Examination of Planning under Uncertainty Algorithms for Cooperative Unmanned Aerial Vehicles

by

Rikin Bharat Gandhi

B.S. Computer Science,
Carnegie Mellon University, 2003

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2005

Author _____
Department of Aeronautics and Astronautics
January 21, 2005

Certified by _____
Lee C. Yang, Ph.D.
Charles Stark Draper Laboratory
Technical Supervisor

Certified by _____
Nicholas Roy
Professor of Aeronautics and Astronautics
Thesis Advisor

Accepted by _____
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

[*Except for this sentence, this page intentionally left blank*]

Examination of Planning under Uncertainty Algorithms for Cooperative Unmanned Aerial Vehicles

by

Rikin Bharat Gandhi

Submitted to the Department of Aeronautics and Astronautics
on January 21, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

ABSTRACT

Cooperation is essential for numerous tasks. Cooperative planning seeks actions to achieve a team's common set of objectives by balancing both the benefits and the costs of execution. Uncertainty in action outcomes and external threats complicates this task. Planning algorithms can be generally classified into two categories: exact and heuristic. In this thesis, an exact planner, based on Markov decision processes, and a heuristic, receding horizon controller are evaluated in typical planning problems. The exact planner searches for an optimal policy with global contingencies, while the heuristic controller sequentially approximates the global plans over local horizons.

Generally, the two planners trade mission and computational performance. Although the results are limited to specific problem instances, they provide characterizations of the algorithms' capabilities and limitations. The exact planner's policy provides an optimal course of action for all possible conditions over the mission duration; however, the algorithm consumes substantial computational resources. On the other hand, the heuristic approach does not guarantee optimality, but may form worthy plans without evaluating every contingency.

On a fully-observable battlefield, the planners coordinate a team of unmanned aerial vehicles (UAVs) to obtain a maximum reward by destroying targets. Stochastic components, including UAV capability and attrition, represent uncertainty in the simulated missions.

For a majority of the examined scenarios, the exact planner exhibits statistically better mission performance at considerably greater computational cost in comparison to the heuristic controller. Scalability studies show that these trends intensify in larger missions that include increasing numbers of UAVs and targets. Additionally, sensitivity trials are used to capture each algorithm's robustness to real world planning environments where planners must negotiate incomplete or inaccurate system models. The mission performances of both methods degrade as the quality of their system models worsen.

Technical Supervisor: Lee C. Yang
Title: Senior Member of Technical Staff
The Charles Stark Draper Laboratory, Inc.

Thesis Advisor: Nicholas Roy
Title: Assistant Professor
Department of Aeronautics and Astronautics

[*Except for this sentence, this page intentionally left blank*]

DEDICATION

In January 1963, MIT/Draper fellow Edwin "Buzz" Aldrin submitted his doctoral thesis in dedication "to the crew members of this country's present and future manned space programs. If only I could join them in their exciting endeavors!" Six years later on July 20, 1969 he leaped mankind onto the lunar surface with Neil Armstrong.

To all those who aspire to explore and soar above our blue origins

[*Except for this sentence, this page intentionally left blank*]

ACKNOWLEDGEMENT

Thank you.

I thank the many family, friends, professors, advisors, employers, co-workers, astronauts, scientists, writers, musicians, artists, athletes, theologians, and dreamers who have supported and inspired me for the journey ahead.

This thesis was prepared at The Charles Stark Draper Laboratory, Inc.

Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Rikin B. Gandhi

January 21, 2005

[*Except for this sentence, this page intentionally left blank*]

Contents

List of Figures

Nomenclature

N	number of identical, unmanned aerial vehicles (UAVs)
$\mathbf{X}_k$	set of participating UAV positions at time t_k
$\mathbf{x}_j(t)$	two-dimensional coordinates of the jth vehicle at time t
M	number of fixed, target sites
$\mathbf{Y}_k$	set of participating target positions at time t_k
$\mathbf{y}_i$	two-dimensional coordinates of the ith target
D	UAV's initial fuel
O	UAV's initial munitions
$max\_strikes$	maximum number of strikes on a single target
R	reward function
R_{max}	maximum reward valuation
T_i	indicator of ith target site's destruction
W_i	value for ith target site
$f_j(t_k)$	available fuel of UAV j at time t_k
$g_j(t_k)$	available munitions of UAV j at time t_k
$z_{ij}(t_k)$	number of strikes that have been executed by UAV j on target i at time t_k
C	constant velocity of the UAVs
$\mathcal{M}$	Markov decision process model
$\mathbf{S}$	set of possible system states
$\mathbf{A}$	set of possible joint actions
P	Markovian state transition model
α	normalized relevance state value
β	discount factor
p	stationary policy function
V_π	expected value of policy p
V^*	optimal value function
p^*	optimal policy
E	Bellman error of a value function
$\mathbf{RP}$	set of receding horizon optimizations
t_k	time of the kth optimization
$\mathbf{u}_k$	joint-heading assignment at time t_k
H_k	length of planning horizon at time t_k
δ_{ij}	relative distance function between vehicle j and target i
$\tilde{q}_{ij}$	normalized relative proximity function
Δ_i	"capture radius" of target i

[*Except for this sentence, this page intentionally left blank*]

Acronyms

DBN	dynamic Bayesian network
FAA	Federal Aviation Administration
GHz	gigahertz
GPS	global positioning system
HMM	hidden Markov model
LP	linear programming
MDP	Markov decision process
NP-hard	nondeterministic polynomial-time hard
POMDP	partially-observable Markov decision process
RAM	random access memory
RH	receding horizon
SPUDD	stochastic planning using decision diagrams
UAV	unmanned aerial vehicle

[*Except for this sentence, this page intentionally left blank*]

1 Introduction

"He who fails to plan, plans to fail"

– Unknown

Everyday experiences are adventures in decision-making. The simple choice of taking a short-cut to work balances the risk of encountering traffic and the cost of greater mileage. A commuter might keep alternate routes in case of congestion, based on intuition, traffic reports, or previous experience. Meanwhile, a more care-free commuter waits until she is in a jam to consider her options. Consider a similar scenario at a grocery store where there is a special sale on cookies. A customer may balance his purchases for the foods he needs with those he desires to fit inside his shopping basket, depending on his food stock at home, pricing, diet, marketing, emotion, weather, etc. While one customer may forgo the sale to restock his supply of milk, another may first visit the cookies aisle and not keep enough space for milk. Decision-makers must select a course of action (or plan) to attain some objective. The pervasiveness of uncertainty in the real world causes some to construct contingencies for off-nominal circumstances. Others delay such decisions until the events actually occur.

1.1 Motivation

The planning problem is complicated in the presence of other decision-makers (or agents) with similar or conflicting goals. This section provides multi-agent planning examples to illustrate the importance of automated decision-making and the difficulty of selecting an "optimal" decision.

1.1.1 Firefighting example

Consider the firefighting scenario that is illustrated in Figure 1-1. The alarm rings at the firehouse for a fire at an old apartment complex. First, the fire chief must judge the strength of the blaze to select assets to send to the scene. The chief must dispatch enough firefighters and trucks to control the current situation, while maintaining adequate resources at the firehouse for other emergencies. Then, the fire truck drivers must choose routes to get to the scene. Their decision will likely be based on distance and anticipated traffic conditions. The drivers may plan alternate routes prior to their departure as contingencies to possible jams. Once the fire trucks arrive at the scene, the chief coordinates his firefighters to efficiently rescue occupants and extinguish the blaze. Following the chief's guidance, each individual firefighter needs to select a specific path within the building to reach his assignment. The fighter might choose intermediate waypoints based on proximity, anticipated danger, or an unexpected event (such has hearing an occupant's call for help).

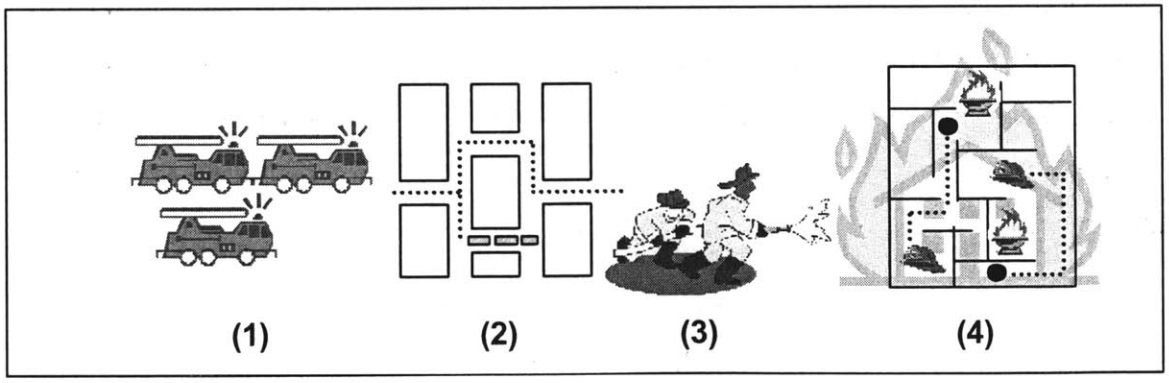

Figure 1-1 Planning for firefighting: (1) allocate fire trucks, (2) plan routes to scene, (3) coordinate firefighters, (4) firefighters choose paths to rescue occupants

The allocation and path-planning flow for the firefighting problem is illustrated in Figure 1-1. The fire chief allocates the fire trucks in (1) and the firefighters in (3). The fire truck drivers determine their routes in (2) and the firefighters choose their paths within the building in (4). Each decision-maker uses a varied set of information resources to make their choice. Each choice impacts the safety of the firefighters and occupants, and their ability to accomplish their rescue and extinguishing objectives. Uncertainty pervades nearly every aspect of this scenario. The fire chief, truck drivers, and firefighters keep contingencies to preclude the effects of unexpected events. Still, the emergency of the fire limits the time that can be spent planning for every possible situation.

