binary decision variables $\mathbf{p}_{\mathrm{cw}} \cup \mathbf{p}_{\mathrm{ccw}}$.

$$\begin{aligned} \dot{\mathbf{x}}(N) &= v_{\max} \sum_{\mathrm{i}\,\in\,\mathcal{P}} (p_{\mathrm{cw,i}} + p_{\mathrm{ccw,i}})\mathbf{p}_i \\ \sum_{\mathrm{i}\,\in\,\mathcal{P}} (p_{\mathrm{cw,i}} + p_{\mathrm{ccw,i}}) &= 1 \\ \mathbf{p}_{\mathrm{cw}} \in \{0,1\}^{\mathrm{N_P}} &\quad , \quad \mathbf{p}_{\mathrm{ccw}} \in \{0,1\}^{\mathrm{N_P}} \end{aligned}$$

where $\mathcal{P} = \{1 \ldots N_P\}$. To enforce orthogonality, the binary variables $\mathbf{p}_{\mathrm{cw}} \cup \mathbf{p}_{\mathrm{ccw}}$ simultaneously determine the direction of $\mathbf{r} = \mathbf{w} - \mathbf{c}$ from the set $Q = Q_{\mathrm{cw}} \cup Q_{\mathrm{ccw}}$ of perpendicular unit vectors. The set Q_{cw} (Q_{ccw}) is constructed so that $\mathbf{q}_{\mathrm{ccw,i}}$ ($\mathbf{q}_{\mathrm{ccw,i}}$) is the correct direction for $\mathbf{r}$ if the vehicle reaches the clockwise (counter-clockwise) turning circle on a tangent in the direction $\mathbf{p}_i$.

$$\mathbf{q}_{\mathrm{cw,i}} = \begin{bmatrix} -p_{i,2} \\ p_{i,1} \end{bmatrix}$$

$$\mathbf{q}_{\mathrm{ccw,i}} = \begin{bmatrix} p_{i,2} \\ -p_{i,1} \end{bmatrix}$$

The direction for $\mathbf{r}$ is selected as

$$\mathbf{r} = \rho \sum_{i\ \in\ \mathcal{P}} (p_{\mathrm{cw,i}}\mathbf{q}_{\mathrm{cw,i}} + p_{\mathrm{ccw,i}}\mathbf{q}_{\mathrm{ccw,i}})$$

If a turning circle is chosen from C_{cw} by setting one of $\mathbf{b}_{\mathrm{cw,i}} = 1$, then the optimization must choose a direction for $\mathbf{r}$ from Q_{cw} by setting one of the $p_{\mathrm{cw,j}} = 1$ and similarly for the counter-clockwise direction. The following constraints are added to enforce this.

$$\begin{aligned} \sum_{\mathrm{j}\ \in\ \mathcal{P}} p_{\mathrm{cw,j}} &= \sum_{\mathrm{i}\in\mathcal{C}} b_{\mathrm{cw,i}} \\ \sum_{\mathrm{j}\ \in\ \mathcal{P}} p_{\mathrm{ccw,j}} &= \sum_{\mathrm{i}\in\mathcal{C}} b_{\mathrm{ccw,i}} \end{aligned}$$

To enforce tangency, $\mathbf{v}$ must be parallel to $\dot{\mathbf{x}}(N)$ and thus perpendicular to the selected direction for $\mathbf{r}$.

$$\begin{aligned} \mathbf{v} &= \mathbf{c} + \mathbf{r} - \mathbf{x}(N) \\ \forall\ i \in \mathcal{P} : & \\ \mathbf{v} \cdot \mathbf{q}_{\mathrm{cw,i}} &\leq \epsilon + M(1 - p_{\mathrm{cw,i}}) \\ \mathbf{v} \cdot \mathbf{q}_{\mathrm{cw,i}} &\geq -\epsilon - M(1 - p_{\mathrm{cw,i}}) \\ \mathbf{v} \cdot \mathbf{q}_{\mathrm{ccw,i}} &\leq \epsilon + M(1 - p_{\mathrm{ccw,i}}) \\ \mathbf{v} \cdot \mathbf{q}_{\mathrm{ccw,i}} &\geq -\epsilon - M(1 - p_{\mathrm{ccw,i}}) \end{aligned}$$

where ϵ is a small positive number and M is a large positive number. The tangency constraints are enforced for $\mathbf{q}_{\mathrm{cw,i}}$ or $\mathbf{q}_{\mathrm{ccw,i}}$ if $p_{\mathrm{cw,i}} = 1$ or $p_{\mathrm{ccw,i}} = 1$, and are relaxed otherwise. These constraints guarantee that the velocity vector $\dot{\mathbf{x}}(N)$ is tangent to one of the turning circles. Obstacle avoidance constraints (Eqns. 2.4 and 2.5), the dynamics model (Eqn. 2.6), the velocity limit (Eqn. 2.7), and the control force limit (Eqn. 2.8) are also enforced in this formulation.

Obstacle avoidance over the tangent portion of the connecting segment is enforced by constraining a discrete set of interpolating points along $\mathbf{v}$ to lie outside of all obstacles as given in Eqn. 2.13. Over the turn from $\mathbf{w}$ to the selected state of known cost $\mathbf{x}_{\mathrm{c}}$, constraints on the selected direction for $\mathbf{r}$ must be added to enforce obstacle avoidance. The turning path from $\mathbf{w} = \mathbf{c}_{\mathrm{cw,i}} + \rho\mathbf{q}_{\mathrm{cw,j}}$ to the state of known cost $\mathbf{x}_{\mathrm{c,i}}$ is checked before the trajectory design begins. If the path penetrates any obstacles, then the simultaneous selection of $\mathbf{x}_{\mathrm{c,i}}$ and $\mathbf{q}_{\mathrm{cw,j}}$ is excluded by setting $a_{\mathrm{cw,i,j}} = 0$. The following constraints enforce this exclusion:

$$\forall\ j \in \mathcal{P}:$$

$$\begin{aligned} p_{\mathrm{cw,j}} &\leq \sum_{i\in\mathcal{C}} a_{\mathrm{cw,i,j}} b_{\mathrm{cw,i}} \\ p_{\mathrm{ccw,j}} &\leq \sum_{i\in\mathcal{C}} a_{\mathrm{ccw,i,j}} b_{\mathrm{ccw,i}} \end{aligned}$$

3.3.2 Selection of Unit Vectors in P

This discussion has not yet addressed the selection of particular directions for the unit vectors $\mathbf{p}_{\mathrm{i}} \in P$. There is some freedom in this choice. It is desirable minimize the number N_P of unit vectors in P, since each of the directions requires two binary variables to be added to the program, and the complexity of MILPs grows non-polynomially with the number of binary variables they include. However, the terminal state $\mathbf{x}(N)$ must be positioned on one of the tangent lines corresponding to the available unit vectors. If too few unit vectors are used, it might not be possible for the vehicle to reach any tangent line at the end of its planning horizon, resulting an infeasible program.

It is straightforward to avoid this by adding a unit vector to P. After the tree of trajectory segments is found as described in Section 3.2.2, and before each trajectory optimization is performed, a connecting segment from the vehicle's current state $[\mathbf{x}(0)^T\ \dot{\mathbf{x}}(0)^T]^T$ to a state of known cost $[\mathbf{x}_{\mathrm{ci}}^T\ \dot{\mathbf{x}}_{\mathrm{ci}}^T]^T$ is found using a modified version of the routine FIND-CONNECTING-

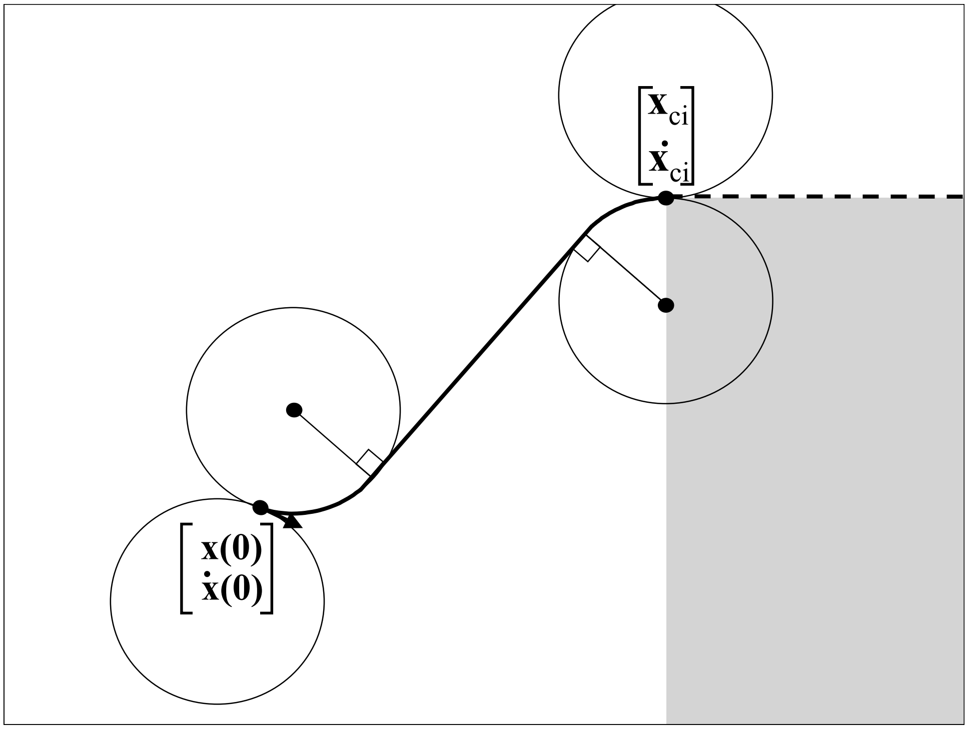

$\begin{bmatrix}\mathbf{x}_{ci}\\ \dot{\mathbf{x}}_{ci}\end{bmatrix}$
$\begin{bmatrix}\mathbf{x}(0)\\ \dot{\mathbf{x}}(0)\end{bmatrix}$

Fig. 3-5: The Trajectory between Two States Returned by FIND-CONNECTING-SEGMENT. This routine connects $[\mathbf{x}(0)^T\ \dot{\mathbf{x}}(0)^T]^T$ to $[\mathbf{x}_{\text{ci}}^T\ \dot{\mathbf{x}}_{\text{ci}}^T]^T$. First, it finds the shortest line that is tangent to turning circles for both states, and compatible with the turning direction of the circles. The turning paths from $[\mathbf{x}(0)^T\ \dot{\mathbf{x}}(0)^T]^T$ to the tangent, and from the tangent to $[\mathbf{x}_{\text{ci}}^T\ \dot{\mathbf{x}}_{\text{ci}}^T]^T$ are then added.

SEGMENT. An example of such a path is shown in Fig. 3-5. The state of known cost is chosen to minimize the associated total path length. The state $[\mathbf{x}(N)^T\ \dot{\mathbf{x}}(N)^T]^T$ is found that would be reached at the end of the planning horizon if the connecting segment and the following tree segment were followed. By adding the vehicle's heading $\dot{\mathbf{x}}(N)/\|\dot{\mathbf{x}}(N)\|$ to P, a feasible heading is guaranteed to be available to the trajectory optimization. If FIND-CONNECTING-SEGMENT cannot find a path to a state of known cost before the first optimization is performed, then the trajectory design problem is infeasible. After the first optimization is solved, FIND-CONNECTING-SEGMENT will always find a path to a state of known cost. This addition guarantees that the optimization program remains feasible, and

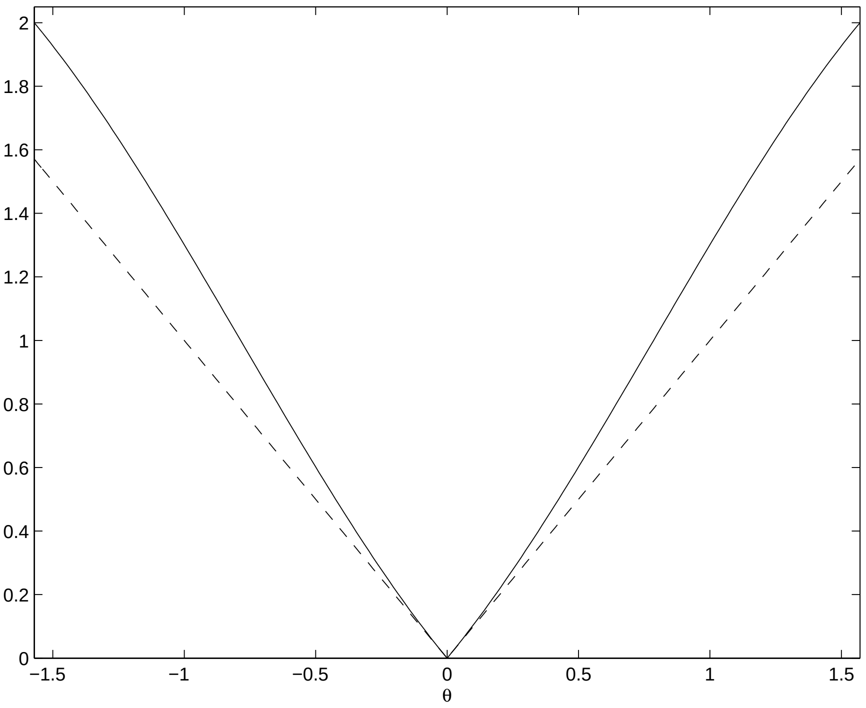


Fig. 3-6: Comparison of Upper Bounding Function $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ to $|\theta|$. $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ is shown with solid line, $|\theta|$ is shown with dashed line.

adding more unit vectors may make it possible to find solutions of lower cost.

3.3.3 Arc Length Approximation

In addition to enforcing the tangency requirement, the stable controller requires a nonlinear relationship to exactly evaluate the arc length $\rho|\theta|$ covered by the vehicle as it traverses the turning circle. This arc length contributes to the total length of the trajectory followed to the goal, so it must be included in the cost function.

The discretized stable controller replaces exact evaluation of $|\theta|$ in Eqn. 3.14 with an upper bound $|\theta|'$ on $|\theta|$ that is valid whenever $|\theta| \leq \pi/2$. By using an upper bound, the convergence constraint of Eqn. 3.6 is maintained.

$|\theta|$ is the magnitude of the angle between the vehicle's heading at the start of the turn, $\hat{\dot{\mathbf{x}}}(N) = \dot{\mathbf{x}}(N)/v_{\max}$, and the vehicle's heading at the end of the turn, $\hat{\dot{\mathbf{x}}}_{\mathrm{c}} = \dot{\mathbf{x}}_{\mathrm{c}}/v_{\max}$. Fig. 3-6 shows that the one-norm $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ provides an upper bound on $|\theta|$ whenever $|\theta| \leq \pi/2$.

This can also be established analytically by considering, without loss of generality, the case where $\hat{\dot{\mathbf{x}}}(N) = [1\ 0]^T$. Then

$$
\begin{aligned}
\hat{\dot{\mathbf{x}}}_{\mathrm{c}} &= \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \\
\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1 &= |\cos(\theta) - 1| + |\sin(\theta)|
\end{aligned}
$$

Over the range $0 \leq \theta \leq \pi/2$

$$
\begin{aligned}
\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1 &= -\cos(\theta) + 1 + \sin(\theta) \\
\frac{d}{d\theta}(\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1) &= \sin(\theta) + \cos(\theta) \geq 1 \geq \frac{d\theta}{d\theta} \qquad (3.16)
\end{aligned}
$$

The first inequality is true by the triangle inequality. At $\theta = 0$, $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1 = 0$, and Eqn. 3.16 shows that $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ increases faster than θ over $0 \leq \theta \leq \pi/2$, so $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ must be greater than θ over this range. $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ is symmetric about the $\theta = 0$ axis because

$$
\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1 = |\cos(\theta) - 1| + |\sin(\theta)| = |\cos(-\theta) - 1| + |\sin(-\theta)|
$$

Therefore, $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ must also be greater than $|\theta|$ for $-\pi/2 \leq \theta \leq 0$. This proves that $\|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1$ is an upper bound on $|\theta|$ over $|\theta| \leq \pi/2$, as required.