1.1.2 Air traffic control example

Now, consider the cooperative task of air traffic controllers. Air traffic controllers are responsible for directing aircraft within the national air space to their destinations. While airlines select high-level flight routes, air traffic controllers provide detailed bearings to aircraft within predefined quadrants of airspace. For each aircraft that enters her airspace, a controller provides pilots with specific navigation instructions and updated weather and traffic conditions. The controllers direct pilots to maintain headings and altitudes for specified durations to fly through particular intermediate (waypoint) locations. The hierarchical planning structure of the aircraft routing is shown in Figure 1-2. The airlines provide (1) aircraft allocation and (2) high-level coordination of flights to city-pair routes. Air traffic control provides pilots with (3) detailed waypoint guidance in each section of airspace. Uncertainties related to weather, traffic, or emergencies have led the Federal Aviation Administration (FAA) to provide controllers with contingency directives for known possible situations. For unanticipated events, the guidelines serve the controllers as a basis for making justified decisions.

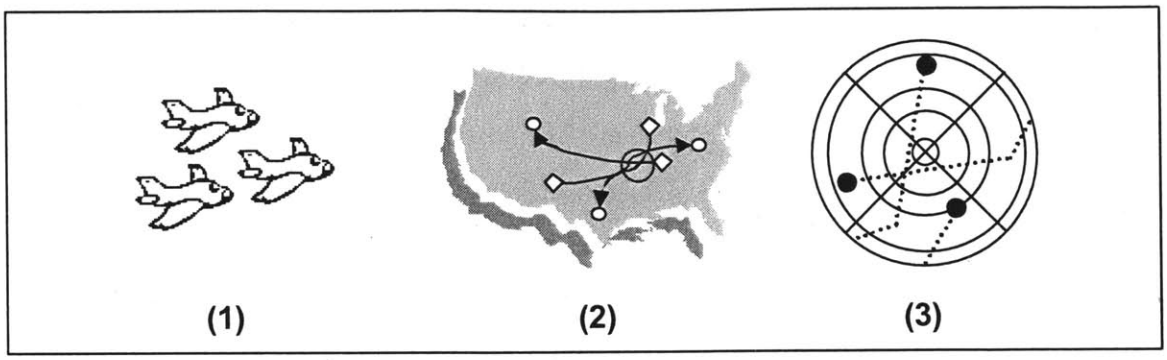

Figure 1-2 Air traffic control routing: (1) airlines allocate aircraft and (2) select city-pair routes, (3) controllers provide pilots with detailed guidance through airspace

1.2 Overview

The decision-making agents in the firefighting and air traffic control examples are cooperative. That is, the decision-makers share common objectives or some subset of them. Decision-makers must coordinate and sequence actions that have immediate and long-term effects. These decisions are complicated by uncertainty. Uncertainty is ubiquitous in the real world. The results of an action, system dynamics, or even the environment itself may be unknown and nondeterministic.

Technology and algorithm advances have enabled researchers to address some of these complex decision-making under uncertainty problems. Scalability to handle large real world planning problems is a continuing challenge. The automated methods that are used to solve such planning problems range from exact to heuristic. In this thesis, an exact planner, based on Markov decision processes (MDPs), which searches for an optimal policy with global contingencies, is compared to a heuristic, receding horizon controller that sequentially selects approximate plans over more localized spaces. The algorithms are examined in various scenarios to characterize their capabilities and limitations. The two approaches are characterized by distinctive computational requirements and behavioral attributes. This thesis focuses on evaluating both classes of automated decision-making methods for cooperative planning under uncertainty problems. The empirical examination benchmarks each method's planning ability in

simulated, mission scenarios based on behavioral performance, computational scalability, and robustness.

1.3 Decision-making for cooperative planning

This section provides the framework for the decision-making problem that is considered in this thesis.

1.3.1 Decision problems

Decision-making problems are based on the interaction of the agent with the world and other agents. The term "agents" simply refers to decision-makers. In the cooperative planning examples, the fire chief, firefighters, airline schedulers, air traffic controllers, and pilots have functional roles as agents. The decision problem is to select actions that achieve these goals. In the previous examples, the fire chief faced the problem of allocating his firefighters to effectively extinguish the fire and rescue occupants, and the air traffic controller had to determine the safest routes for aircraft in her airspace.

The decision problem is dependent on the desired contingency capability. Pre-mission contingency planning provides agents with alternative courses of action in the presence of uncertain events. Oppositely, some planners delay the elaboration of possible situations, and may sequentially make decisions over local spaces.

1.3.2 Decision-makers

Agents' decisions result in actions that affect the system. As shown in Figure 1-3, agents choose actions that influence the state of the system to achieve some immediate or future reward. Typically, this selection is based on the agent's perception (p) of the system state (s). An agent acquires this perception as a function (P) of the system state. In a fully-observable environment, the perception function provides a true view of the system. Otherwise, the system state is partially-observable. An action selection function (A) maps the agent's perception to a particular action choice (a). Once an action has been

performed, a transition function (T) determines its effects on the actual state. The agent may receive some reward (r) based on a reward function (R) of the system state.

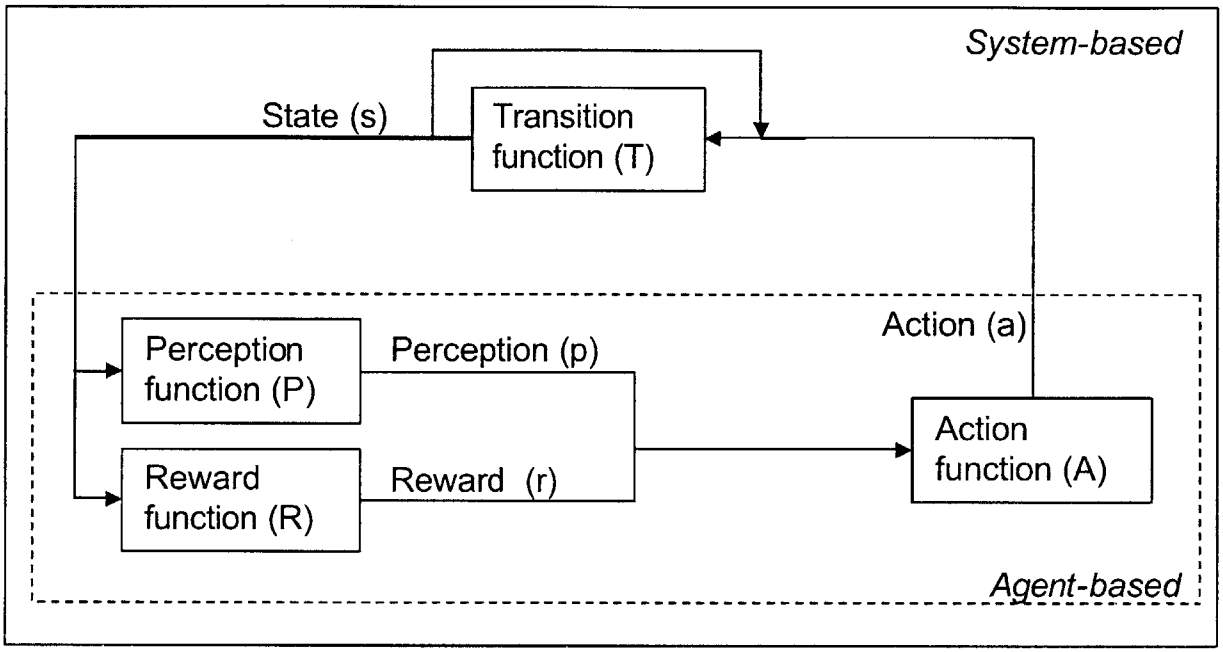

Figure 1-3 Generic view of an agent's interaction with a system. Functionality that occurs on the agent-side is indicated separately from those in the system.

An idealized air traffic control example clarifies this process in Figure 1-4. Suppose an air traffic controller is providing guidance to an aircraft in her assigned airspace. The controller perception (p) of the aircraft's position is based on her radar monitor (P). Assuming the radar monitor is always complete and accurate, the controller can observe the state of every aircraft in her airspace. The controller now directs (a) the aircraft's pilot to a certain heading and altitude. Her guidance causes the pilot to fly (T) to the specified waypoint. Now, the state (s) of the system changes because the aircraft's position has changed. A supervisor might observe the state of the system and recognize (r) the controller for safely managing her airspace. Notice that the aircraft pilots are not viewed as decision-makers in this idealized example. That is, the aircraft are assumed to strictly follow the controller's guidance.

The joint action decisions of all participating agents factor into the overall system dynamics. For example, one air traffic controller might direct an aircraft to move from its current position to a destination in another's airspace. The congestion of a particular quadrant of airspace is then determined by the collective decisions of the nation's air traffic controllers (decision-makers).

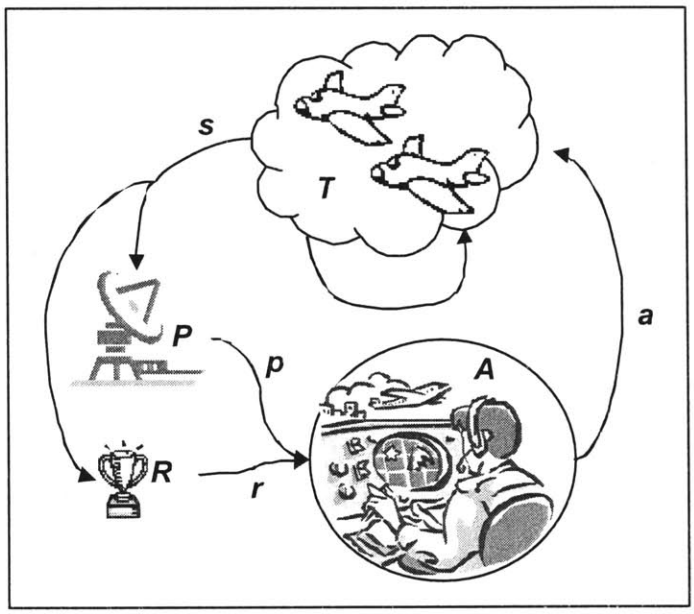

Figure 1-4 Interaction of an air traffic controller "agent" with airspace "system"

1.3.3 Decision systems

Decision-makers select actions to interact with a system. Aspects of the world that are ignored or irrelevant belong to the environment. To produce a plan of action, the decision-maker must have a representation of the system that is to be acted upon.

In the real world, systems continually evolve with time. Each system component is characterized by a set of state variables, and a system's overall state is defined by the aggregation of these component states. For instance, an air traffic controller's quadrant of airspace might be described by one or more state variables, including aircraft positions, squawk codes, weather conditions, etc. The national airspace's overall state is determined by the totality of these state variables for every quadrant that comprises it.

The completeness and accuracy of a planner's system model impacts its behavioral and computational performance. Problems in artificial intelligence are problems of agents interacting with an external world [35]. These interactions can vary across a limitless space. A system's dimensions from a decision-maker's perspective include:

- *episodic v. sequential tasks*

  An agent's task that is independent of past performance and does not affect future objectives is episodic. Sequential tasks depend on an interconnected sequence of decisions.

- *single-agent v. multiple agent*

  Systems may be comprised of one or more agents.

- *discrete v. continuous states and actions*

  Discrete systems can be divided into categories. That is, each perception belongs to a distinct set of possibilities. Continuous systems lack such classifications with only ranges for the perception.

- *fully-observable v. partially-observable states*

  An agent uses its perception of the system to determine its course of action. If each agent has a complete, true view of every state, the system is fully-observable. Partially-observable systems include states that are inaccessible to the agents.

- *Markovian v. non-Markovian dynamics*

  In Markov system, the effects of an action taken in a particular state depend only on that state and not on prior history. In non-Markovian systems, historical data from a previous state is sometimes needed to accurately forecast the future.

- *deterministic v. stochastic v. strategic transitions*

In a purely deterministic system, state transitions are well-defined by a function of the previous state. Strategic systems are essentially deterministic systems with multiple agents. Stochastic systems include elements of uncertainty in state or transitions between states. That is, an action performed in one state might produce a different transition in the same state at a later time. Still, the probabilities that form the basis for these stochastic transitions are fixed over time.

- *synchronous v. asynchronous*

  State transitions in synchronous systems occur whenever an agent performs an action. For synchronous systems with multiple agents, time progresses only when a joint action is performed. That is, the system "waits" for the agents to take action. Synchronous systems do not require actions to be completed in a fixed time window. In asynchronous systems, the system continuously changes without reference to the agents.

- *static v. dynamic*

  Static systems, an analogous concept to synchronicity, do not change while an agent perceives the state of the system or deliberates its next course of action. Time is a critical factor in dynamic systems. Some real world agents have to contend with dynamic, asynchronous systems that advance along a continuum of change.

1.3.4 Decision rewards

Decision-makers have a capitalistic basis for selecting their course of action – maximizing reward. Positive rewards are typically awarded for specific states which satisfy mission objectives. Negative rewards are usually assigned to unfavorable states or costly actions. The number of lives saved and property damages prevented are possible rewards for the firefighters. Similarly, air traffic controllers might view the number of flights guided without incident as a rewarding quantity.

Rewards provide an assessment of an agent's decision-making performance. For rewards to affect an agent's course of action, the agent must have some ability to quantify the reward that has been earned. The ability for an agent to forecast its expected future rewards based on a perception of the current system is an important property of the planning methods examined in this thesis.

1.3.5 Decision planning

In a fully-observable and deterministic system, agents may simply follow a sequence of actions that are set prior to mission execution. This approach depends on complete and accurate models of the system's initial state and dynamics. Suppose all aircraft were known to enter an air traffic controller's airspace at fixed intervals and coordinates. The controller could issue the same guidance to the same aircraft everyday. Unfortunately, the real world is not so predictable. Routes alter, emergencies occur, weather fluctuates, etc. A decision-maker may choose to follow a deterministic approach until a particular event occurs and keep a conditional plan to respond appropriately. Conditional planners prescribe actions that are contingent on the appearance of certain system attributes. For instance, the air traffic controller may provide aircraft with specific routing instructions depending on the visibility conditions of her airspace.

Using stochastic models of uncertainty, policies can elaborate conditional plans to relate action choices with every possible state of the system (stationary) or past events (non-stationary). Decision-makers may use these policies to maximize the expected utility of actions over the mission duration. Stationary policies provide a set of contingency options for systems in which a known set of possible states may occur with some uncertainty. Universal stationary policies, which map the complete set of possible state to action choices, can be difficult to scale.

On the other hand, some re-planning methods segment the mission duration into more manageable planning blocks. For instance, a greedy approach might sequentially select actions that provide immediate gain, rather than forecasting states that might occur over the mission duration. Although this heuristic reduces complexity, it does not guarantee optimality or correctness.

The methodologies of planners that generate stationary policies and those that sequentially re-plan are illustrated in Figure 1-5. Stationary policies formulate universal contingency plans over the entire mission duration to handle uncertainty. On the other hand, a re-planning method may successively plan over shorter time horizons and react to unexpected events. At time step i, this planner selects actions RP_i, which are executed over the particular horizon length. This thesis examines both planners that generate universal stationary policies and those that sequentially re-plan.

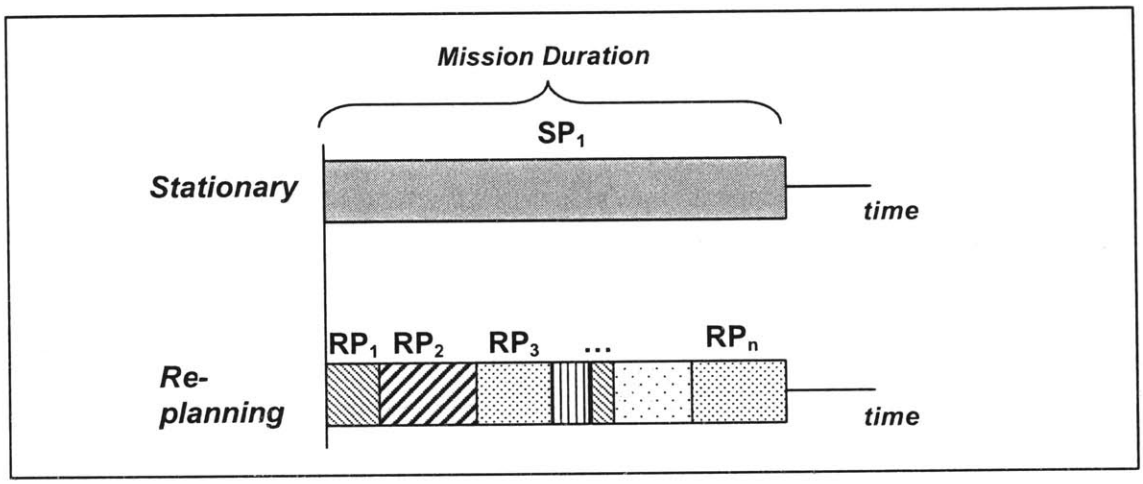

Figure 1-5 Mission planning with stationary policies versus successive re-planning

1.3.6 Cooperation, coordination, and communication

Cooperative agents perform actions in pursuit of achieving the team's common set of objectives. Cooperation is essential for numerous tasks. These include swarms of unmanned aerial vehicles (UAVs) performing military surveillance and reconnaissance, earthquake rescuers working with search dogs and robots to find missing persons, and firefighters extinguishing a blaze. Agents are cooperating if the addition of a new agent increases the productivity of the group or the actions of the agents avoid or solve possible conflicts [25]. Sergiy Butenko proposed a unified framework for cooperation [8]:

1. requires more than one entity,

2. the entities have some dynamic behavior that influences the decision space,

3. the entities share at least one common objective, and

4. entities are able to share information about themselves and the system

Cooperation usually requires some level of coordination. Coordination involves managing an individual agent's local actions to support the team's mission objectives. Some systems utilize a unifying coordinator to direct individual entities as a team, others rely on the participants to cooperatively interact, and still others use a hybrid approach in which a team member is provincially designated as a coordinator. In the first, the coordinator is a centralized planner that sees all, decides all, and tells all. As illustrated in Figure 1-6a, the coordinator observes the system, selects a mutually beneficial course of action for the team, and informs individual participants to interact with the system accordingly. Figure 1-6b depicts the planning structure of a more decentralized approach. Other circumstances might require individuals to coordinate themselves. In these cases, each team member is a decision-maker that fully incorporates the agent attributes shown in Figure 1-3. Additionally, some cooperative systems use a hybrid approach in which a participating agent operates as both a coordinator and a team member. In the event the leading agent is lost, its role can be transferred to an available teammate.