The following constraints are added to the discretized controller to evaluate $|\theta|' \geq \|\hat{\dot{\mathbf{x}}}(N) - \hat{\dot{\mathbf{x}}}_{\mathrm{c}}\|_1 \geq |\theta|$

$$
\begin{aligned}
|\theta|' &\geq (\hat{\dot{x}}(N)_1 - \hat{\dot{x}}_{\mathrm{c}1}) + (\hat{\dot{x}}(N)_2 - \hat{\dot{x}}_{\mathrm{c}2}) \\
\text{and } |\theta|' &\geq (\hat{\dot{x}}(N)_1 - \hat{\dot{x}}_{\mathrm{c}1}) - (\hat{\dot{x}}(N)_2 - \hat{\dot{x}}_{\mathrm{c}2}) \\
\text{and } |\theta|' &\geq -(\hat{\dot{x}}(N)_1 - \hat{\dot{x}}_{\mathrm{c}1}) + (\hat{\dot{x}}(N)_2 - \hat{\dot{x}}_{\mathrm{c}2}) \\
\text{and } |\theta|' &\geq -(\hat{\dot{x}}(N)_1 - \hat{\dot{x}}_{\mathrm{c}1}) - (\hat{\dot{x}}(N)_2 - \hat{\dot{x}}_{\mathrm{c}2})
\end{aligned}
$$

In order to restrict the controller to turn at most $\pi/2$ radians, heading directions $p_{\text{cw,i}}$ ($p_{\text{ccw,i}}$) that would result in a larger turn to the state of known cost $\mathbf{x}_{\text{c,i}}$ are also excluded by setting $a_{\text{cw,i,j}} = 0$ ($a_{\text{ccw,i,j}} = 0$).

3.3.4 Linearized Terminal Penalty

The use of a discrete-time dynamics model and fixed time-step size in the optimization problem requires one further modification to satisfy the assumptions of the stability analysis of the receding horizon controller. The stability analysis was performed using the length of the trajectory that $\kappa_f(\mathbf{x})$ would be fly to the goal as the terminal penalty for that state $F(\mathbf{x})$. Stability was proven assuming that the optimal trajectory, starting at any state, would have a lower cost than that associated with using $\kappa_f(\mathbf{x})$, so that $V_N^O(\mathbf{x}^+) \leq V_N(\mathbf{x}^+, \tilde{U}(\mathbf{x}))$. Because the optimization uses a discrete-time model of the aircraft dynamics and a fixed time-step size, the last trajectory designed by the optimization can only arrive at the goal exactly at a time-step, while the trajectory followed by $\kappa_f(\mathbf{x})$ is defined over continuous time, and can arrive at the goal at a time that not an integer number of time steps. The optimized trajectory may therefore arrive a fraction of a time step later than that designed by $\kappa_f(\mathbf{x})$, and may be longer than the trajectory followed by $\kappa_f(\mathbf{x})$.

This potential violation of the assumption that $V_N^O(\mathbf{x}^+) \leq V_N(\mathbf{x}^+, \tilde{U}(\mathbf{x}))$ does not greatly affect the proof that receding horizon controller reaches the goal, since the assumption can only be violated when the goal is actually reached. This emphasizes that the stability conditions of Eqns. 3.3–3.6 are sufficient but not necessary for stability. The condition of Eqn. 3.6 can be satisfied by increasing the terminal penalty by the maximum distance that can be covered over 1 time step, $v_{\max}\Delta T$ if the terminal state is not equal to the goal state. When the goal is actually reached, the terminal penalty decreases by at least the excess distance covered by the controller during the arrival delay. This is evaluated using the binary variable b_{goal}, which is one only if the goal is reached. The linearized terminal penalty is then

$$F(\mathbf{x}) = L(\mathbf{v}) + \rho|\theta|' + \sum_{\mathrm{i}\in\mathcal{C}}(b_{\mathrm{cw,i}} + b_{\mathrm{ccw,i}})d_{\mathrm{i}} + (1 - b_{\mathrm{goal}})v_{\max}\Delta T$$

where $L(\mathbf{v})$ approximates the two-norm of the vector $\mathbf{v}$ by taking the maximum value of its projection onto a series of unit vectors, ΔT is the time step size, and b_{goal} is evaluated using the same constraints given in Eqns. 2.2 and 2.3.

3.4 Simulation Results

Simulation results are presented below for the stable receding horizon trajectory designer. The stable receding horizon trajectory designer is capable of planning near-optimal trajectories in highly constrained environments, and does so with computational effort that is comparable to that of the unstable trajectory designer.

3.4.1 Operation of the Trajectory Designer

Successive trajectory segments designed by the stable trajectory designer are shown in Fig. 3-7. Although the controller selects a turning circle as part of every solution, it does not necessarily execute a trajectory that follows the circumference of a circle because the connecting segment that follows the suboptimal path is beyond the executing horizon, and the controller can replan it with new turning circles available before it is executed. This emphasizes that despite the sub-optimality of the path that is implicitly constructed from tangent lines and turning circles, the optimality of the executed trajectory is not limited to this construction. The tangent lines and turning circles simply provide a straightforward geometric construction of a feasible path to the goal. The stable receding horizon trajectory planner took 58.9 seconds to design this trajectory, and arrived at the goal in 36 steps.

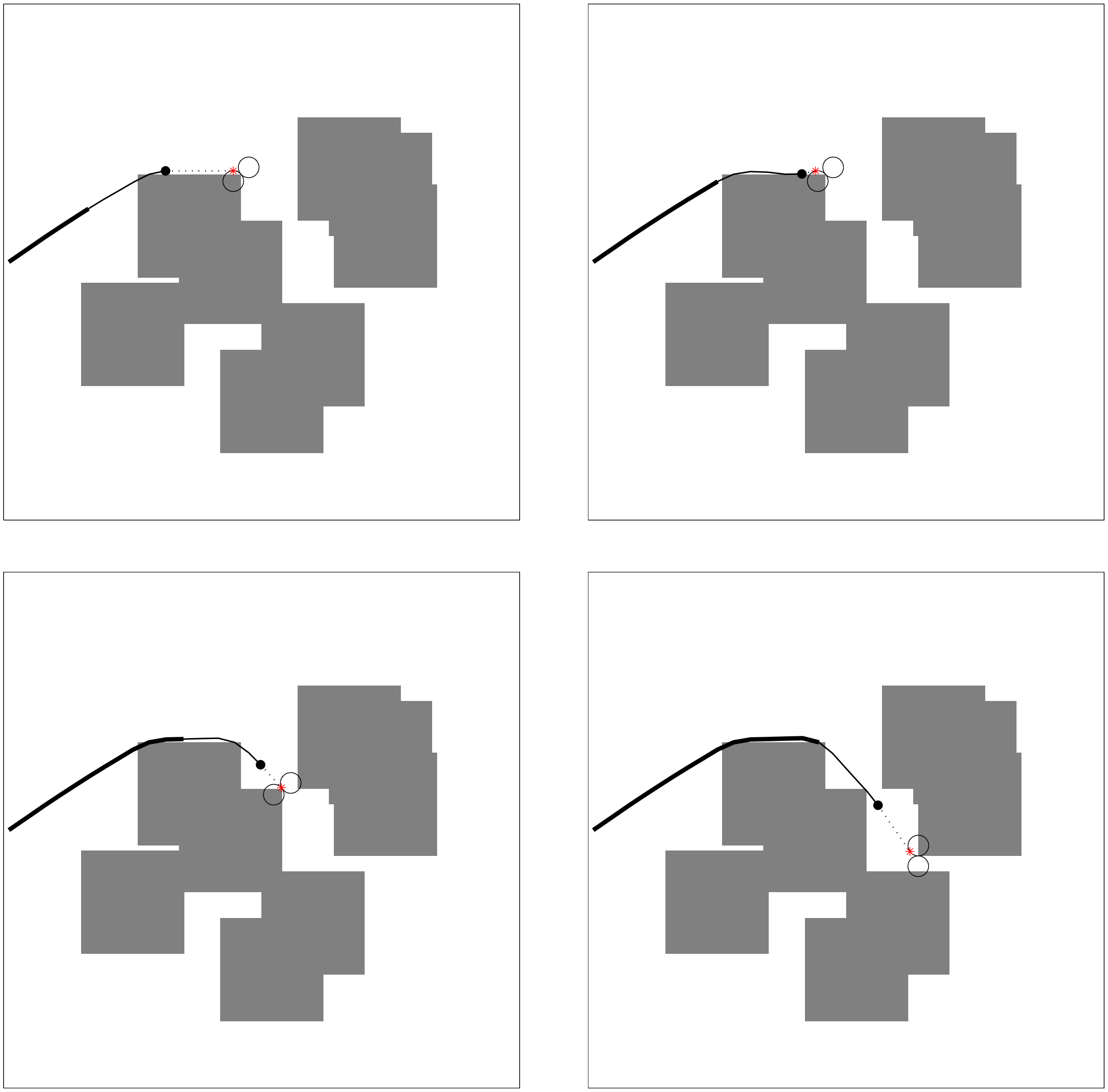


Fig. 3-7: Plans of the Stable Trajectory Designer. Successive trajectory segments designed by the stable trajectory designer are shown. Starting point is at left, goal is at right. The thick line shows executed steps, the thin line shows planned steps, and the dashed line shows $\mathbf{v}$. The terminal point $\mathbf{x}(N)$ is shown with $\bullet$, and $\mathbf{w}$ is shown with $*$.

3.4.2 Trajectory Design in Highly Constrained Environments

As described in Section 3.2, there is always an implicitly constructed kinodynamically feasible path from the terminal state $\mathbf{x}(N)$ chosen by the stable receding horizon designer to the goal.

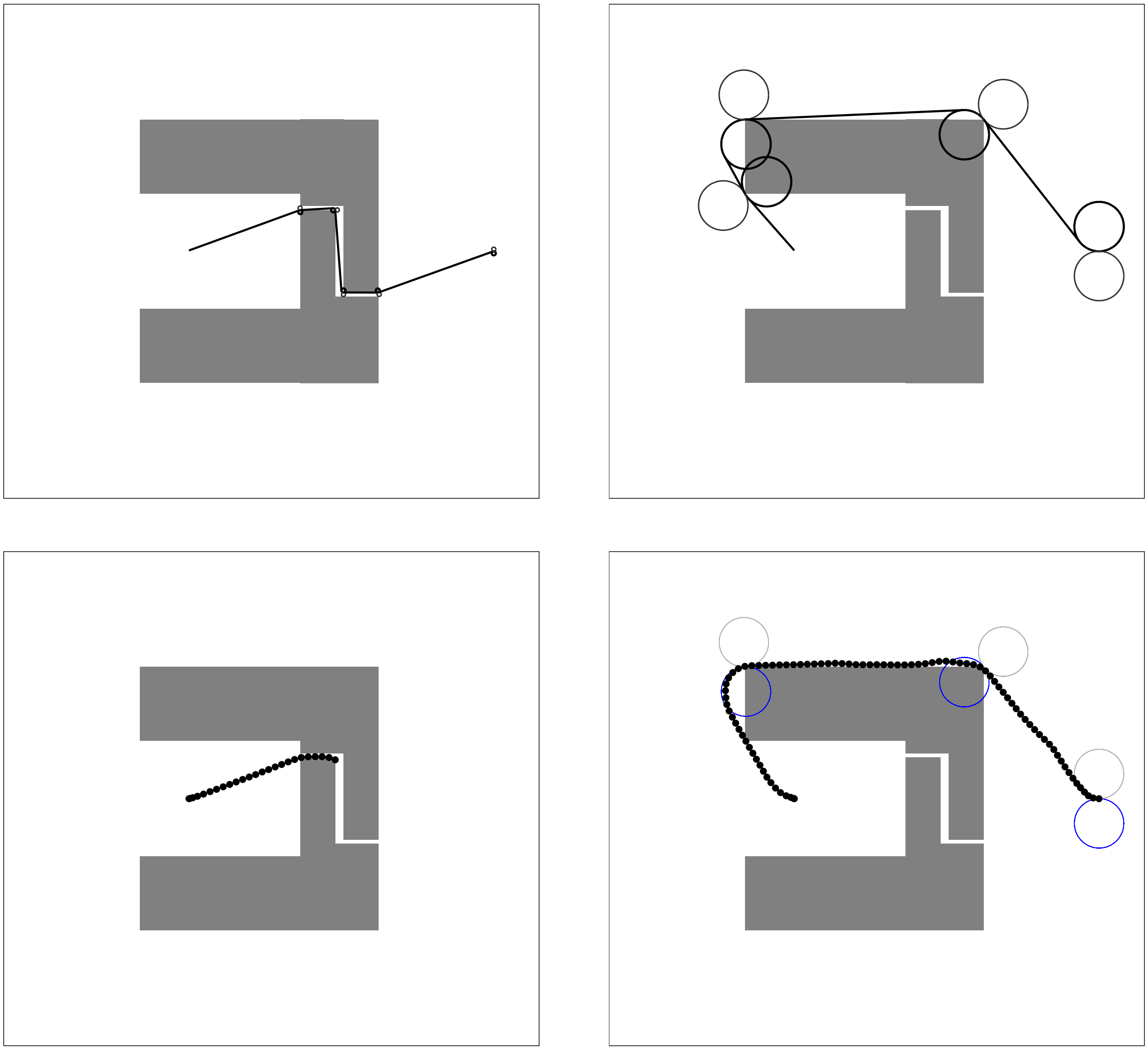


Fig. 3-8: Trajectory Examples in Highly Constrained Environment. Start is at center, goal is at right. Top row: kinodynamically feasible trajectories from starting point to goal, found during trajectory tree construction. Top left shows tree for vehicle with turning radius $\rho = 0.5$, top right corresponds to $\rho = 6$. Bottom row shows trajectories planned given $\rho = 6$. Bottom left is planned using unstable receding horizon trajectory planner. Note that optimization problem becomes infeasible before the goal is reached. Bottom right is planned with stable receding horizon controller: note that narrow path through obstacles is ruled out by trajectory tree construction and a longer, but kinodynamically feasible, path is actually followed.

Table 3.1: Cumulative Solution Times for Long Trajectories

Trajectory Design	Without Initial Guess		With Initial Guess	
Formulation	Sol'n. Time (s)	Num. Steps	Sol'n. Time (s)	Num. Steps
Unstable	136.2	160	91.45	160
Stable	168.2	161	116.8	161

This prevents the designer from forming a plan that relies on passing through a series of gaps between obstacles past the planning horizon that is kinodynamically infeasible.

An obstacle field with such a series of gaps is shown in figure 3-8. The unstable receding horizon trajectory planner is not prevented from selecting this path past the planning horizon, and does so because the narrow gap provides the shortest straight-line path to the goal. The same straight line path would be found in the construction of the kinodynamically feasible path to the goal for a vehicle with turning radius $\rho = 0$. However, if the vehicle's minimum turning radius is sufficiently larger than 0, it cannot pass through the same series of gaps and the trajectory design problem becomes infeasible. If a vehicle actually followed this path as it was being constructed, it would collide with an obstacle after this point.