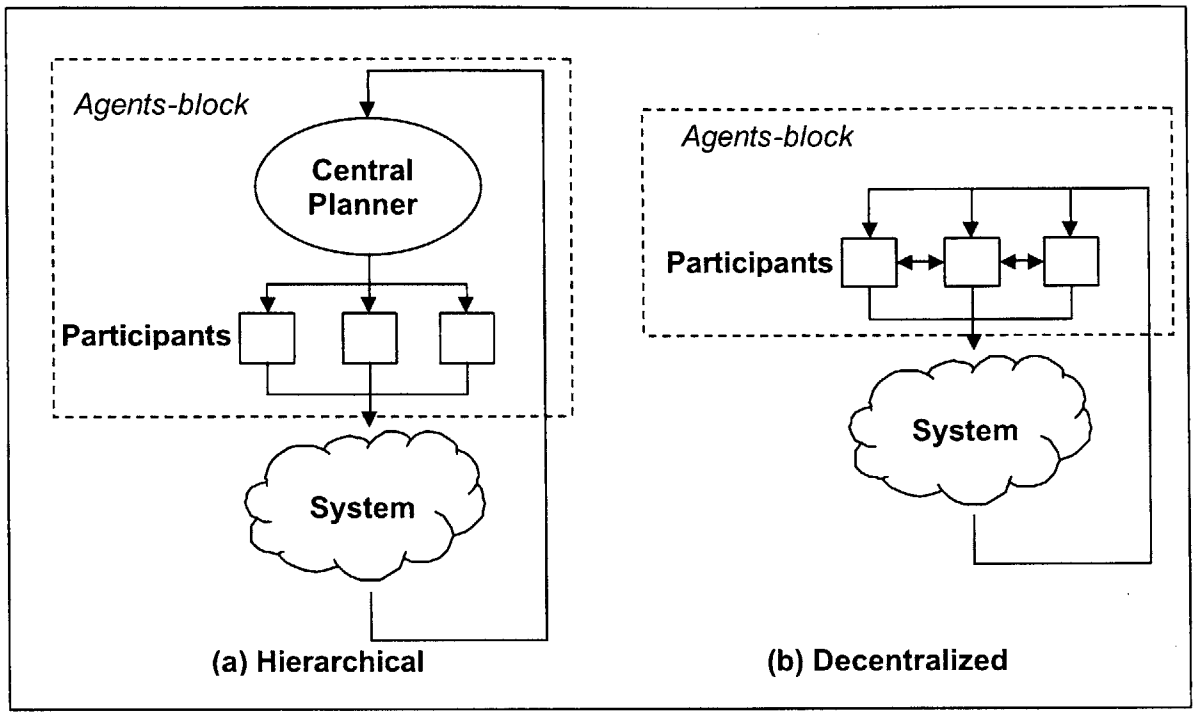

Figure 1-6 Comparison of (a) hierarchical and (b) decentralized approaches to cooperative planning. Entities with agent roles are highlighted.

An important property of positive cooperation is the improvement of some aspect of the goal fulfillment process with each additional team member. That is, the participants achieve their common goal with greater speed, better performance, and/or lesser cost by operating cooperatively rather than individually. Purposes for cooperation include accomplishing tasks that might be impossible to perform alone, improving the productivity of agents, increasing the number of tasks performed within a given time, reducing the time to perform tasks, and improving the use of resources. In the firefighting scenario, the fire chief must determine the most effective number of firefighters to send into a burning building. The chief must choose enough firefighters to extinguish the blaze without unnecessarily risking lives. At first, each additional firefighter contributes positively to rescuing occupants and extinguishing the fire; however, the cooperative value of dispatching additional firefighters increases to a point of diminishing returns. Eventually, the building becomes overcrowded with firefighters that neither speed the mission nor improve the team's ability to succeed. Instead, these extra firefighters represent a liability.

Communication usually emerges as a rational mechanism for cooperation and coordination [35]. Even in fully-observable systems, agents might wish to ensure that their actions positively contribute to the fulfillment of their goals. Communication expands the perceptive capacities of agents and can serve as a fundamental means to distribute tasks and coordinate actions. To this end, hierarchical coordinators exchange state and strategy information with each individual agent, and decentralized team members communicate with each other. In the firefighting scenario, the chief might coordinate his team by receiving firefighter reports, selecting the next course of action, and responding to firefighters with appropriate directions. Appropriate devices for this interaction might be wireless, GPS-enabled handheld computers or simple two-way radios. Now, suppose the firefighters did not have a fire chief. The fire fighters could coordinate themselves to best rescue occupants and quickly extinguish the fire. To successfully cooperate, each firefighter might desire knowledge of the states, actions, and plans of his comrades. By exchanging only the most relevant attributes of their mission, the firefighters support efficient communication and enhance individual decision-making. This information may be transmitted amongst firefighters by computer or radio to allow an individual firefighter to determine his next course of action based on a collective perception of the system state. Of course, neither the chief nor the firefighters really have a complete and accurate view inside the fiery building. De-centralized planning and distributed partially-observable Markov decision processes (POMDPs) research seek to address such concerns.

1.3.7 Thesis structure

The organization of this thesis is as follows:

Chapter 1: An introduction with motivation for problems of cooperative planning under uncertainty. Fundamental concepts and terms that are used throughout the thesis are introduced in this chapter.

Chapter 2: The specific problem of cooperative UAV mission planning is described and formalized. In addition, the strategy for evaluating exact and heuristic planning methods in simulation is discussed.

Chapter 3: An overview is provided for the exact, Markov decision process-based planner and heuristic, receding horizon controller that are investigated in this thesis. Optimization and representation differences of these exact and heuristic approaches are also described. Application examples are drawn from simulated, cooperative UAV scenarios.

Chapter 4: The Markov decision process and receding horizon controllers are simulated in a diverse set of cooperative, multiple-UAV scenarios. Mission and computational performance of the exact planner is weighed against the heuristic controller to determine the strengths and weaknesses of each approach. Scalability studies measure the trends of these performance metrics in scenarios that include increasing numbers of UAVs and targets. Additionally, sensitivity trials capture each algorithm's robustness to real world planning environments where planners must negotiate incomplete or inaccurate system models. The scenarios are meant to empirically examine the trade-offs of global, contingency planning and sequential, local re-planning.

Chapter 5: The main contributions of this thesis are summarized. The final discussion concludes with open problems and future directions.

[*Except for this sentence, this page intentionally left blank*]

2 Problem

"The probable is what usually happens"

– Aristotle

2.1 Objectives

Cooperative planning seeks actions to achieve a team's common set of objectives by balancing both the benefits and the costs of execution. Uncertainty in action outcomes and external threats complicate this task. Planners can be classified into two categories: exact or heuristic. In this thesis, an exact planner, based on Markov decision processes, and a heuristic, receding horizon controller are evaluated in typical planning problems. The exact planner searches for an optimal policy with global contingencies, while the heuristic controller sequentially selects approximate plans over more localized horizons. Generally, the two planners trade mission and computational performance. The presented results are limited to specific problem instances, and provide characterizations of the algorithms' capabilities and limitations in each scenario.

2.2 Overview

Unmanned vehicles, including airborne drones and minesweeping robots, are becoming an increasing feature in the battlefield theatre. Civilian counterparts are also in development for disaster relief, environmental monitoring, and planetary exploration [8]. Unmanned vehicles in current use, such as the Global Hawk, lack significant autonomy and are remotely guided by teams of human operators. This technology is expensive, and restrictive in scalability and range. Recent advances in hardware and artificial intelligence have allowed researchers to consider cooperative control systems that involve multiple autonomous vehicles in dynamic, uncertain environments.

In this thesis, the planning problem is comprised of sets of unmanned aerial vehicles (UAVs) and targets. Reducing both human casualties and cost, UAVs will have a rising role in battlefields of the future. Military UAVs will be used primarily in three classes of missions: surveillance, reconnaissance, and strike.

To evaluate the planning algorithms, the UAVs simulated in this thesis are engaged in a visit-and-destroy mission over an extended battlefield of fixed targets with known positions. As shown in Figure 2-1, a centralized planner hierarchically controls the actions of the UAVs on the battlefield. The planner perceives the current state of the system, which includes the component states of the UAVs and targets, and formulates a cooperative plan to maximize the rewards acquired from destroying targets in the face of uncertainty and constraints. Plans are issued to individual UAVs, which execute the planner's task guidance on the battlefield by moving to particular locations or executing strikes on specific target sites. The UAVs' actions on the battlefield alter the system's state, and the planning cycle repeats. Essentially, the software agent that provides the core planning functionality has three cyclical tasks:

1. perceive the state of the system
2. determine a cooperative course of action
3. provide task guidance to each UAV

The managerial software planner unifies the UAVs as a team by coordinating them cooperatively. This division relies on the assumption that the true state of the system is fully-observable.

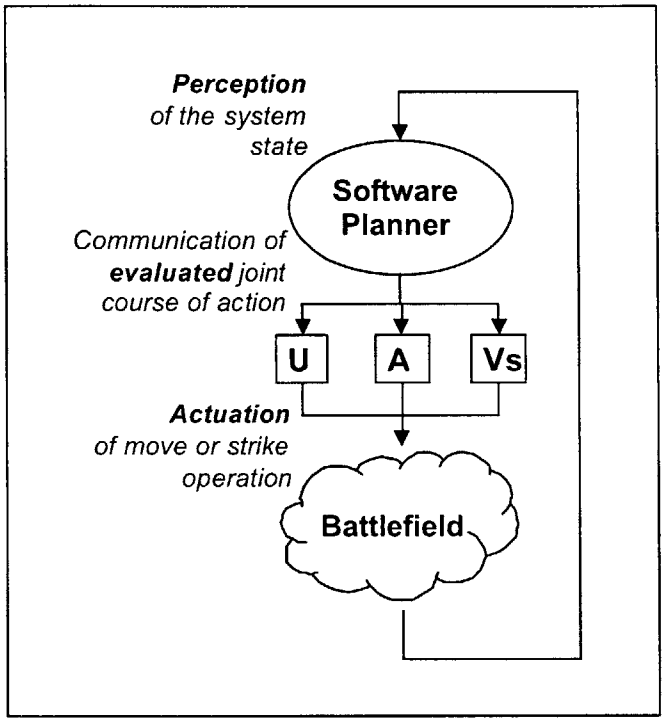

Figure 2-1 Hierarchical planning approach used in cooperative UAV simulation

2.3 Formulation

Before proceeding, the discussion of system states and models should be formalized. This thesis is focused on examining exact and heuristic algorithms for cooperative planning. In the considered scenarios, the UAVs must collectively visit-and-destroy a predetermined set of targets within resource constraints.

The simulated scenarios are extensions of the classic traveling salesperson problem. The traveling salesperson problem attempts to solve the deceptively simple question: given a number of cities and the costs for traveling from one to the other, what is the cheapest roundtrip route that visits each city and then returns to the starting city? Significant caveats that distinguish this problem from the traveling salesperson problem

include the presence of multiple vehicles, non-routing actions, stochastic state transitions, vehicle attrition, and resource constraints.

2.3.1 Simulation scenario

The simulated, cooperative UAV scenarios share similarities with the firefighting and air traffic control examples described in Chapter 1. Indeed, the planning algorithms examined in this thesis can be applied to a diverse set of domains.

The simulation scenario, shown in Figure 2-2, is defined by a two-dimensional mission space in which positions are given by latitude and longitude coordinates. A set of M stationary, target sites $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$ gives the ith target's location as $\mathbf{y}_i \in \Re^2$. In addition, N identical, unmanned aerial vehicles (UAVs) have positions in a set $\mathbf{X}$, where the position of the jth vehicle at time t is denoted as $\mathbf{x}_j(t) \in \Re^2$.

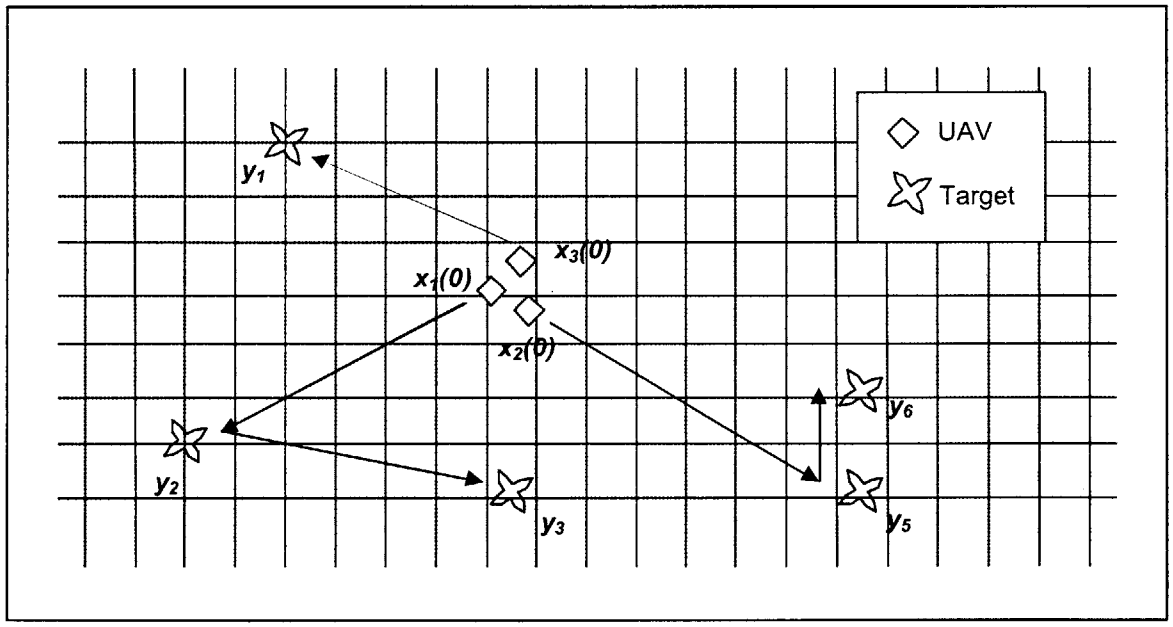

Figure 2-2 Cooperative three-UAV, six-target simulation scenario

The UAVs are initialized at predetermined, airborne locations that belong to the set $\{\mathbf{x}_1(0), \ldots, \mathbf{x}_N(0)\}$. All UAVs are equally equipped and capable. The vehicles are assumed to travel at a constant velocity C throughout the missions. The planner coordinates and

sequences the UAVs' actions to the fixed target sites. Probability distributions describe the UAVs' attrition rates and capability to successfully destroy target sites. Fuels costs for transits between sites are proportional to distance, and strike costs are fixed. Fuel and munitions limit the UAVs and a *max_strikes* constraint bounds the number of attacks performed on a single target. Because of UAV attrition and target destruction, the numbers of participating vehicles and targets are a subset of the original N UAVs and M targets at any given mission time. The mission terminates when all targets are destroyed, every UAV is lost, or the surviving UAVs have expended either their fuel or munitions.

The objective of the cooperative UAV mission is to claim a maximum reward over the mission duration. Based on the number of targets destroyed, the UAVs accumulate a reward $\sum_{i=1}^{M} T_i W_i$, where T_i is a binary variable that indicates whether target y_i is destroyed and W_i is its value. Admittedly, this construction of the reward function biases planning strategies towards destroying targets, while disregarding UAV losses. Although the costs associated with UAV attrition could be incorporated into the reward model, these additions substantially increase the computational requirements of the exact planner evaluated in this thesis and limit the sizes of missions that can be reasonably examined. Still, by modeling the probability of UAV attrition, a planner should avoid vehicle losses to maintain greater opportunities for gaining reward.

2.3.2 Automated solvers

There are two fundamental techniques for tackling the cooperative UAV mission planning problem:

1. exact algorithms to find optimal solutions
2. heuristic algorithms to find acceptable solutions

A critical difference between exact and heuristic planners is the solution that each is intended to provide. The exact planner evaluated in this thesis seeks policies that are optimal in the presence of uncertainty over the mission duration. Moreover, the MDP-

based planner's policy includes contingencies for every possible condition. The heuristic, receding horizon controller collapses this search by sequentially re-planning over localized spaces. In some cases, this locally optimal solution is the global optimum. Otherwise, the solution is suboptimal.

Typically, planners that search for a stationary policy with universal contingencies are computationally limited by the size of the mission domain. Less comprehensive heuristic algorithms tend to be more scalable and can produce provably good results.

2.3.3 System dimensionality

In this thesis, the UAVs have a joint mission to visit-and-destroy a set of targets.

- *sequential tasks*

  Planners select the UAVs' sequential courses of action to maximize the success of this task.

- *multiple agent*

  Simulated scenarios include one or more UAVs to examine the cooperative behaviors produced by each planning approach.

- *discrete states and actions*

  Practical problems may be comprised of a large or infinite number of states and actions. For instance, battlefield positions and fuel have values along a continuous spectrum. These states have been discretized to transform the system into a finite state-space. Fuel quantities are divided into predefined increments, and positions belong to a fixed set of coordinate nodes. The physical health of the UAVs and targets also are represented by Boolean states: alive or dead.