3.4.3 Design of Long Trajectories

The stable receding horizon controller is guaranteed to reach the goal if its first optimization problem is feasible, but this controller's MILP formulation requires more binary variables than that of the unstable trajectory designer. The computational impact of this increase was examined by comparing the solution times of both controllers with equal planning horizons for a very long trajectory.

In order to shorten the solution time, candidate trajectory solutions were provided for each optimization problem that was solved. Starting the branch-and-bound search procedure with a lower upper bound on the optimal cost allows more of the search space to be pruned because its linear relaxation has a cost higher than the upper bound, shortening the search.

In the case of the unstable receding horizon controller, the candidate trajectory was

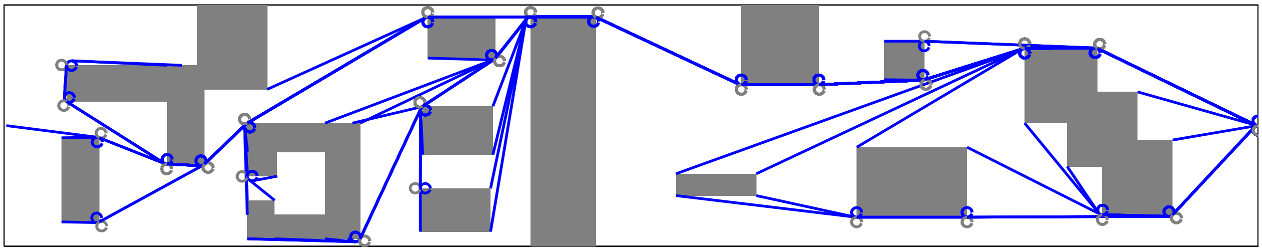


Fig. 3-9: Tree of Trajectory Segments for Long Trajectory. This tree of trajectory segments was produced by the stable receding horizon controller while solving the same trajectory design problem as shown in Fig. 2-7.

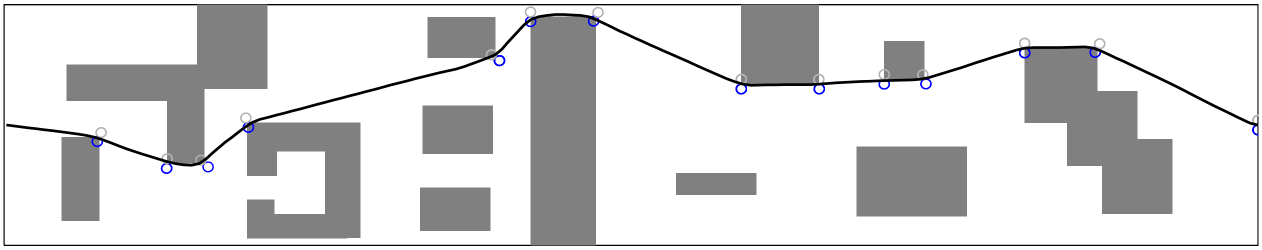


Fig. 3-10: Sample Long Trajectory Designed by Stable Controller.

constructed by reusing the unexecuted trajectory points from the previous solution, and replacing the executed steps by extrapolating along the line of visibility to the next state of known cost. The candidate trajectory for the stable receding horizon controller was constructed by reusing unexectued steps, and choosing trajectory points along the connecting and tree segments to the goal. Both of these procedures can be executed very rapidly. These candidate solutions were feasible more often for the stable controller than for the unstable controller, since there can be a sharp turn between the last state $\mathbf{x}(N)$ of the previous unstable controller's trajectory and the line of visibility, while the stable controller's terminal heading was guaranteed to be the same as that of the connecting segment.

The computation times for both formulations are given in Table 3.1, and the stable controller's tree of trajectories and solution is shown in Fig. 3-10. The trajectories found with and without providing candidate solutions were almost identical. These results show that the stable trajectory designer takes 23.7% longer than the unstable trajectory designer, and that the length of the trajectory planned by the stable controller was only 1 step longer than that designed by the unstable controller.

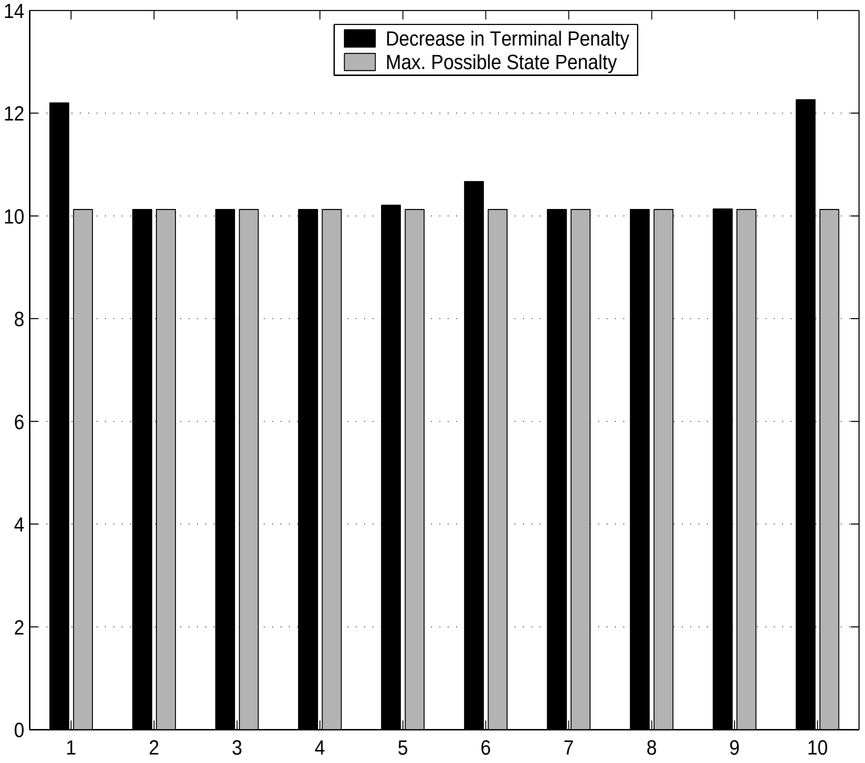


Fig. 3-11: Rate of Decrease of Stable Controller's Terminal Penalty. The decrease in terminal penalty exceeds the state penalty when a turn is executed because an upper bound on the turn's arc length is included in the terminal penalty.

3.4.4 Rate of Decrease of Terminal Penalty

The analysis of section 3.2 shows that a terminal controller $\kappa_f(\mathbf{x})$ exists which satisfies the constraint of Eqn. 3.6. This constraint specifies that as a series of trajectories are designed, each terminal cost must be less than the previous by at least the previous state and control penalties over the execution horizon. The state and control penalty was chosen as the distance that would be covered by $\kappa_f(x)$ over the execution horizon of the previous plan. Since the constraint of Eqn. 3.6 is satisfied, the optimizing controller should also be able to design trajectories whose terminal cost decreases at least the distance that $\kappa_f(\mathbf{x})$ would follow. This property was verified by solving a series of optimization problems to plan a trajectory to the goal, and recording the actual decrease in the terminal penalty. This decrease is plotted in Fig. 3-11, and is compared to the maximum distance that $\kappa_f(\mathbf{x})$ could cover over the execution horizon. In each case the decrease in terminal penalty was verified to be greater than the maximum distance that $\kappa_f(\mathbf{x})$ could cover. Since the terminal penalty is an over bound on the distance to the goal, and its rate of decrease is bounded from below, the controller must reach the goal in bounded time.

3.5 Conclusions

This chapter presents a stable receding horizon trajectory designer. Its stability is guaranteed by constraints that specify that each time a trajectory segment is optimized, a kinodynamically feasible trajectory exists past the extent of the planning horizon to the goal. This trajectory provides the next optimization with a feasible solution, and the stability of the controller is guaranteed. Computational results show that the stable trajectory designer is capable of designing trajectories through highly constrained obstacle fields, while requiring computation time that is modestly longer than that of the unstable controller. In less constrained environments, the stable receding horizon controller does not plan trajectories that are significantly longer than those of the unstable controller.

Chapter 4

Minimum Time Allocation

4.1 Introduction

Chapters 2 and 3 presented techniques for performing trajectory design. Before this can be performed, the position of the UAV's goal must be assigned. A sequence of goals for each UAV can be found by solving an allocation problem which divides the fleet's tasks amongst its members. However, many common choices for objectives to optimize when making this allocation are a function of the paths that each UAV would take to its goal, so the allocation problem relies on the other's result. These problems are therefore coupled, with each relying on the result of the other. This can be handled by designing trajectories for all possible allocations, then choosing the detailed trajectories that minimize the cost. However, each trajectory design problem is computationally demanding. Exactly designing all but a small set of trajectories is impractical, so another approach is required.

This chapter presents an approach to solving the allocation and trajectory design problems which partially decouples them to make their solution computationally tractable, while maintaining the essential aspects of the coupling in order to approach optimality. The approach presented in this chapter is applied to find minimum completion time solutions to the allocation problem. Completion times for various allocation options are estimated using path approximations. These path approximations can be constructed using the joined line segment path approximations that were applied in Chapter 2, or the kinodynamically feasible paths presented in Chapter 3. Joined line segment paths are presented in this chapter. This approach distributes the computational effort of some steps between parallel processing platforms to improve the solution time. This partially-decoupled approach is shown to yield coordinated mission plans that are very close to the optimal solution [24].

The allocation problem is formulated as a MILP problem. This formulation provides the ability to include more complex constraints such as capability constraints, which allow only a subset of the fleet to visit particular waypoints, and constraints which allow waypoints to be visited only in relative or absolute time windows. The resulting MILP problems can be formulated in AMPL [9] and solved readily using commercially available software such as CPLEX [10]. The combination of the approximate cost algorithm and task allocation formulation presented in this chapter provides a flexible and efficient means of assigning multiple objectives to multiple vehicles under various detailed constraints.

This chapter will define the allocation problem, then present a cost estimation algorithm. This is followed by a formulation of the allocation problem. A series of allocation problems is presented that demonstrate the effects of adding constraints. Results are them presented to evaluate the effects of the partial decoupling on performance and computational tractability.

4.2 Problem Formulation

The algorithm described here assumes that a set of tasks has been identified which must be performed by the team. The team is made up of N_V UAVs with known starting states and maximum velocities. The starting state of UAV v is given by the v^{th} row $[\ x_{0\text{v}} \quad y_{0\text{v}} \quad \dot{x}_{0\text{v}} \quad \dot{y}_{0\text{v}}\ ]$ of the matrix $\mathbf{S}_0$, and the maximum velocity of the UAVs is given $v_{\max}$. The waypoint locations are assumed to be known, and the position of waypoint w is given by the w^{th} row $[\ B_{\text{wx}} \quad B_{\text{wy}}\ ]$ of the matrix $\mathbf{B}$. The application of the algorithms presented in this chapter to No Fly Zones that are bounded by polygons is straightforward, but the case where the polygons are rectangles will be presented here for simplicity. The location of the lower-left corner is given by $(Z_{\mathrm{j1}}, Z_{\mathrm{j2}})$, and the upper-right corner by $(Z_{\mathrm{j3}}, Z_{\mathrm{j4}})$. Together, these two pairs make up the j^{th} row of the matrix $\mathbf{Z}$. Finally, the UAV capabilities are represented by a binary capability matrix $\mathbf{K}$. The entry K_{vw} is 1 if vehicle v is capable of performing the tasks associate with waypoint w, and 0 if not.

This algorithm produces a trajectory for each vehicle, represented for the v^{th} vehicle by a series of trajectory points $\mathbf{x}(v,i) \in \mathcal{R}^2 : i = 1, 2, \ldots, t_v$, where t_v is the time at which aircraft v reaches its final waypoint. The finishing times of all vehicles make up the vector $\mathbf{t}$.

This work is concerned with coordination and control problems in which the cost is a function of the resulting trajectories. This is a broad category of coordination and control problems, and includes costs that involve completion time or radar exposure. While a cost function has been chosen that penalizes both the maximum completion time and the average completion times over all UAVs, the approach presented here can be generalized to costs that involve other properties of the trajectories. The cost used in this chapter can be written as

$$\begin{aligned} \bar{t} &= \max_{v} \ t_v && (4.1) \\ J_1(\bar{t}, \mathbf{t}) &= \bar{t} + \frac{\alpha}{N_V} \sum_{v=1}^{N_V} t_v && (4.2) \end{aligned}$$

where $\alpha \ll 1$ weights the average completion time compared to the maximum completion time. If the penalty on average completion time were omitted (*i.e.,* $\alpha = 0$), the solution could assign unnecessarily long trajectories to all UAVs except for the last to complete its mission. Note that, because this cost is a function of the completion time for the entire fleet, it cannot be evaluated exactly until detailed trajectories have been planned that visit all the waypoints and satisfy all the constraints. The minimum cost coordination problem could be solved by planning detailed trajectories for all possible assignments of waypoints to UAVs and all possible orderings of those waypoints, then choosing the detailed trajectories that minimize cost function $J_1(\bar{t}, \mathbf{t})$, but there are many such possibilities and designing each is computationally demanding.

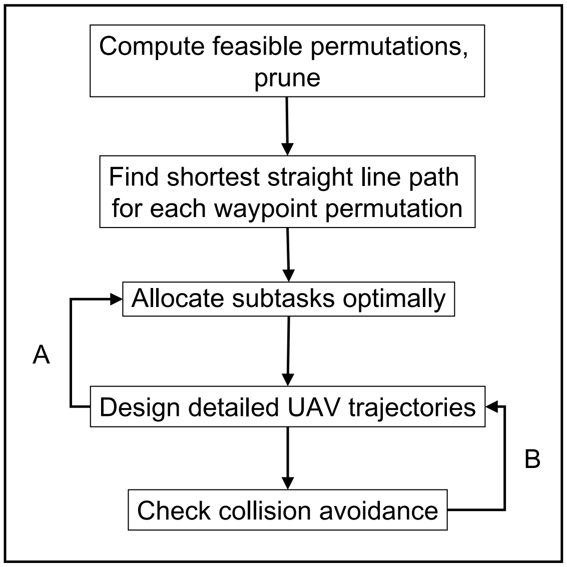

Compute feasible permutations, prune
Find shortest straight line path for each waypoint permutation
Allocate subtasks optimally
A
Design detailed UAV trajectories
B
Check collision avoidance

Fig. 4-1: Task Assignment and Trajectory Planning Algorithm

4.3 The Allocation Algorithm

4.3.1 Overview

Instead planning detailed trajectories for all possible task allocations, the algorithm presented in this chapter constructs estimates of the finishing times for a subset of the feasible allocations, then performs the allocation to minimize the cost function evaluated using the estimates. Next, detailed UAV trajectories are designed, and checked for collisions between vehicles. The main steps in the algorithm are shown in Fig. 4-1.

First, a list of all unordered feasible task combinations is 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 is 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. The minimum finishing time for each combination is estimated by dividing the length of the shortest path by the UAV's maximum speed. Some of the tasks 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, and solve the allocation problem faster.