- *fully-observable states*

The UAVs operate in a fully-observable system. At each time step, the planner knows the health and position of every UAV and target, and each UAV's remaining fuel and munitions.

- *Markovian dynamics*

  System dynamics are modeled in discrete time as Markovian processes. That is, the future depends only on the system's present state, not the past.

- *stochastic and strategic transitions*

  UAV transits between sites are deterministically strategic; however, the probability distributions associated with each UAV's strike success and attrition introduces stochastic elements into the problem formulation. These stochastic attributes and Markovian dynamics represent uncertainty in the mission.

- *synchronous and static*

  The UAVs' joint actions synchronize their interaction with the static system.

A review of the system dimensions discussed in Section 1.3.3 is provided in Figure 2-3. Each dimension can be varied to represent the world more completely and realistically, though elaborate models tend to increase the complexity of the planning problem. Figure 2-3 categorizes the complexity of system dimensions. Properties of the planning systems considered in this thesis are highlighted.

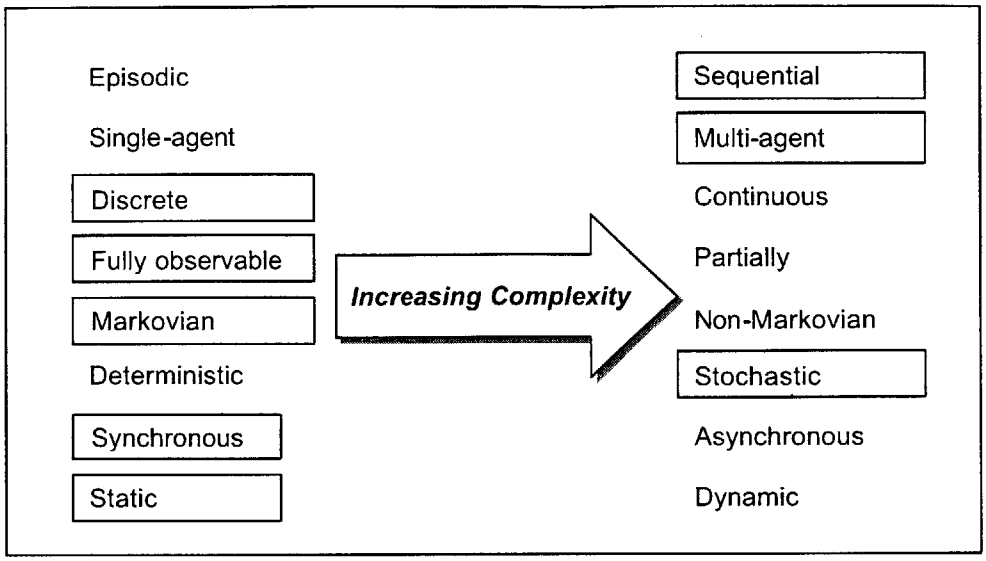

Figure 2-3 Complexity classification of possible system dimensions

In summary, the examined scenarios share the following characteristics: sequential, multi-agent, discrete state and action spaces, fully-observable states, Markovian dynamics, stochastic state transitions, synchronous actions, and static systems. Aspects of the examined systems purposefully belong to both orders of complexity to simplify the simulated systems and provide real world generality.

2.3.4 Evaluating planning algorithms

A chief concern of this thesis is to characterize the capabilities and limitations of the tested, planning algorithms. Alas, there is no "silver bullet" evaluation standard. By intuition, one planner might be better than another if it is more likely to find the optimal solution or policy. Often though, users of plans do not seek optimality. They prefer a plan that is good enough, rather than wait for an optimal policy generator to converge. This trade-off between optimality and resource consumption is typical of many artificial intelligence problems.

For the purposes of this thesis, optimality depends on the cooperative behavior of the UAVs, determined by the centralized planner. As described in Section 1.3.6, positive cooperation is defined by improving objective completion efficiency with the addition of each agent. Cooperative behavior is evaluated on three mission performance metrics:

speed, reward, and cost. Simulated mission times do not accurately characterize speed because the tested algorithms differ in trajectory control strategies. Instead, the relative speed of mission completion is captured by considering vehicle-to-target assignment and strike sequencing. These attributes offer a high-level perspective into the cooperation exhibited by the UAVs. In this thesis, the UAVs share the objective to collectively obtain a maximum reward by destroying targets.

An algorithm that generates plans with favorable mission behavior and poor computational performance is impractical. Planning that consumes hours of computation is not feasible for real-time systems, and plans that are "good enough" may be preferred over waiting for an optimal policy generator to converge. Computational performance is measured on the basis of planning time and memory consumption. Algorithms may display unique performance characteristics, depending on the execution machine and test scenario. To ensure consistent computability statistics, all simulations are performed on a single 1.8 GHz Linux workstation with 512 MB of RAM.

Since action decisions can induce a probability distribution over the set of possible states for each initial state of the nondeterministic system, Monte-Carlo simulations assess average-case performance [30]. Scalability tests evaluate the planners' mission and computational performance trends in scenarios that include increasing numbers of UAVs and targets. Additionally, sensitivity trials are used to capture each algorithm's robustness to real world planning environments where planners must negotiate incomplete or inaccurate system models.

A primary interest of the artificial intelligence community is the formalization of real world problems. Researchers must balance simplifying the real world for efficient computability with modeling the inherent complexity that defines reality. This thesis does not introduce any new models of the real world. Instead, the presented results examine several existing methods and models. This thesis follows an experimental paradigm. Because this thesis focuses on the empirical behavior of planners, rigorous mathematical proofs are not provided. Additionally, relevant implementation details are described at a high-level because they are described more thoroughly by the authors of the referenced works.

[*Except for this sentence, this page intentionally left blank*]

3 Planning Algorithms

"It is a truth very certain that when it is not in our power to determine what is true we ought to follow what is most probable"

– Aristotle

3.1 Overview

Planning algorithms are designed to select a course of action, which leads participating agents to best achieve their objectives. Generally, planning problems are composed of five basic ingredients [35]:

1. a description of the initial system
2. a set of actions that can be performed on the system
3. a description of the goal states for the system
4. a description of the system constraints
5. a valuation function that describes action costs and state rewards

The planner must find a sequence of actions from a particular initial state to a goal state that satisfies constraints and maximizes value. Decision-makers tend to balance the potential of attaining a goal state, the risk of causing an unfavorable state, and the cost of

performing the action. Two planning approaches are evaluated in the context of the cooperative UAV, target assignment and sequencing problem: optimal policy generation using Markov decision processes and heuristic approximation using receding horizon. A key objective of this thesis is to examine the strengths and weaknesses of each method. This chapter provides an overview of the Markov decision processes and receding horizon planning methods.

3.2 Markov decision processes

Markov decision processes (MDPs) formalize some problems of planning under uncertainty. Based on probability and utility theory, MDPs can weigh the benefits and trade-offs of following a particular plan. MDPs adhere to the Markovian property which implies that the future probabilistic behavior of a process is conditional on the current state, independent of past history. In the presence of stochastic action outcomes, MDP solvers compute the long-term value or expected utility of performing a particular action in a particular state to formulate an optimal policy, which maximizes the expected utility of actions.

3.2.1 Historical origins

In the early 1900s, Andrei Markov began work that would bring forth the theory of stochastic processes. He studied state sequences in which future states could be predicted with knowledge only of the current state. That is, for sequences that possess the Markov property, the future only depends on the present and is independent of prior history. At the time, Markov thought his revolutionary idea was applicable only to literary texts. Indeed, he famously proved his discovery of Markov chains by calculating the probability of vowel positions in A. S. Pushkin's poem "Eugeny Onegin." He showed that in Pushkin's poem the probability of finding two consecutive vowels is 0.128, and the probability of a vowel following a consonant is 0.663 [1]. Markov's tedious calculations were rewarded when Norbert Weiner began to rigorously treat continuous Markov processes in 1923, and Andrei Kolmogorov formed a general theory for

stochastic processes in the 1930s. Markov processes are now used in a wide-range of fields, including social sciences, atomic physics, quantum theory, and genetics [1].

Richard Bellman formalized the concept of Markov decision processes in the 1950s [2]. Later, MDPs were recognized as a fundamental mathematical construct for representing planning problems in the presence of uncertainty [13]. MDPs are used to formalize domains in which actions may have probabilistic results and agents have access to the system's state. Indeed, the MDP model serves as a basis for algorithms that provably find optimal policies (mappings from system states to actions) given a stochastic model of the system and the goal [9]. Comprehensive coverage of the MDP framework and controller is provided in [5, 19, 33].

While the exact planner examined in this thesis follows the traditional MDP paradigm, partially-observable Markov decision processes (POMDPs) handle problems in which a system's state is not completely known at all times. The POMDP model addresses the uncertainty associated with partially-observable domains by uniformly treating actions that affect the system and those that affect the agent's state information [9]. Michael Littman related various Markov models, based on state observability and control over state transitions, as shown in Figure 3-1 [26]. To learn more about these models, the reader may consult the referenced works.