With these estimated finishing times available, the task allocation problem can be performed to find the minimum of the estimated costs. Once the optimal task allocation is found using the estimated completion times, detailed kinematically and dynamically feasible trajectories that visit the assigned waypoints can be planned and checked for collision avoidance between UAVs [29]. If the minimum separation between UAVs is violated, the trajectories can be redesigned to enforce a larger separation distance (shown by loop B in Fig. 4-1). If desired, or if the completion time of the detailed trajectory plan is sufficiently different from the estimate, detailed trajectories can be planned for several of the task allocations with the lowest estimated completion times. The task allocation can then be performed using these actual completion times (shown by loop A in Fig. 4-1).

This strategy also casts the task allocation and detailed trajectory planning problems in a form that allows parts of the computation to be distributed to parallel platforms, as shown in Fig. 4-2. The process of estimating costs is independent for each vehicle, so they can be performed separately. The detailed trajectory design for each vehicle can be similarly distributed. The parallel platforms could be processors on board the UAVs, or could be several computers at a centralized command and control facility. Having described how the steps in the algorithm are related, methods for performing them will be described next.

4.3.2 Finding Feasible Permutations and their Costs

This section presents a detailed description of the process for developing a list of feasible task assignments, finding approximate finishing times for each task assignment, and pruning the list. This step accepts the aircraft starting states $\mathbf{S}_0$, capabilities $\mathbf{K}$, obstacle vertex position $\mathbf{Z}$, and waypoint positions $\mathbf{B}$. The algorithm also accepts two upper boundaries that can help to prune unfavorable permutations: $n_{\max}$ specifies the maximum number of waypoints that a UAV can visit on its mission, and $t_{\max}$ specifies the maximum time that any UAV can

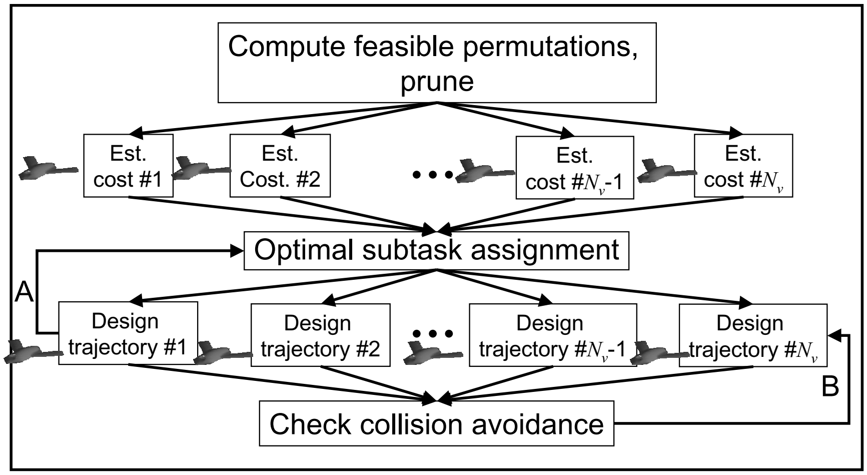

Compute feasible permutations, prune
Est. cost #1
Est. Cost. #2
Est. cost #N_v-1
Est. cost #N_v
Optimal subtask assignment
A
Design trajectory #1
Design trajectory #2
Design trajectory #N_v-1
Design trajectory #N_v
B
Check collision avoidance

Fig. 4-2: Distributed Task Assignment and Trajectory Planning Algorithm

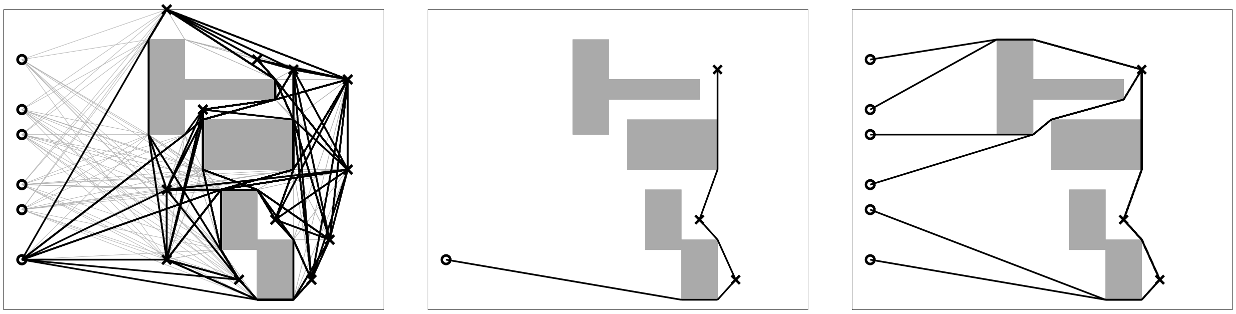


Fig. 4-3: Visibility Graph and Shortest Paths for the Allocation Problem. Left plot shows visibility graph and shortest paths between UAV 6 and all waypoints. Center plot shows shortest path for UAV 6 over one combination of waypoints. Right plot shows shortest paths for all UAVs over same combination of waypoints

fly on its mission. In order to further reduce the decision space, the algorithm produces, for each UAV and each combination of $n_{\max}$ or fewer waypoints, only the order in which to visit the waypoints that gives the shortest finishing time.

The steps in this algorithm are listed in Algorithm 3, and are depicted in Fig. 4-3, in which a fleet of UAVs (shown with $\circ$) must visit a set of waypoints (shown with $\times$). The first step is to find the visibility graph between the UAV starting positions, waypoints, and obstacle vertices. The visibility graph is shown in the left plot with grey lines. Next, UAV 6 is considered, and the visibility graph is searched to find the shortest paths between its starting point and all waypoints, as shown in the plot at left with black lines. In the center plot, a combination of three waypoints has been chosen, and the fastest path from UAV 6's starting position through them is shown. The order of arrival for this combination is found by forming all possible permutations of the unordered combination of waypoints, then summing the distance over the path associated with each order-of-arrival from UAV 6's starting point. The UAV is assumed to fly this distance at maximum speed, and the order-of-arrival with the shorted associated finishing time is chosen. In the plot at right, the fastest path to visit the same combination of waypoints is shown for each vehicle. Note that the best order-of-arrival at these waypoints is not the same for all vehicles.

Pseudocode for this algorithm is given in Alg. 3. It produces three sets of data which fully describe all permutation of waypoints for all vehicles, and are provided to the allocation formulation. These are the $\mathbf{P}$, whose element P_{dvwp} entry is 1 if the d^{th} destination visited by permutation p for vehicle v is waypoints w and 0 if not; $\mathbf{V}$, whose V_{wvp} entry is 1 if waypoint w is visited by permutation p for vehicle v, and 0 if not; and $\mathbf{T}$, whose T_{dvp} entry is the time at which the d^{th} destination on vehicle v's p^{th} permutation. For permutations of fewer than $n_{\max}$ waypoints, $T_{\mathrm{n_{max}vp}}$ assigned the time that vehicle v reaches its last waypoint. All of the permutations produced by this algorithm are guaranteed to satisfy the associated UAV's capabilities.

In this algorithm, finding the shortest distance between a set of waypoints and starting points on line 1 would be performed by finding the visibility graph between the points and vertices of obstacles, then applying a shortest path algorithm, as described in Chapter 2. This approach applied Dijkstra's Single Source Shortest Path Algorithm, since only one goal was considered. Because the distance between all waypoints is required here, the Floyd-Warshall All-Pairs Shortest Path algorithm [6] is appropriate. Note that the iterations through the "**for** loop" between lines 2 and 25 of Algorithm 3 are independent, and can be distributed to parallel processors. The corresponding matrices from each processor can then be combined and passed onto the next stage in the algorithm, the task allocation problem.

```
Find distance between all waypoint pairs $(i, j)$ as $D(i, j)$ using $\mathbf{Z}$ and $\mathbf{B}$;
for all UAVs $v$ do
    p:=1;
    for all numbers $n_\mathrm{c}$ of waypoints to visit, $n_\mathrm{c} := 1, \ldots, n_\mathrm{max}$ do
        Find distance $d(i)$ between start point of UAV $v$, all waypoints $i$ using $\mathbf{S}_0$, $\mathbf{Z}$, and $\mathbf{B}$;
        for all combinations $\mathcal{C}$ of $n_\mathrm{c}$ waypoints that $v$ is capable of visiting do
            for all permutations $i$ of waypoints $[w_1, \ldots, w_{n_\mathrm{c}}]$ in $\mathcal{C}$, with $i := 1 \ldots n_\mathrm{c}!$ do
                $T'_\mathrm{1i} := d(w_1)/v_\mathrm{max}$;
                $P'_\mathrm{1w_1i} := 1$;
                for $d := 2 \ldots n_\mathrm{c}$ do
                    if $T'_\mathrm{(d-1)i} > t_\mathrm{max}$ then
                        go to next $i$;  \\ Permutation is too long
                    end if
                    $T'_\mathrm{di} = T'_\mathrm{(d-1)i} + D(w_\mathrm{d-1}, w_\mathrm{d})/v_\mathrm{max}$;  \\ Cumulative time from start
                    $P'_\mathrm{dw_di} = 1$;
                end for
                $\mathrm{T}_{\mathrm{n_{max}i}} = \mathrm{T}_{\mathrm{n_ci}}$
            end for
            $V_\mathrm{wvp} = 1 : \forall\ w = [w_1, \ldots, w_{n_\mathrm{c}}]$;
            $i_\mathrm{min} = \mathrm{argmin}_i\ T'_{n_\mathrm{c}\mathrm{i}}$;  \\ Choose the fastest permutation.
            $T_\mathrm{dvp} = T'_{\mathrm{d}i_\mathrm{min}} : \forall\ d = [1, \ldots, n_\mathrm{c}]$;
            $P_\mathrm{dvwp} = P'_{\mathrm{dw_d}i} : \forall\ d = [1, \ldots, n_\mathrm{c}], \forall\ w = [w_1, \ldots, w_{n_\mathrm{c}}]$;
            $p \leftarrow p + 1$;
        end for
    end for
end for
```

Algorithm 3: Algorithm for finding shortest paths between waypoints

4.3.3 Task Allocation

The previous section outlined a method of rapidly estimating completion times for individual vehicles for the possible waypoint allocations. This section presents a method of selecting which of these assignments to use for each vehicle in the fleet, subject to fleet-wide task completion and arrival timing constraints.

The basic task allocation problem is formulated as a *Multi-dimensional Multiple-Choice Knapsack Problem* (MMKP) [19]. In this classical problem, one element must be chosen from each of multiple sets. Each chosen element uses an amount of each resource dimension, and incurs a cost. The choice from each set is made to minimize the cost subject to multidimensional resource constraints. In the UAV task allocation problem, the choice of one element from each set corresponds to the choice of one of the N_P permutations for each vehicle. Each resource dimension corresponds to a waypoint, and a permutation uses 1 unit of resource dimension i if it visits waypoint i. The visitation constraints are then transformed into constraints on each resource dimension. The completion times are the costs in this problem. Thus, the overall objective is to assign one permutation (element) to each vehicle (set) that is combined into the mission plan (knapsack), such that its cost is minimized and the waypoints visited (resources used) meet the constraint for each waypoint (dimension). The problem can be written as

$$\begin{aligned} &\min_{\mathbf{x}} J_2 = \sum_{\mathrm{v}=1}^{N_V} \sum_{\mathrm{p}=1}^{N_P} T_{\mathrm{n_{max}vp}} x_{\mathrm{vp}} \\ \text{subject to} \quad &\sum_{\mathrm{v}=1}^{N_V} \sum_{\mathrm{p}=1}^{N_P} V_{\mathrm{ivp}} x_{\mathrm{vp}} \geq w_i : \ \forall\, i \ \in \ \{1, \ldots, N_W\} \\ &\sum_{\mathrm{p}=1}^{N_P} x_{\mathrm{vp}} = 1 : \ \forall\, v \ \in \ \{1, \ldots, N_V\} \end{aligned} \tag{4.3}$$

where $n_{\max}$ is the maximum number of waypoints that any UAV can visit, $T_{\mathrm{n_{max}vp}}$ gives the completion time of vehicle v's p^{th} permutation, and x_{vp} is a binary decision variables equal to 1 if vehicle v performs permutation p, and 0 otherwise. The cost in this problem formulation minimizes the sum of the times to perform each selected permutation. The first constraint enforces that waypoint i is visited at least w_i times (typically $w_i = 1$). The second constraint prevents more than one permutation from being assigned to each vehicle. The MMKP formulation is the basic task allocation algorithm. However, modifications are made to the basic problem statement to include additional cost considerations and constraints.

4.3.4 Modified Cost: Total Mission Time

The first modification for the UAV allocation problem is to change the cost. The cost in Eq. 4.2 is a weighted combination of the sum of the individual mission times (as in the

MMKP problem) and the total mission time. The new cost is

$$\begin{aligned}
\min_{x_{\mathrm{vp}}} J_3 &= \bar{t} + \frac{\alpha}{N_V}\sum_{\mathrm{v}=1}^{N_V} T_{\mathrm{n_{max}v}} \\
T_{\mathrm{dv}} &= \sum_{\mathrm{p}=1}^{N_P} T_{\mathrm{dvp}} x_{\mathrm{vp}} \\
\bar{t} &\geq T_{\mathrm{n_{max}v}} : \forall \ \mathrm{v} \ \in \ \{1 \dots N_V\}
\end{aligned} \tag{4.4}$$

where $\bar{t}$ selects the maximum completion time of all UAVs. The solution to the task allocation problem is 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 desired cost (mission completion time).

4.3.5 Timing Constraints

Solving the task allocation as a centralized problem allows the inclusion of complex constraints on when a waypoint is visited. For example, a typical constraint might be that an anti-aircraft defense site at A must be visited at least Δt units of time before another vehicle can proceed to a high value target at waypoint B.

The timing constraints are met by either altering the order in which waypoints are visited, delaying when a vehicle begins a mission, or assigning a loitering time to each waypoint. The formulation presented here introduces a decisions variable $t_{0\mathrm{v}}$ to represent the time at which vehicle v starts, and then executes its mission without delay. To construct the constraint, an intermediate variable t_{w} is introduced for every waypoint w to evaluate the time at which it is visited as

$$\begin{aligned}
&\forall \mathrm{w} \in \{1,2,\dots,N_W\}, \forall \mathrm{d} \in \{1,2,\dots,n_{\mathrm{max}}\}, \forall \mathrm{v} \in \{1,2,\dots,N_V\}, \forall \mathrm{p} \in \{1,2,\dots,N_P\} : \\
&\quad t_{\mathrm{w}} \geq t_{0\mathrm{v}} + T_{\mathrm{dvp}} - M(1 - x_{\mathrm{vp}} P_{\mathrm{dvwp}}) - \epsilon \\
&\quad t_{\mathrm{w}} \leq t_{0\mathrm{v}} + T_{\mathrm{dvp}} + M(1 - x_{\mathrm{vp}} P_{\mathrm{dvwp}}) + \epsilon
\end{aligned} \tag{4.5}$$

where M is a large positive number. The constraints of Eqns. 4.5 combine to enforce an equality relationship $t_{\mathrm{w}} = t_{0\mathrm{v}} + T_{\mathrm{dvp}}$ if vehicle v visits waypoint w as its d^{th} destination of its p^{th} permutation, and are relaxed otherwise. The timing constraint can then be enforced directly in terms of the intermediate variables t_{w} as

$$\forall c \in 1, 2, \ldots, N_C : \sum_{\mathrm{w}=1}^{N_W} R_{\mathrm{cw}} t_{\mathrm{w}} \geq \Delta t_{\mathrm{c}} \tag{4.6}$$

where $R_{\mathrm{cw}} = -1$ if waypoint w must be visited first in the c^{th} constraint, and $R_{\mathrm{iw}} = 1$ if waypoint w must be visited last in the c^{th} constraint. Simultaneous arrival at the same location can be enforced by placing two waypoints at the same location, and applying two arrival time constraints. Time window constraints relative to the start of the mission can be enforced by including only a 1 entry in R_{c} . Note that this formulation does not allow the same vehicle to visit both waypoints unless one of the original permutations met the timing constraint, and that for very tightly constrained problems, the decision space pruning might have to be relaxed to maintain feasibility. Constraints 4.5 and 4.6 are added to the original problem in Eq. 4.3 to form a task allocation problem including timing constraints. The cost must also be altered to include the UAV start times as

$$J_4 = \bar{t} + \frac{\alpha}{N_V} \sum_{\mathrm{v}=1}^{N_V} (t_{0\mathrm{v}} + T_{\mathrm{n_{max}v}}) \tag{4.7}$$

The constraints presented here for delaying individual start times can be generalized to form other solutions to the timing constraint, such as allowing a UAV to loiter at a waypoint before going to the next objective.