Markov Models		Control over state transitions?	
		YES	NO
Completely observable states?	YES	**MDP** Markov Decision Process [33]	**Markov Chain** [15]
	NO	**POMDP** Partially-Observable Markov Decision Process [9]	**HMM** Hidden Markov Model [34]

Figure 3-1 Relationship of various Markov models

3.2.2 Fully-observable framework

Finite, fully-observable Markov decision processes (MDP) are defined by a five-tuple $\mathcal{M} = (\mathbf{S}, \mathbf{A}, P, R, \beta)$. $\mathbf{S} = \{\mathbf{S}_1, ..., \mathbf{S}_S\}$ is a finite set of system states that describe possible states of the system, $\mathbf{A} = \{A_1, \ldots, A_A\}$ is a finite set of actions that can be performed by the agents, P is the Markovian state transition model, R is a reward function, and β is a discount factor. The initial state of the system is $\mathbf{S}_1$, and the number of possible states in the model is given by $|\mathbf{S}|$. Actions trigger stochastic state transitions that have a probability $P(\mathbf{S}_j \mid \mathbf{S}_i, \mathbf{A}_k)$. $P(\mathbf{S}_j \mid \mathbf{S}_i, \mathbf{A}_k)$ is the probability that state $\mathbf{S}_j$ is reached after taking action $\mathbf{A}_k \in \mathbf{A}$ in a prior state $\mathbf{S}_i$, where $\mathbf{S}_i, \mathbf{S}_j \in \mathbf{S}$. A reward function $R(\mathbf{S}_i) : \mathbf{S} \mapsto \Re$ provides a mapping between possible system states and real-number valuations. The rewards are bounded by a maximum reward, R_{max}, where $R_{\max} \geq |R(\mathbf{S})|, \forall \mathbf{S}$. A discount factor $\beta \in [0,1]$ prioritizes the collection of rewards by discounting those that are available farther in the future.

In the cooperative UAV simulations, the planner is given the initial coordinates of the UAVs and the fixed locations of the target sites. The system's state includes the health and position of every UAV and target, and each UAV's fuel and munitions. The possible values of these states are shown in Figure 3-2.

System States	
UAV Health	Alive, Dead
UAV Position	$x_1, x_2, \ldots, x_N$
UAV Fuel	0, ..., D
UAV Munitions	0, ..., O
Target Health	Alive, Dead
Target Position	$y_1, y_2, \ldots, y_M$

Figure 3-2 Fully-observable, UAV mission state variables

The UAVs have two possible actions of varying cost: move between any two sites or strike a particular target. The state transition model represents both the determinism of certain attributes, such as fuel and position, and the stochastic nature of UAV strike

capability and attrition. The reward valuation of a state $\mathbf{S}_k$ is $R(\mathbf{S}_k) = \sum_{i=1}^{N} T_i W_i$ where T_i indicates whether target y_i is destroyed and W_i is its value. UAVs are constrained by fuel (D), munitions (O), and *max_strikes*. The mission terminates in five possible states: (1) all the targets are destroyed, (2) all targets are struck to their *max_strikes* constraint, (3) every UAV is lost, (4) the surviving UAVs have expended their fuel, or (5) the surviving UAVs have expended their munitions.

The planner should determine a joint course of action for the UAVs that maximizes reward. A stationary policy function p : **S ? A** describes a specific plan for an agent, where $p(\mathbf{S}_i)$ gives the action to be taken by the agent in system state $\mathbf{S}_i$. In the presence of uncertainty, the optimal policy provides courses of action of maximum expected utility or total discounted reward. The value function $V_\pi \in \Re^N$ of a policy p is determined by the expected discounted reward accumulated from an initial state $\mathbf{S}_i$. The expected value $V_\pi(\mathbf{S}_i)$ of a policy p for an initial state $\mathbf{S}_i$ satisfies [33]:

$$V_\pi(\mathbf{S}_i) = R(\mathbf{S}_i) + \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \pi(\mathbf{S}_i)) \cdot V_\pi(\mathbf{S}_j) \tag{3.1}$$

A policy p is optimal if $V_\pi \geq V_{\pi'}$ for all $\mathbf{S}_i \in \mathbf{S}$ and all policies p'. That is, an optimal policy identifies maximizing actions for the global set of possible states. The optimal value function V^* is the value of any optimal policy p^*.

3.2.3 MDP optimal policy solvers

Several algorithms exist for generating an optimal policy for an MDP. Three typical methods are linear programming, value iteration, and policy iteration. Each method uses a different approach to calculate the value function. The value function $V_\pi(\mathbf{S}_i)$ gives the value for every possible state $\mathbf{S}_i \in \mathbf{S}$ in the system's state space. MDP solvers determine the optimal value function V^* to compute the value of an optimal policy p^*.

3.2.3.1 Linear programming

Manne first proposed the linear programming (LP) approach to obtain the optimal value function for an MDP [31]. The LP variables $V(\mathbf{S}_i)$ for each state $\mathbf{S}_i \in \mathbf{S}$ represent the value $V_\pi(\mathbf{S}_i)$ for starting in state $\mathbf{S}_i$. The LP is defined by

$$\begin{aligned} &\text{Variables:} && V(\mathbf{S}_i), \forall \mathbf{S}_i \in \mathbf{S} \\ &\text{Minimize:} && \textstyle\sum_{\mathbf{S}_i} \alpha(\mathbf{S}_i) V(\mathbf{S}_i) \\ &\text{Subject to:} && V(\mathbf{S}_i) \geq R(\mathbf{S}_i) + \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \mathbf{A}_k) \cdot V(\mathbf{S}_j), \forall \mathbf{S}_i \in \mathbf{S}, \mathbf{A}_k \in \mathbf{A} \end{aligned} \tag{3.2}$$

where $\alpha(\mathbf{S}_i)$ is the LP cost-vector or state relevance weighting, which is positive and normalized to sum to one [19]. Equation 3.2 subjects $V(\mathbf{S}_i)$ to the constraint that it is either greater than or equal to $R(\mathbf{S}_i) + \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \mathbf{A}_k) \cdot V(\mathbf{S}_j)$. The minimization of $\sum_s \alpha(\mathbf{S}_i) V(\mathbf{S}_i)$, however, ensures $V(\mathbf{S}_i)$ equals this constraint.

3.2.3.2 Value iteration

Value iteration is a commonly used alternative approach for constructing optimal policies [2]. The algorithm builds a series of n-steps-to-go value functions V^n, starting with an initial estimate of the value function $V^0 = R$. The value at the next step is given by

$$V^{n+1}(\mathbf{S}_i) = R(\mathbf{S}_i) + \max_{\mathbf{A}_k \in \mathbf{A}} \left\{ \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \mathbf{A}_k) \cdot V^n(\mathbf{S}_j) \right\} \tag{3.3}$$

The sequence of value functions V^n linearly converges to the optimal value function V^*, which provides an optimal policy p^* that maximizes Equation 3.3.

3.2.3.3 Policy iteration

Policy iteration solves for the optimal policy of MDPs by iteratively searching in the space of policies [23]. Starting with an initial policy p^0, the algorithm includes both phases of value determination and policy improvement. In the value determination phase, the value function V_π^t is established for the policy p^t. Policy improvement selects the next policy by $\pi^{t+1} = \max_a V_\pi^t$. Policy iteration converges to the optimal policy p^* [33].

In practice, policy iteration tends to be faster than the linear programming approach [33]. Puterman also showed that the convergence of policy iteration is bounded by the number of iterations required for value iteration. Policy iteration tends to find the optimal policy in fewer iterations than value iteration, though each iteration is more computationally expensive [19].

3.2.3.4 Curse of dimensionality

MDP optimal policy computation has been shown to be P-complete [32]. P-complete decision problems are the hardest problems that can be solved in polynomial time with parallel computers. MDPs suffer from three curses of dimensionality: large state spaces, large action spaces, and large outcome spaces. Although standard MDP algorithms usually converge in relatively few iterations, each iteration requires computation time at least linear in the size of the state space (for value iteration, more for other algorithms) [6]. The very design of an MDP's optimal policy with contingencies for a global state space is intrinsically affected by the dimensionality of a system. For instance, the cooperative UAV system's states include the health and position of every UAV and target, and each UAV's remaining fuel and munitions. Each system state $\mathbf{S}_i \in \mathbf{S}$ is described by an assignment of these component state variables $\mathbf{S}_i = \{s_1,...,s_s\}$. Consequently, the number of possible states is exponential in the number of state variables. Richard Bellman described this exponential relationship as a "curse of dimensionality" [2].

Multi-agent systems include another curse of dimensionality. Each of the UAVs' joint actions $\mathbf{A}_k \in \mathbf{A}$ represented in the system model is defined by a set of actions $\mathbf{A}_k = \{a_1,...,a_a\}$, where the action for UAV $_j$ is a_j. Viewing each agent's action as an action variable, the number of joint actions is exponential in the number of action variables [19].

Representation of the MDP model is impractical in systems described by many state variables, involving many agents, or including agents with many actions. For example, the transition model for performing a particular joint action in a certain state assigns a probability distribution over states in the next time step. In large systems, a tabular representation of this model is restrictive because it requires a set of entries that is exponential in the state and action spaces. Similarly, a tabular representation of the reward function, which assigns values to the set of system states, also limits scalability.