4.4 Simulations

4.4.1 Simple Allocation Example

A small problem is first considered to show how the assignment changes when constraints are added. The basic allocation problem includes two UAVs and four waypoints, and is solved using the partially decoupled method described in this chapter.

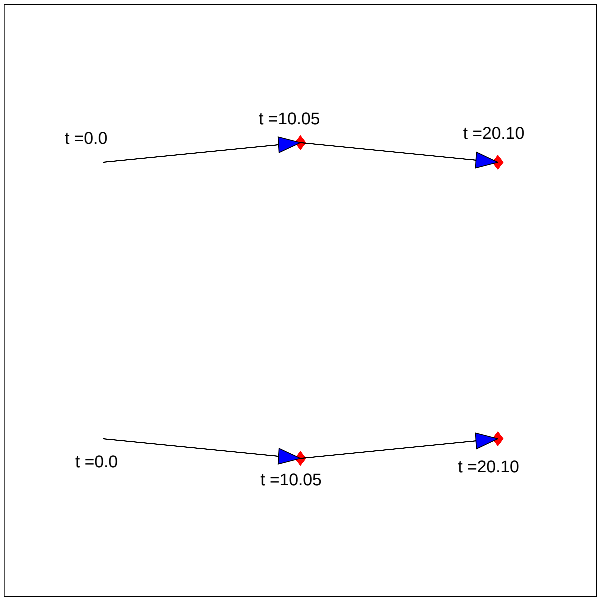
The first scenario, shown in Fig. 4-4, is the basic allocation problem. Each UAV is capable of visiting every waypoint. The solution for this problem is straightforward, with each vehicle visiting waypoints that are close. This allocation is guaranteed to minimize the cost function evaluated using estimated mission completion times which neglect vehicle dynamics. A formulation of this problem that accounts for the vehicle dynamics to find the globally optimal allocation is presented in [24]. In this approach, a large optimization problem is formed to allocate waypoints to vehicles and design detailed trajectories simultaneously. This method was also applied to the allocation problem of Fig. 4-4, and resulted in the same allocation found using the partially decoupled method.

In the second scenario, the ability of the upper vehicle to visit the upper right waypoint is removed, resulting in the allocation shown in Fig. 4-5. The third scenario adds a timing constraint specifying that the upper left waypoint must be visited simultaneously with or after the lower right waypoint. The allocation shown in Fig. 4-6 delays the departure of the upper vehicle, which visits only the upper left waypoint. Fig. 4-7 includes an obstacle in the environment. This results in a modified allocation, reflecting the fact that the partially decoupled approach uses path approximations that avoid obstacles. The same allocation was found with the fully coupled approach for all of these examples.

In each case, the time required to find the allocation using the partially decoupled approach, and then to design detailed trajectories was less than required to solve the fully coupled optimization problem. The times are shown in Table 4.1. The long computation times required by the fully coupled approach emphasize the fact that even for simple problems, the combined allocation and trajectory design problem is extremely computationally demanding.

4.4.2 Large-Scale Comparison to Fully Coupled Approach

In each of the four allocation problem examples presented in Section 4.4.1, the partially decoupled approach quickly gave the same globally-optimal allocation as the fully-coupled,


t =0.0
t =10.05
t =20.10
t =0.0
t =10.05
t =20.10

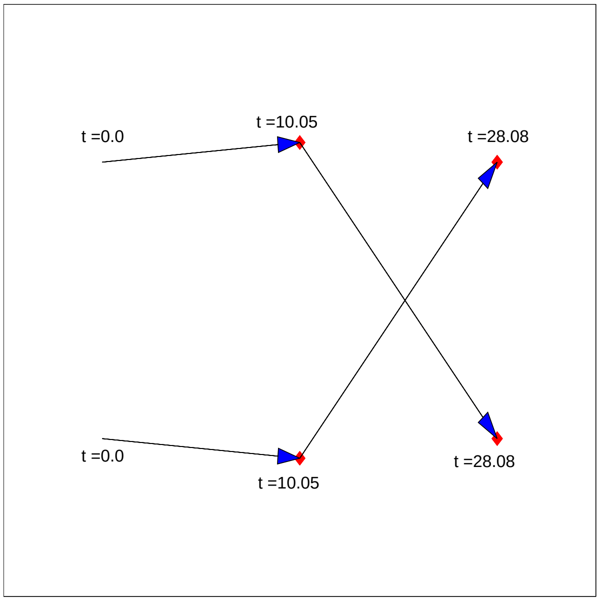
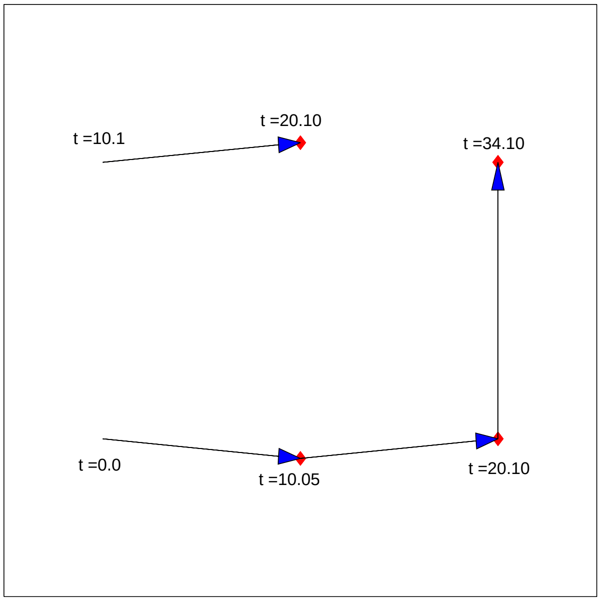

t =0.0
t =10.05
t =28.08
t =0.0
t =10.05
t =28.08

Fig. 4-4: Allocation for two vehicles and four targets with full capabilities, no timing constraints

Fig. 4-5: Allocation for problem of Fig. 4-4, with capability of upper vehicle to visit upper right waypoint removed.


t =10.1
t =20.10
t =34.10
t =0.0
t =10.05
t =20.10

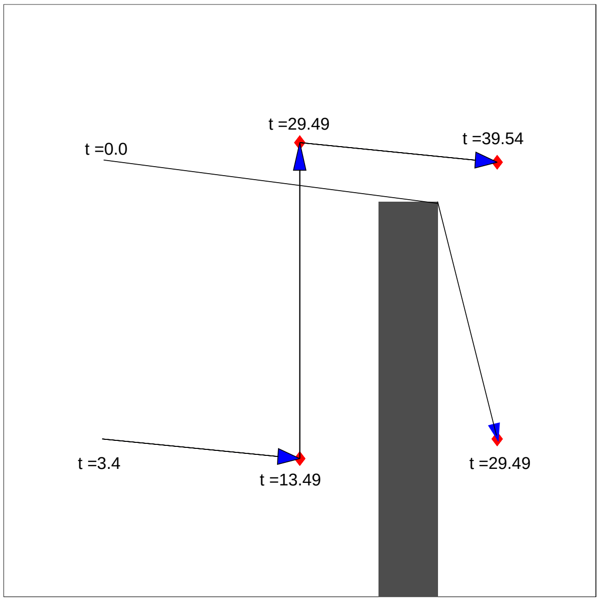

t =0.0
t =29.49
t =39.54
t =3.4
t =13.49
t =29.49

Fig. 4-6: Allocation for problem of Fig. 4-5, with added constraint to visit lower right waypoint before or at same time as upper left.

Fig. 4-7: Allocation for problem of Fig. 4-6, with added obstacle.

Table 4.1: Computation Time for Small Allocation and Trajectory Design Problems

	Partially Decoupled Comp. Time(s)			Fully Coupled
Problem	Allocation	Trajectory Design	Total	Comp. Time(s)
1	0.16	1.43	1.59	12.11
2	0.22	9.34	9.56	26.89
3	0.22	11.13	11.35	146.28
4	0.19	21.05	21.24	981.43

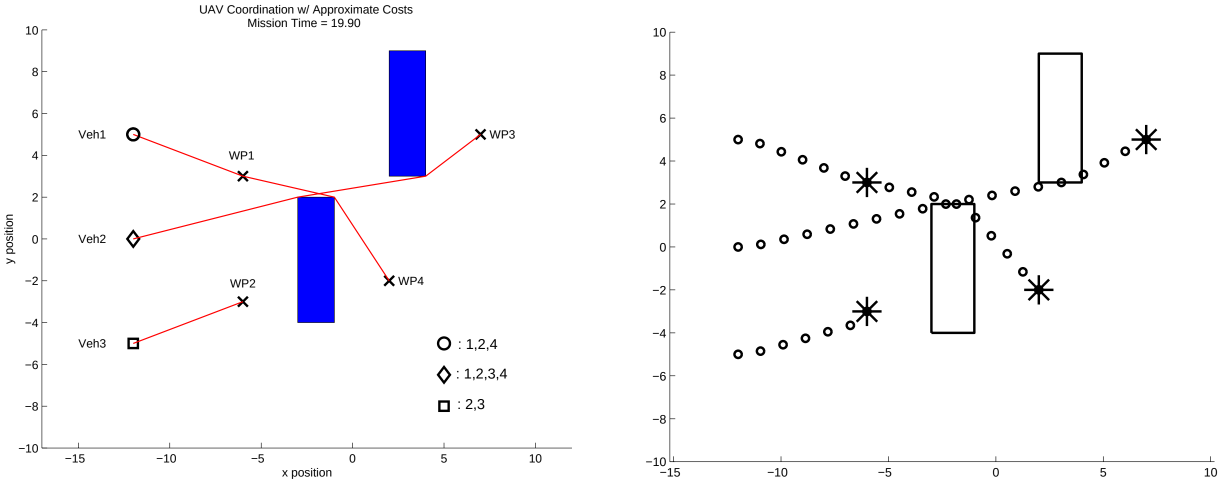


Fig. 4-8: Comparison of Partially Decoupled and Fully Coupled Approaches. The solution found using the partially decoupled approach is shown at left, and that of the fully coupled approach is shown at right.

but computationally demanding formulation. In order to evaluate whether the partially decoupled approach would consistently give the globally optimal allocation, both formulations were applied to a set of 50 problems [25]. Each problems included three vehicles, four waypoints and two obstacles. The starting positions of vehicles, waypoint positions, and obstacle locations were chosen at random.

The solution of one problem from this set with both formulations is shown in Fig. 4-8. With a computation time limit of 10 minutes, the fully coupled formulation was able to find the optimum in 37 problem instances. For these problem instances, the solution times of the two methods are compared in Table 4.2. On average, the partially-decoupled formulation was able to find the allocation and plan detailed trajectories more than 44 times faster than

Table 4.2: Computation Time for Random Allocation and Trajectory Design Problems

Method	Computation Time (s)	
	Mean	Max.
Coupled	165	577
Decoupled		
Assignment	0.54	0.61
Trajectories	3.19	6.51

the fully coupled approach. The partially-decoupled formulation found the global optimum in 36 of the 37 cases. In the one case where it did not, the resulting discrete time trajectories required 1 additional time step for completion. These results indicate that the approach presented in this chapter decouples these problems appropriately, and achieves significant improvement in computation time with negligible loss of performance.

4.4.3 Complex Allocation Problem

A much larger allocation problem was attempted in order to examine the computational effort required by the partially decoupled approach. This large problem includes a fleet of 6 UAVs of 3 different types and 12 waypoints of 3 different types. The UAV capabilities are shown in Fig. 4-9. There are also several obstacles in the environment. Again the objective is to allocate waypoints to the team of UAVs in order to visit every waypoint once and only once in the minimum amount of time. There are no timing constraints in this scenario. The solution is shown in Fig. 4-9. All waypoints are visited subject to the vehicle capabilities in 23.91 time units. This problem was solved in 27 seconds by the partially decoupled approach, but could not be solved in reasonable time by the fully coupled approach.

In order to understand the difficulty of this problem, a "greedy" heuristic was applied to it. This heuristic makes allocation decisions one waypoint at a time. It calculates the increase in the cost function in Eq. 4.2 associated with allocating each waypoint to each capable vehicle. The vehicle-waypoint allocation with the smallest associated increase in the cost is selected. The allocated waypoint is removed from consideration. This procedure is repeated until all waypoints are allocated. The greedy heuristic solved the large scenario

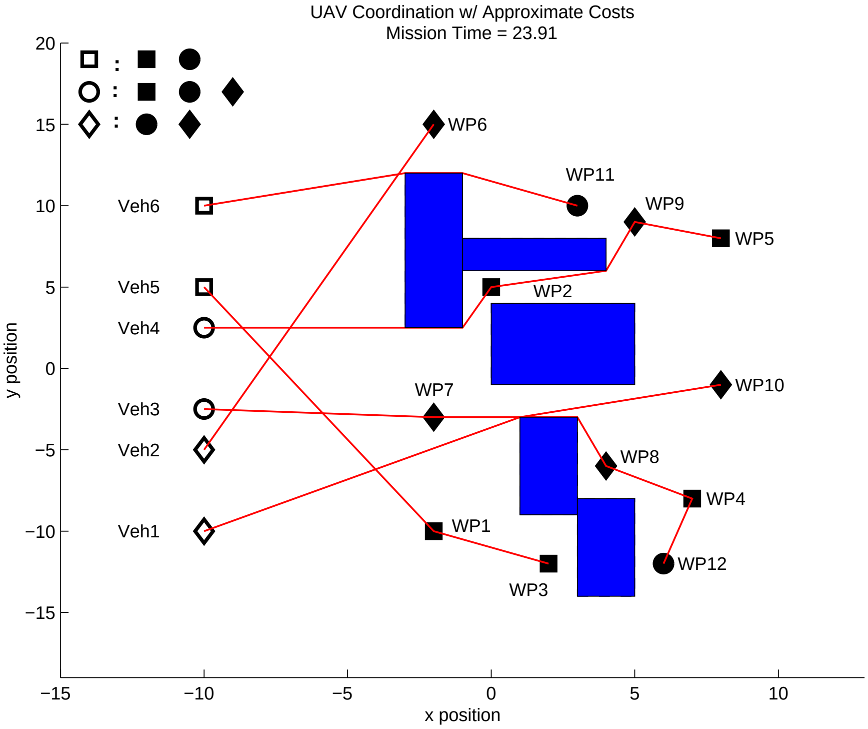


Fig. 4-9: Scenario has three pairs of heterogenous vehicles with 12 waypoints (3 different types). Figure legend shows which vehicles can visit each waypoint. The scenario demonstrates task allocation for a large problem with heterogenous vehicles using the approximate cost method.

shown in Fig. 4-9 with a maximum completion time of 28.20 time units, an increase of 17.9%. In this coordination plan, the last waypoint to be allocated caused a large increase in the completion time. This clearly shows that locally justified decisions do not provide globally optimal fleet coordination plans, and that the MILP-based method presented here provides significantly better results than "greedy" heuristics.

4.5 Conclusions

This chapter presents an approach to the task allocation and detailed trajectory design components of the optimal fleet coordination problem which partially decouples these problems. It efficiently estimates finishing times associated with the different allocation options, and provides this information to the allocation optimization. The allocation is an extension of the

MMKP problem, and is solved as a MILP problem. A small number of detailed trajectories are then designed to perform the allocated tasks.

Results were presented to show that this approach is an appropriate way of decoupling the trajectory design and allocation problems. It is able to solve larger problems than the fully coupled approach with negligible loss in performance, and to produce better performance than a greedy heuristic.

Chapter 5

Maximum Expected Score Allocation

5.1 Introduction

The previous chapter described a method of solving the allocation problem that computes a set of possible waypoint permutations for each vehicle to perform, and applies MILP to select the optimal allocation from amongst them. This chapter extends that approach to scenarios with more sophisticated objectives. This new formulation addresses the uncertainty in the UAV coordination problem by modeling the probability of UAV loss, and exploiting the ability of UAVs to cooperate in order to reduce this probability.

Real world air operations planners employ 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 before a follow-on group attacks higher value targets, preserving their survival. When each UAV has some capability to destroy anti-aircraft defenses and to attack high value targets, designing the mission plan to exploit this cooperation optimally becomes more challenging.

Cooperation is not just desirable, but is in fact crucial to designing successful missions in the heavily defended environments where UAVs are most likely to be used. A successful method of performing the allocation cannot simply assume the mission will always be executed as assumed, given an adversary in the environment who is actively attempting to cause failure. Simulations will be presented to show that ignoring the probability of UAV loss results in mission plans that are likely to fail, and that modeling this probability but ignoring its coupling to each UAV's mission can neglect the fleet's only way to succeed.

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 formulation presented in this chapter approaches this by capturing not only the value of the waypoints that each vehicle visits and of returning the vehicle safely to its base, but also by capturing the probability of these events. In order to maximize mission score as an expectation, this stochastic formulation designs coordination plans which optimally exploit the coupling effects of cooperation between UAVs to improve survival probabilities. This allocation is shown to recover real-world air operations planning strategies, and to provide significant improvements over approaches that are not cognizant of managing attrition.

5.2 Optimization Program Formulations

This chapter present three formulations of the allocation problem that are progressively more cognizant of its stochastic properties. The first is a purely deterministic formulation that assumes no UAV is lost. The second is a deterministic equivalent that models the probability of UAV loss, without taking into consideration the reduction in this probability that comes from destroying anti-aircraft defenses. The third is a stochastic optimization which models both the probability of UAV loss and the ability of cooperation to reduce the probability of loss.

These formulations extend the minimum completion time formulation of Chapter 4. The waypoint permutations are expanded to include the UAVs' landing positions as their final destination. They also apply the constraints presented in Chapter 4 which force the following variables to take their desired values; V_{wvp} is 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} is 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. In order to emphasize distinctions between formulations, this chapter assigns variable names with tildes to probabilities and scores whose calculation neglects the coupling effects between UAV missions, and variable names without tildes to their equivalents whose calculation takes this coupling into consideration.

The results of applying these formulations to the same allocation problem are presented, and the level of anti-aircraft defense threat in the environment is varied to understand its effects. The expected score of each is calculated in section 5.3. The less sophisticated approaches are shown to achieve worse expected scores for simple problems, and to be unable to plan successful missions for more difficult scenarios. The full stochastic formulation is shown to achieve the highest expected score.

5.2.1 Purely Deterministic Formulation

The first modified formulation extends the cost function of Eqn. 4.7 to include a score $\tilde{s}_{\mathrm{dvp}}$ associated with each waypoint in order to balance completion time against the value of waypoints allocated to vehicles

$$\max_{x_{\mathrm{vp}},t_{0\mathrm{v}}} J_5 = -\alpha_1 \bar{t} - \frac{\alpha_2}{N_V} \sum_{\mathrm{v}=1}^{N_V} (t_{0\mathrm{v}} + T_{\mathrm{n_{max}v}}) + \sum_{\mathrm{d}=1}^{n_{\max}} \sum_{\mathrm{v}=1}^{N_V} \sum_{\mathrm{p}=1}^{N_P} \tilde{s}_{\mathrm{dvp}} x_{\mathrm{vp}} \tag{5.1}$$

where $\tilde{s}_{\mathrm{dvp}}$ an input to the allocation problem representing the score of the d^{th} destination of vehicle v on its p^{th} permutation, and the weights α_1 and α_2 are selected to weight completing the mission quickly against planning longer missions that visit more waypoints. The requirement that every point be visited is relaxed, and this formulation neglects the possibility of UAV attrition. This formulation tends to result in "optimistic" plans, in which risk is ignored in favor of high scores.

An example of a mission plan found with this purely deterministic formulation is shown in Fig. 5-1. In this example, the five waypoints at right are allocated to three vehicles that

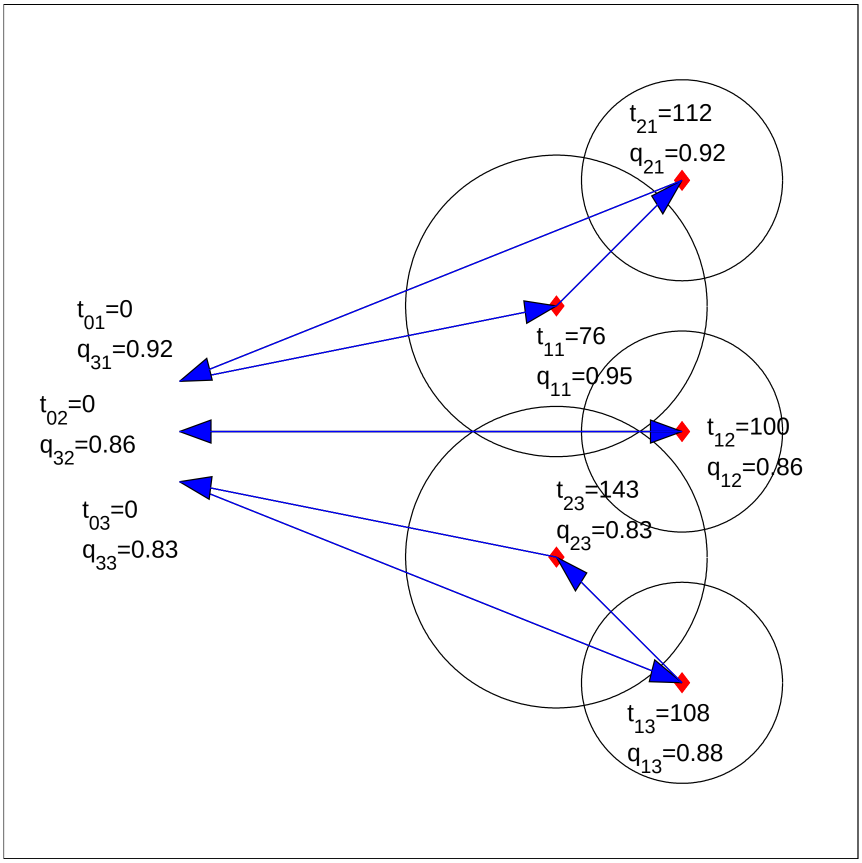


Fig. 5-1: Example Purely Deterministic Allocation. The vehicles start at left and visit waypoints at right. t_{dv} gives the time at which point d is reached on UAV v's mission, including departure time from starting point. q_{dv} is the probability that point d is reached on UAV v's mission. Probability of being shot down is assumed to be proportional length of path in 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.

start at the left. 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. The resulting plan is shown in Fig. 5-1, and resembles that shown in Fig. 4-4, with no UAV path crossings and no delay before any vehicle starts its mission. The weights α_1 and α_2 are chosen sufficiently low that all waypoints are visited, but still encourage fast completion of the mission. The expected score of this mission will be discussed in Section 5.3.

In this work, the probability that a UAV is destroyed is calculated as proportional to the length of its path within the anti-aircraft defense's 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 the performance as the threat level increases.

Under the nominal threat levels, this formulation gave a probability of 0.86 that the high value target at 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. This shows that in well-defended environments, the deterministic formulation plans missions that are highly susceptible to failure.

5.2.2 Deterministic Equivalent of Stochastic Formulation

This second form models the threat that each waypoint poses to UAVs as a fixed quantity, so that destroying it does not decrease the risk to other vehicles. This reduces the problem to multiplying the score 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. This approach allows sophisticated relationships between survival probability and radar exposure to be used. Voronoi diagrams can be used as a basis for path approximations in order to minimize radar exposure, and time and probability values for several different paths can be provided for each ordering of waypoints.

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 $\tilde{q}_{\mathrm{0v}} = 1.0$ for all permutations, and

$$\tilde{q}_{\mathrm{dvp}} \;=\; \tilde{q}_{\mathrm{(d-1)vp}} \prod_{\mathrm{w}=1}^{N_W} \tilde{q}_{\mathrm{dvwp}} \tag{5.2}$$

where $\tilde{q}_{\mathrm{dvwp}}$ 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 Eqn. 5.1 can be modified to use the deterministic equivalent of the score $\tilde{q}_{\mathrm{dvp}}\tilde{s}_{\mathrm{dvp}}$

$$\max_{x_{\mathrm{vp}},t_{0\mathrm{v}}} J_6 = -\alpha_1 \bar{t} - \frac{\alpha_2}{N_V} \sum_{\mathrm{v}=1}^{N_V} (t_{0\mathrm{v}} + T_{\mathrm{n_{max}v}}) + \sum_{\mathrm{d}=1}^{n_{\mathrm{max}}} \sum_{\mathrm{v}=1}^{N_V} \sum_{\mathrm{p}=1}^{N_P} \tilde{q}_{\mathrm{dvp}} \tilde{s}_{\mathrm{dvp}} x_{\mathrm{vp}} \tag{5.3}$$

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. Example allocation plans from the deterministic equivalent formulation are shown in Fig. 5-2. 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. This occurs when the contribution to the expected score of visiting the remaining waypoints is offset by the decrease in expected score of doing so due to a lower probability of surviving to return. The ability to reduce risk through cooperation can be captured by evaluating the actual risk during optimization as a function of the waypoint visitation precedence.

5.2.3 Stochastic Formulation

This section describes a stochastic optimization formulation in which the expectation of score is maximized. This optimization will be shown to exploit phasing by attacking the anti-aircraft defenses before the high value targets, and to preserve the survival of the vehicle which visits the high value target.

In order to determine whether an anti-aircraft defense is in operation while a vehicle flies within its original range, the waypoint visitation precedence is evaluated. If the time that

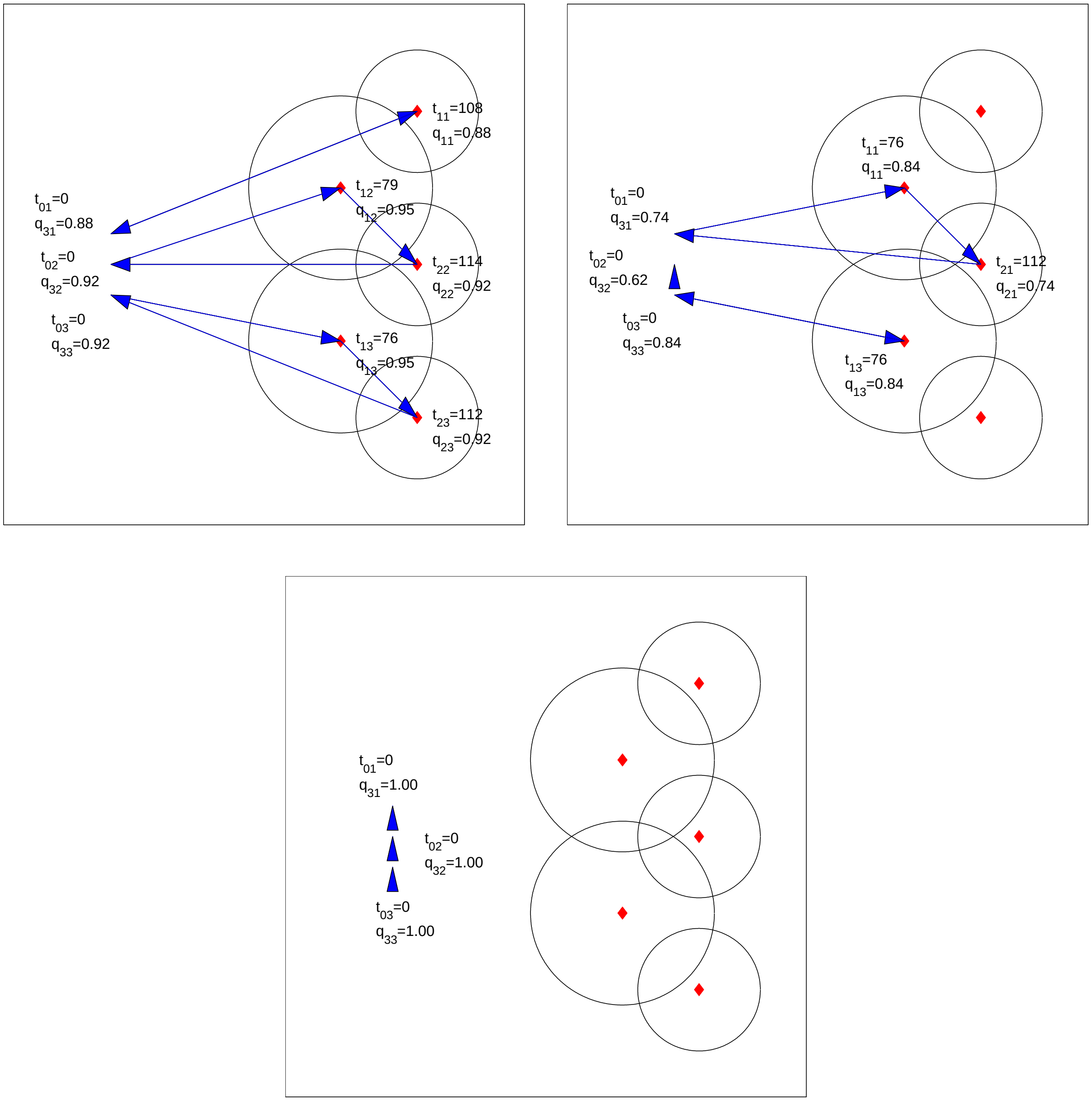

$t_{11}=108$
$q_{11}=0.88$
$t_{12}=79$
$q_{12}=0.95$
$t_{01}=0$
$q_{31}=0.88$
$t_{02}=0$
$q_{32}=0.92$
$t_{22}=114$
$q_{22}=0.92$
$t_{03}=0$
$q_{33}=0.92$
$t_{13}=76$
$q_{13}=0.95$
$t_{23}=112$
$q_{23}=0.92$
$t_{11}=76$
$q_{11}=0.84$
$t_{01}=0$
$q_{31}=0.74$
$t_{02}=0$
$q_{32}=0.62$
$t_{21}=112$
$q_{21}=0.74$
$t_{03}=0$
$q_{33}=0.84$
$t_{13}=76$
$q_{13}=0.84$
$t_{01}=0$
$q_{31}=1.00$
$t_{02}=0$
$q_{32}=1.00$
$t_{03}=0$
$q_{33}=1.00$

Fig. 5-2: Example Deterministic Equivalent Allocations. Nominal probabilities of destruction at top left, increased by factor of 3 at top right, 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 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 which maximizes the expectation of score keeps the UAVs at their base in order to collect the reward for safe return, and the high value waypoint is not visited.

UAV v begins the leg leading to its d^{th} destination is less that the time that 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

$$A_{\mathrm{dvw}} = 1 \Leftrightarrow t_{0\mathrm{v}} + T_{(\mathrm{d}-1)\mathrm{v}} \leq t_{\mathrm{w}} \tag{5.4}$$

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

$$\tilde{q}_{\mathrm{dvw}} = \tilde{q}_{\mathrm{dvpw}} x_{\mathrm{vp}} \tag{5.5}$$

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, so the actual probability q_{dvw} that vehicle v is not destroyed by an anti-aircraft defense at waypoint w is 1. Otherwise, it is $\tilde{q}_{\mathrm{dvw}}$

$$\begin{aligned} q_{\mathrm{dvw}} &\leq \tilde{q}_{\mathrm{dvw}} + M(1 - A_{\mathrm{dvw}}) \\ q_{\mathrm{dvw}} &\leq 1 \end{aligned}$$

Now the actual probability q_{dv} of reaching each destination can be found by evaluating

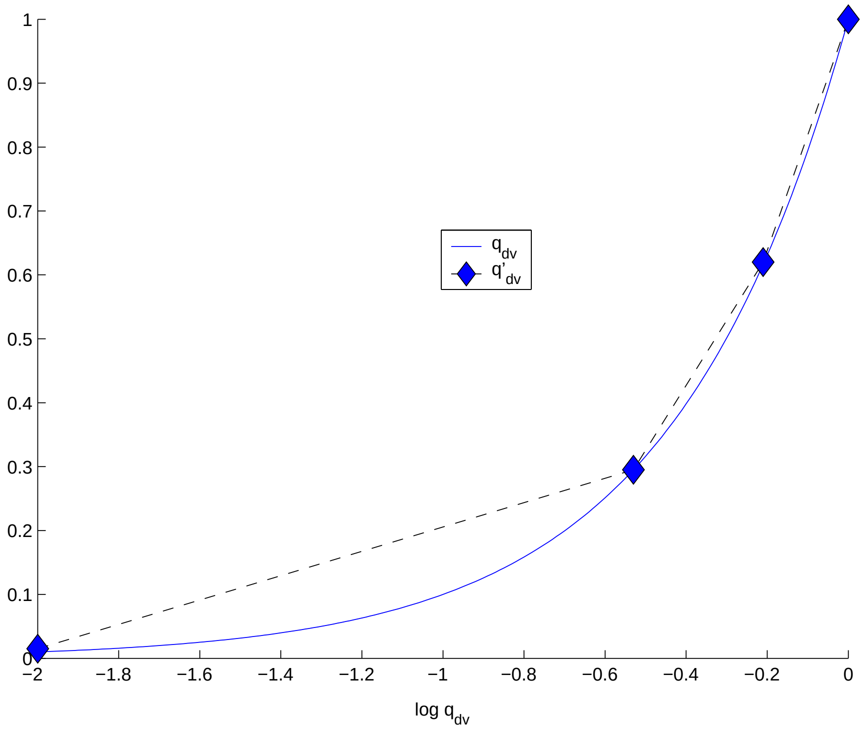


Fig. 5-3: Comparison of Piecewise Linear Approximation q'_{dv} to q_{dv} vs. $\log q_{\mathrm{dv}}$. Note that the approximation is very accurate in the range of probabilities of interest where $q_{\mathrm{dv}} \geq 0.3$.

Eqn. 5.2 in terms of the actual probability of surviving each anti-aircraft defense q_{dvw} as

$$q_{\mathrm{dv}} = q_{\mathrm{(d-1)v}} \prod_{\mathrm{w}=1}^{N_W} q_{\mathrm{dvw}} \tag{5.6}$$

where again, $d = 0$ corresponds to the vehicle's starting position and $q_{0\mathrm{v}} = \tilde{q}_{0\mathrm{v}} = 1.0$. Because Eqn. 5.6 is non-linear in decision variables q_{dvw} and q_{dv}, it cannot be included directly in the formulation, but can be tranformed using logarithms as

$$\log q_{\mathrm{dv}} = \log q_{\mathrm{(d-1)v}} + \sum_{\mathrm{w}=1}^{N_W} \log q_{\mathrm{dvw}} \tag{5.7}$$

While this 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

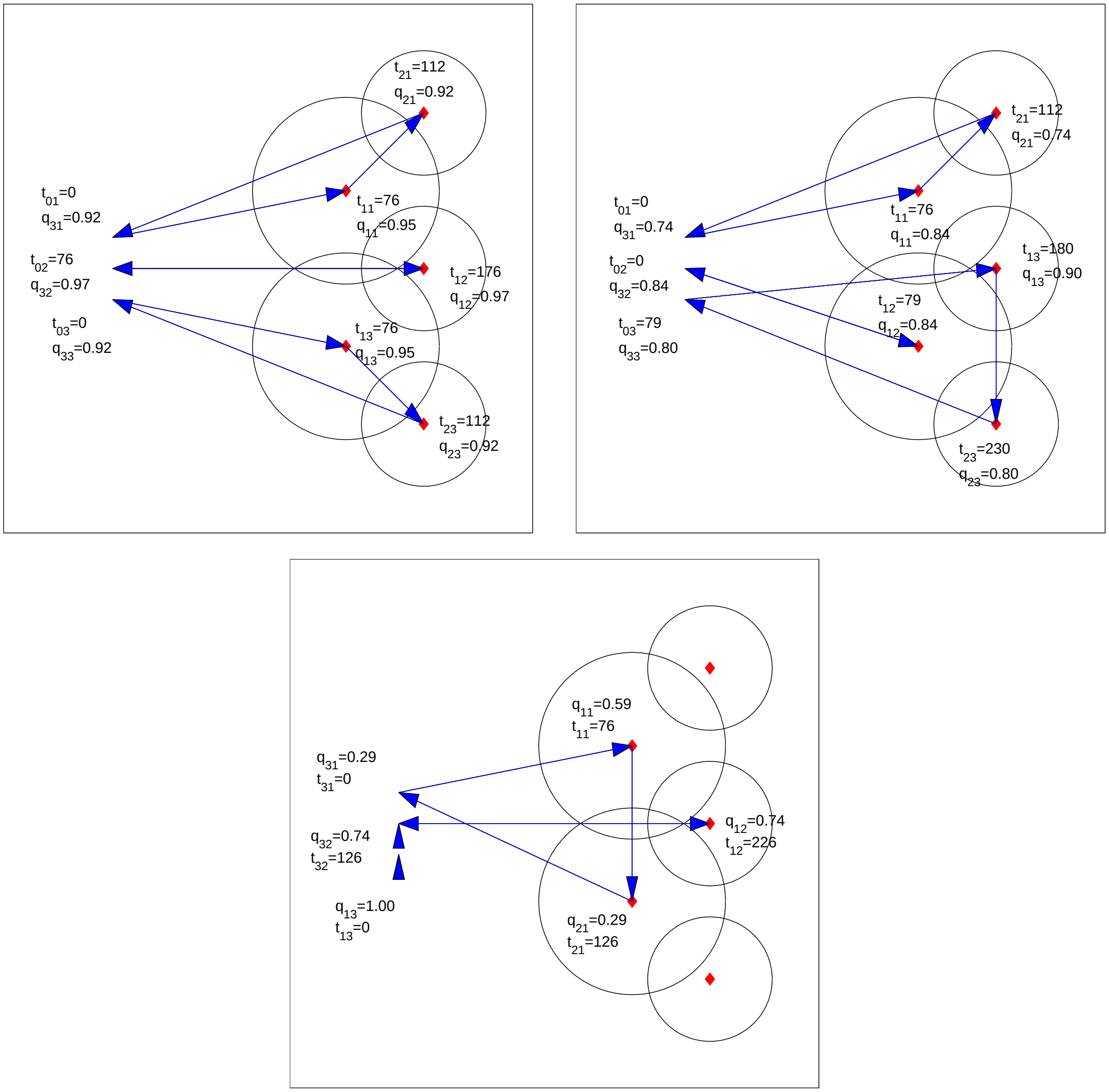

t_{21}=112
q_{21}=0.92
t_{01}=0
q_{31}=0.92
t_{11}=76
q_{11}=0.95
t_{02}=76
q_{32}=0.97
t_{12}=176
q_{12}=0.97
t_{03}=0
q_{33}=0.92
t_{13}=76
q_{13}=0.95
t_{23}=112
q_{23}=0.92
t_{21}=112
q_{21}=0.74
t_{01}=0
q_{31}=0.74
t_{11}=76
q_{11}=0.84
t_{13}=180
q_{13}=0.90
t_{02}=0
q_{32}=0.84
t_{12}=79
q_{12}=0.84
t_{03}=79
q_{33}=0.80
t_{23}=230
q_{23}=0.80
q_{11}=0.59
t_{11}=76
q_{31}=0.29
t_{31}=0
q_{32}=0.74
t_{32}=126
q_{12}=0.74
t_{12}=226
q_{13}=1.00
t_{13}=0
q_{21}=0.29
t_{21}=126

Fig. 5-4: Example Maximum Expected Score Allocation. Nominal probabilities of destruction at top left, increased by factor of 3 at top right, increased by 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.

expected score requires q_{dv}, and this can be recovered approximately as q'_{dv} by raising 10 to the exponent $\log q_{\mathrm{dv}}$ using the piecewise linear function shown in Fig. 5-3. This piecewise linear relationship can be included into a MILP accurately using 3 binary variables, since the exact function is nearly linear in the range of interest where probabilities are above 0.3.

The expectation of the mission score is then found by summing waypoint scores multiplied by the probability of reaching that waypoint. If the score of the d^{th} waypoint visited by vehicles v in its p^{th} permutation is $\tilde{s}_{\mathrm{dvp}}$, then the expectation of the score s_{dv} that will be received from visiting it is

$$\forall p \in \{1, 2, \dots, N_P\} \ : \ s_{\mathrm{dv}} \leq q'_{\mathrm{dv}} \tilde{s}_{\mathrm{dvp}} + M(1 - x_{\mathrm{vp}}) \tag{5.8}$$

and the objective function of the stochastic formulation is

$$\max_{x_{\mathrm{vp}}, t_{0\mathrm{v}}} J_7 = -\alpha_1 \bar{t} - \frac{\alpha_2}{N_V} \sum_{\mathrm{v}=1}^{N_V} (t_{0\mathrm{v}} + T_{\mathrm{n_{max}v}}) + \sum_{\mathrm{d}=1}^{n_{\max}} \sum_{\mathrm{v}=1}^{N_V} s_{\mathrm{dv}} \tag{5.9}$$

The optimal allocations for this problem shown in Fig. 5-4 recovers phasing and preservation of the vehicle that visits the high value target. As the threat level in the environment increases, the upper and lower waypoints are ignored.

5.3 Results

5.3.1 Nominal Environment

After the coordination 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 5.2.3. The resulting expected score, mission completion time, and probability of survival of the three formulations is compared in Table 5.1. The computation time of each formulation is also shown. Note that the expected score of the purely deterministic and stochastic formulations is very different, although the way-

Table 5.1: Results of Several Formulations in Probabilistic Environment with Nominal Threat Levels

Formulation	Expected Score	$\bar{t}$	Probability of Survival UAV 1	UAV 2	UAV 3	Computation Time (s)
Min. Completion Time	251.3	219.5	0.92	0.86	0.83	6.5
Deterministic. Equiv.	263.8	219.5	0.88	0.92	0.92	7.0
Stochastic	273.1	276.5	0.92	0.97	0.92	27.1

point combinations assigned to each vehicle are the same and the allocation differs mainly in timing. This emphasizes the importance of timing of activities.

While some improvement over the completely deterministic formulation is seen in the deterministic equivalent formulation, the stochastic formulation achieves the highest expected score. This formulation also does the best job of protecting the survival of the vehicle that visits the high value target. It is, however, the most computationally demanding formulation.

5.3.2 High Threat Environments

The results of applying all three formulations in high threat environments are shown in Tables 5.2 and 5.3, and indicate that in high threat environments the completely deterministic and deterministic equivalent approaches are incapable of recovering a higher expected score than would be achieved by keeping the UAVs at their base. Also, these two formulations are not capable of designing a plan that is likely to reach the high value target.

5.3.3 Results on Larger Problem

During the solution of the problems presented already, it was noticed that the expected score formulation often quickly found a good answer that was close to optimal, then made very little improvement in the expected score for the rest of its solution time. Since the goal of this research is to develop automated mission planning tools that can apply successful operational strategies in complex environments, finding the optimum provides little advantage over a solution that is very close to it.

Table 5.2: Expected Score in More Threatening Environments. Nominal probabilities of destruction, and probabilities 3 and 7 times higher are considered.

	Expected Score		
Formulation	Nominal	×3	×7
Min. Completion Time	251.3	173.1	81.4
Deterministic. Equiv.	263.7	219.6	150.0
Stochastic	273.15	239.9	208.7

Table 5.3: Probability of Reaching High Value Target in More Threatening Environments. Nominal probabilities of destruction, and probabilities 3 and 7 times higher are considered.

	Probability		
Formulation	Nominal	3	7
Min. Completion Time	0.86	0.57	0.25
Deterministic. Equiv.	0.92	0.74	0.00
Stochastic	0.97	0.9	0.74

To examine this, the expected score formulation was also applied to a large problem with 4 vehicles and 11 targets, and the expected score of the incumbent solution was recorded over time during the optimization process. This optimization was not solved to completion, but achieved a maximum score of about 342 in 60 minutes. However, an incumbent solution with an expected score of about 331 was found in only 18 seconds as shown in Fig 5-5. In this problem, each vehicle can visit 2 waypoints.

This new approach is computationally demanding, which is a result of modeling the ability to influence the probabilities themselves. This is not common in stochastic optimization, but is necessary for planning successful air operations. This formulation can achieve good results quickly. Its loss of 11 units of score is not as significant as its robustness, which comes from managing UAV attrition. This solution can be seen as computationally feasible method of automating UAV mission design, without losing the strategic awareness of the air operations planner. Given that the other formulations presented here have fundamental problems planning missions in threatening environments, the expected score formulation possesses significant advantages.

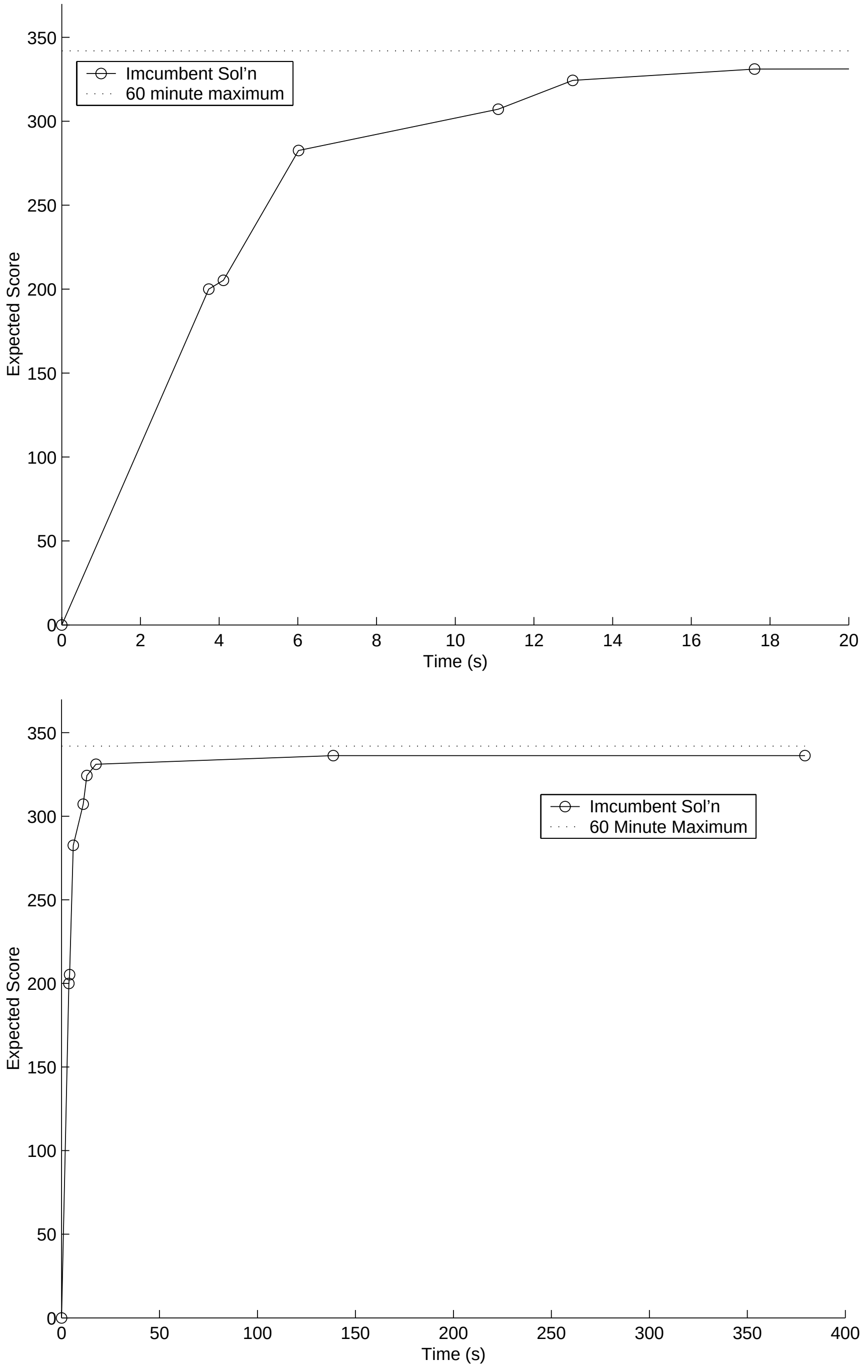


Fig. 5-5: Expected Score of Incumbent Solution vs. Time. The incumbent is compared to the best solution found in 60 minutes. The same problem is being solved in both plots.

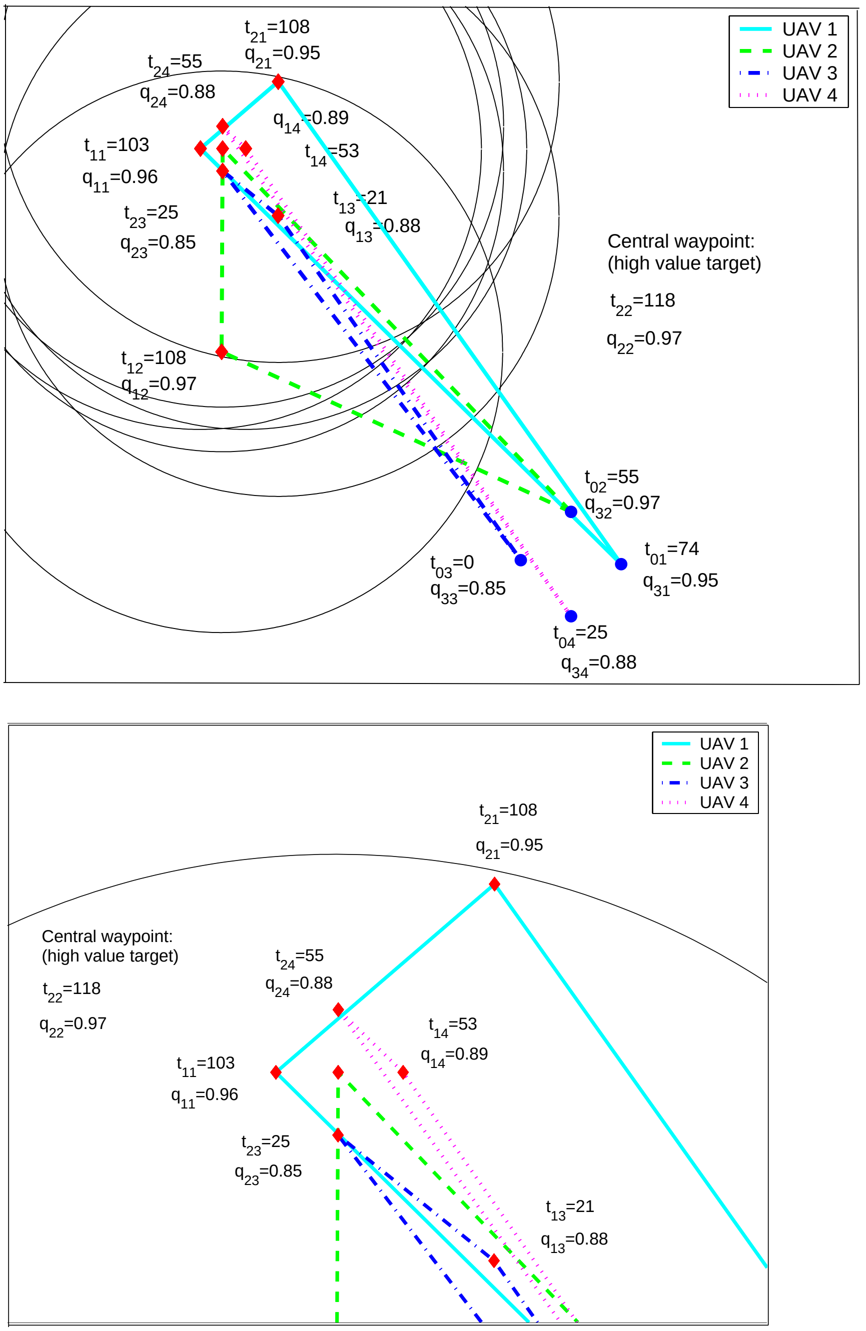


Fig. 5-6: Example Large Allocation Problem. Vehicle starting positions are shown with small circles. The lower plot shows a zoomed in view around the high value target. This solution was found in 18 seconds. Seven of the waypoints represent anti-aircraft defenses, while the 8$^{\text{th}}$, at the center of the tight cluster of waypoints, is a high value target that presents no threat. Note that UAV 2 delays its departure just long enough that 5 of the anti-aircraft defenses have been destroyed. UAV 2 then visits waypoint A at the same time ($t = 108$) as UAV 1 visits waypoint B. Note that the waypoints A and B have been selected as the two that are farthest apart, so that UAV 2 can reach the A without being significantly threatened by B. This preserves UAV 2's survival and minimizes completion time.

Chapter 6

Conclusions

This thesis has presented several approaches to the UAV trajectory design and waypoint allocation problems. These approaches apply MILP to optimize the allocation of a set waypoints to individual UAVs in the fleet, and to optimize the detailed trajectories that the vehicles follow to reach their assigned waypoints. By formulating these problems as MILPs, the solution search process can performed by powerful commercial software such as CPLEX. Such optimization-based approaches could be used as the basis for UAV mission planning tools, enabling UAVs to take on a wider range of tasks and to operate with conventional vehicles in a cohesive manner. The specific contributions of this thesis to the these topics will now be described.

6.1 Contributions to the Trajectory Design Problem

In Chapter 2, a receding horizon formulation of the trajectory design problem was presented. This thesis proposes a novel terminal penalty which can be evaluated based on computationally efficient search of a graph representation of the environment. This choice of terminal penalty accounts for obstacles beyond the planning horizon, and avoids entrapment in concave obstacles. Comparison of the receding horizon trajectory planner to a fixed horizon equivalent shows that the receding horizon planner results in minimal increase in time to reach the goal and significantly reduces the time required to plan a trajectory to the goal. This method is capable of planning trajectories in complex and uncertain environments, and following the resulting trajectories was demonstrated in a hardware testbed.

While this trajectory designer produces good results in practice, it was not proven to be stable. Chapter 3 proved the stability of a modified controller. It presented a method for computing a tree of kinodynamically feasible paths to the goal. A novel use of this path construction technique was made to guarantee that a kinodynamically feasible trajectory exists from the terminal state of each short trajectory segment to the goal. This approach was shown to be capable of planning trajectories in a highly constrained environment, in which the trajectory designer of Chapter 2 failed. The stable receding horizon trajectory designer is also capable of planning long trajectories with a negligible increase in trajectory length, and a moderate increase in computation time.

The trajectory design techniques presented in this thesis have yielded a computationally efficient trajectory designer that provides near-optimal performance and guarantees stability.

6.2 Contributions to the Allocation Problem

In Chapter 4, a decomposition-based approach was presented for the problem of allocating a set of waypoints to the individual vehicles making up a fleet. This approach partially decouples the allocation problem from the detailed trajectory design problem, which would otherwise have to be solved for all possible waypoint permutations in order to exactly evaluate their associated cost. To avoid this computationally intractable task, the allocation is made based on cost values that are rapidly estimated using the same visibility graph search technique presented in Chapter 2. With cost estimates available for the candidate allocations, a MILP is formulated which allocates waypoints to each UAV and minimizes the estimated completion time of the overall allocation, while observing capability, timing and waypoint visitation constraints. Once the allocation has been found, detailed trajectories can be planned for the selected permutations of waypoints for each UAV.

Comparison to a fully coupled, computationally demanding formulation of the combined allocation and trajectory design problem that finds the global optimum was used to evaluate the partially decoupled approach. The partially decoupled approach was found to give the globally optimal allocation for 36 of 37 problem instances, and found an allocation that required 1 extra time step for the other problem instance. More importantly, it required far less computation to solve than the fully coupled formulation, and is able to solve large scale allocation problems for which the global optimum is not available.

This approach was extended to optimize the value of visiting waypoints under the possibility of UAV attrition. This unique stochastic formulation maximizes mission score as an expectation by using sophisticated strategies that are part of real-world air operations planning. These strategies include phasing of mission activities, creating corridors through anti-aircraft defenses, and protecting the survival of aircraft that visit high value targets. The expected score maximization formulation was demonstrated to be capable of planning air operations in high-threat environments, where other formulations that do not model the event of UAV loss or neglect the ability to manage attrition through cooperation break down.

6.3 Future Work

Improving the computational tractability of the expected score formulation would be beneficial to the allocation problem. Exploiting the potential for cooperation between UAVs is a critical aspect of mission planning, and the formulation presented here provides a route to the optimal coordination plan. While the use of pre-compiled waypoint permutations presented in Chapter 5 does yield some reduction in computation, the combination of the expected score formulation presented here and branch-and-bound MILP solution techniques still does not yield solutions quickly enough to be practical for real-time mission planning. Approximate techniques such as tabu search might yield answers with acceptable performance and computation levels.

The architecture of the UAV coordination and control system is an important research topic. The approaches to the allocation problem presented here all require a large optimization problem for the entire fleet to be solved. To allow replanning during operation, the UAVs must communicate extensively to synchronize a database of their states and the world state. Alternative hierarchies which require less communication and reduced the size of the optimization problems to be solved should be considered.

The advice of an expert in the domain of air operations planning would be very helpful in setting research goals, and evaluating the important aspects of real-world air operations planning that the optimization formulation should capture.

Bibliography

[1] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.

[2] D.P. Bertsekas and D.A.Castanon. Rollout algorithms for stochastic scheduling problems. In *Proceedings of the IEEE Conference on Decision and Control*, Tampa FL, December 1998.

[3] USAF Scientific Advisory Board. Uav technologies and combat operations. Technical Report Tech. Tep. SAB-TR-96-01, November 1996.

[4] 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.

[5] P.R. Chanler and M. Pachter. Hierarchical control for autonomous teams. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Montreal, August 2001.

[6] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[7] D.Q.Mayne, J.B.Rawlings, C.V.Rao, and P.O.M.Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

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

[9] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modelling Language for Mathematical Programming*. Boyd and Fraser Publishing Company, Danvers, MA, 1993.

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

[11] A. Jadbabaie, J. Primbs, and J. Hauser. Unconstrained receding horizon control with no terminal cost. In *Proceedings of the American Control Conference*, Arlington, VA, June 2001.

[12] J.L.Ryan, T.G.Bailey, and J.T.Moore. Reactive tabu search in unmanned aerial reconnaisance simulations. In D.J.Medeiros et al., editor, *Procedings fo the 1998 Winter Simulation Conference*, 1998.

[13] T. Karatas and F. Bullo. Randomized searchs and nonlinear programming in trajectory planning. *Proceedings of the IEEE Conference on Decision and Control*, 2001.

[14] Lydia Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Technical Report CS-TR-94-1519, 1994.

[15] T.W.McClain K.B.Judd. Spline based path planning for unmanned air vehicles. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, August 2001.

[16] 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*, November 1999.

[17] S. LaValle and J. Ku. Randomized kinodynamic planning, 1999.

[18] 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.

[19] M. Moser, D. Jokanovic, and N. Shiratori. An algorithm for the multidimensional multiple-choice knapsack problem. *IEICE Trans. Fundamentals*, E80-A(3):582–589, March 1997.

[20] Nick Pohlman. Estimation and control of a multi-vehicle testbed using gps doppler sensing. Master's thesis, Massachusetts Institute of Technology, 2002.

[21] M.Pachter P.R.Chandler, S.Rasmussen. Uav cooperative path planning. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Denver, CO, August 2000.

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

[23] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science*, pages 421–427. IEEE, 1979.

[24] A. Richards, J. Bellingham, M. Tillerson, and J. How. Co-ordination and control of multiple uavs. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*. AIAA, 2002.

[25] A. Richards, J. Bellingham, M. Tillerson, and J. How. Coordination and control of multiple uavs. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Monterey, CA, Aug 2002.

[26] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the American Control Conference*, Anchorage, AK, May 2002.

[27] A. Richards, J. How, T. Schouwenaars, and E. Feron. Plume avoidance maneuver planning using mixed integer. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*. AIAA, Aug 2001.

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

[29] 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.

[30] C. Schumacher, P.R. Chandler, and S. Rasmussen. Task allocation for wide area search munitions via network flow optimization. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Montreal, Canada, August 2001.

[31] 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*, Anchorage AK, May 2002.

[32] J. Tierno. Distributed autonomous control of concurrent combat tasks. In *Proceedings of the American Control Conference*, Arlington, VA, June 2001.

[33] H. P. Williams and S. C. Brailsford. *Advances in Linear and Integer Programming*, chapter Computational Logic and Integer Programming, pages 249–281. Clarendon Press, 1996.

[34] J. M. Wohletz, D.A.Castanon, and M.L.Curry. Closed-loop control for joint air operations. In *Proceedings of the American Control Conference*, Arlington VA, June 2001.