Although advances in hardware have provided faster processors and greater memory at lower cost, classical MDP planners are limited by the curse of dimensionality. Researchers have focused on developing computational and representational methods for solving MDPs without intensive enumeration of the complete state space [6]. Aggregation methods view a set of states as a single aggregate state [5]. Abstraction, another form of aggregation, has also been used to drop certain details of the model [14]. These techniques exploit the structure of a problem to compactly represent the reward function and the transition function. In fact, many of these abstract MDPs can be automatically generated with probabilistic rules [21] or dynamic Bayesian network (DBN) action representations [13].

3.2.4 Stochastic planning using decision diagrams

A particularly powerful, dynamic abstraction method for solving MDPs is stochastic planning using decision diagrams (SPUDD). Developed by Jesse Hoey, Robert St-Aubin, et al., SPUDD utilizes algebraic decision diagrams (ADDs) to represent value functions and policies [22]. ADDs extend binary decision diagrams, which only permit Boolean transitions for state variables. ADDs can include descriptions of the value and policy functions in the natural language of the problem domain. SPUDD's dynamic

programming algorithm follows the classical MDP value iteration paradigm described in Section 3.2.3. The expected number of value iterations is significantly condensed by exploiting regularities in the ADD action and reward networks. Compact decision graphs aggregate equivalent states during dynamic programming computation. Unlike decision trees, decision graphs allow identical subtrees of the same value to be merged into one. This reduces both the number of expected value computations and maximizations needed by dynamic programming. Hoey and St-Aubin showed that their SPUDD approach scales better with larger state spaces than a classic, tabular value iteration MDP algorithm [22]. The state space can be further reduced by specifying the tolerance of policy satisfaction and the depth of value iteration. These approximation techniques do not guarantee, however, that SPUDD will produce an optimal policy.

3.2.4.1 Action representation

Actions are taken to interact with the system. Dynamic Bayesian Networks (DBNs) can be used to describe this interaction based on an action's effects on particular state variables of the system. Indeed, DBNs model stochastic processes as directed graphs, which represent a system's state in terms of state variables and their interdependencies [22]. Furthermore, a DBNs' graphical representation exploits the conditional independence of state variable transitions and is usually quite compact.

For instance, the Markovian transition model $P(\mathbf{S}_j \mid \mathbf{S}_i,\mathbf{A}_k)$ gives the probability that state $\mathbf{S}_j$ is reached after taking action $\mathbf{A}_k \in \mathbf{A}$ on a prior state $\mathbf{S}_i$, where $\mathbf{S}_i,\mathbf{S}_j \in \mathbf{S}$. A DBN for an action $\mathbf{A}_k$ requires both a set of prior state variables $\mathbf{S}_i = \{s_{i1},...,s_{iS}\}$, which describe the state of the system before performing $\mathbf{A}_k$, and analogous states after execution $\mathbf{S}'_i = \{s'_{i1},...,s'_{iS}\}$. A sample representation for a UAV's strike action is shown in Figure 3-3a. In this example, there are two UAVs x_1 and x_2, one target y_1, and two possible target site locations loc_1 and loc_2. The two UAVs begin their mission from a common, initial position *home*. Directed arcs from variables in $\mathbf{S}_i$ to $\mathbf{S}'_i$ indicate causal influences to the effected state.

Since the process is fully-observable, the DBN is used only to predict future state transitions and not pre-action states. These predictions allow the SPUDD planner to

compute the expected utilities of a global state space and generate a contingency-based policy for uncertain events.

Individual DBNs are required for each action $\mathbf{A}_k \in \mathbf{A}$. Figure 3-3b depicts two of the Markovian transition tables for UAV x_1's strike action (x_1_*strike*). Adhering to the Markov property, the first table shows that x_1's health (x_1_*alive'*) after executing a strike is solely dependent on its current health (x_1_*alive*). If UAV x_1 is currently alive (x_1_*alive* = T), the probability that it remains alive (x_1_*alive'* = *T*) after performing a strike is 0.6. On the other hand, the probability that UAV x_1 survives (x_1_*alive'* = *T*) a strike is 0.0, if it was previously lost (x_1_*alive* = *F*).

The second table considers the affect of the strike action on the state of target y_1's health. Target y_1's subsequent health state (y_1_*alive'*) after a strike operation is dependent on the current states of its health (y_1_*alive*), UAV x_1's health (x_1_*alive*), and x_1's position (x_1_*loc*). If target y_1 was previously destroyed (y_1_*alive* = *F*), the probability that it survives (y_1_*alive'* = *T*) UAV x_1's strike is 0.0, regardless of other states. If both target y_1 and UAV x_1 are currently alive (y_1_*alive* = *T* and x_1_*alive* = *T*), the probability that y_1 survives x_1's strike (y_1_*alive'* = *T*) is 1.0 if x_1 struck at its starting position (x_1_*loc* = *home*) and 0.5 if x_1 struck at either site loc_1 (x_1_*loc* = loc_1) or loc_2 (x_1_*loc* = loc_2). Finally, if UAV x_1 was lost (x_1_*alive* = *F*) and target y_1 is still active (y_1_*alive* = *T*), the probability that y_1 remains alive (y_1_*alive'* = *T*) after a strike is 1.0. Notice that the state of UAV x_2 does not factor into the representation of UAV x_1's strike action.

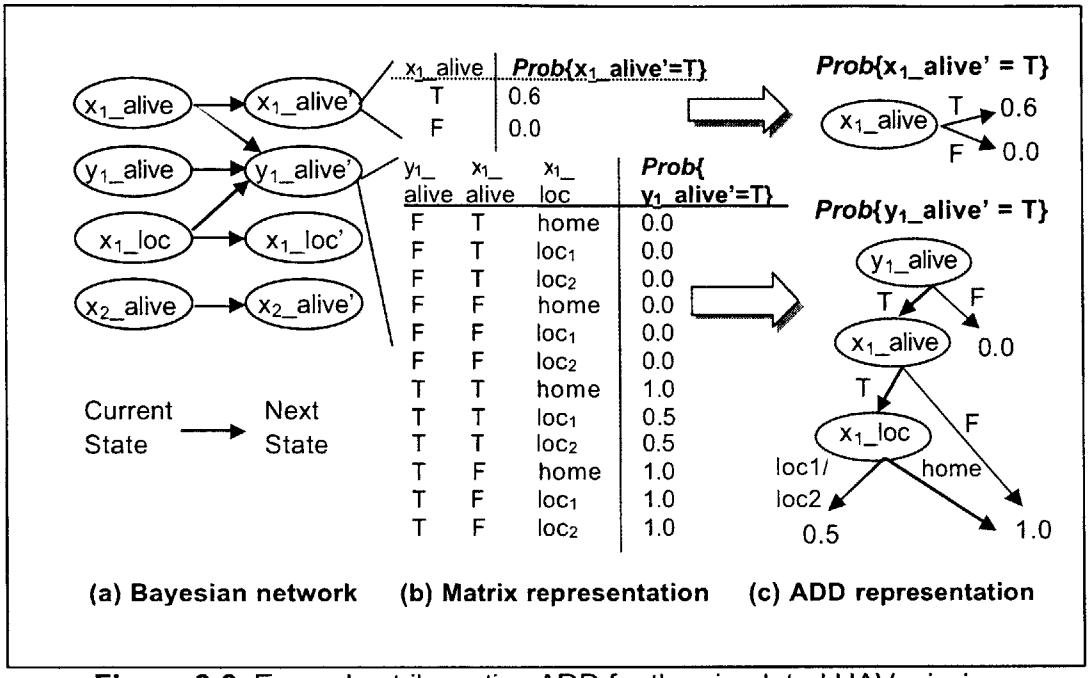

Figure 3-3 Example strike action ADD for the simulated UAV mission

Figure 3-3c shows the condensed ADD representation that SPUDD automatically formulates from the tabular input. Nodes in the ADD represent current states, and leaves represent conditional probabilities of transitioning to a future state. The top ADD indicates the probability that UAV x_1 is alive ($x_1\_alive' = T$) after a strike execution is either 0.6 or 0.0 depending on its previous health (x_1_alive). The second ADD shows that the probability that target y_1 survives ($y_1\_alive' = T$) UAV x_1's strike is dependent on its current health (y_1_alive), x_1's health (x_1_alive), and x_1's position (x_1_loc). The ADD representation exploits table regularities to reduce the table representation of target y_1's strike survival, which includes twelve conditional parameters, to a tree with three nodes and three leaves. Whereas a tabular representation of the Markovian transition model grows exponentially with the number of state variables, ADDs exploit context-specific independence in the distributions to merge identical subtrees of the same value into one [22]. For instance, SPUDD's ADD uses one node and one leaf to effectively represent target y_1's 0.0 probability of surviving x_1's strike if it is currently destroyed ($y_1\_alive = F$). Conversely, the traditional matrix representation includes six table entries to represent this same dependency. The abstraction technique employed by SPUDD is

related to previous work in constructing DBN conditional probability tables with tree- and rule-based representations [6].

3.2.4.2 Reward representation

The reward function shown in Figure 3-4 also benefits from the ADD's compressed representation. For the two-UAV and one-target example, the reward valuation is based on the states of the target y_1, and UAV x_1 and x_2. The ultimate reward is determined by the final conditions of these states.

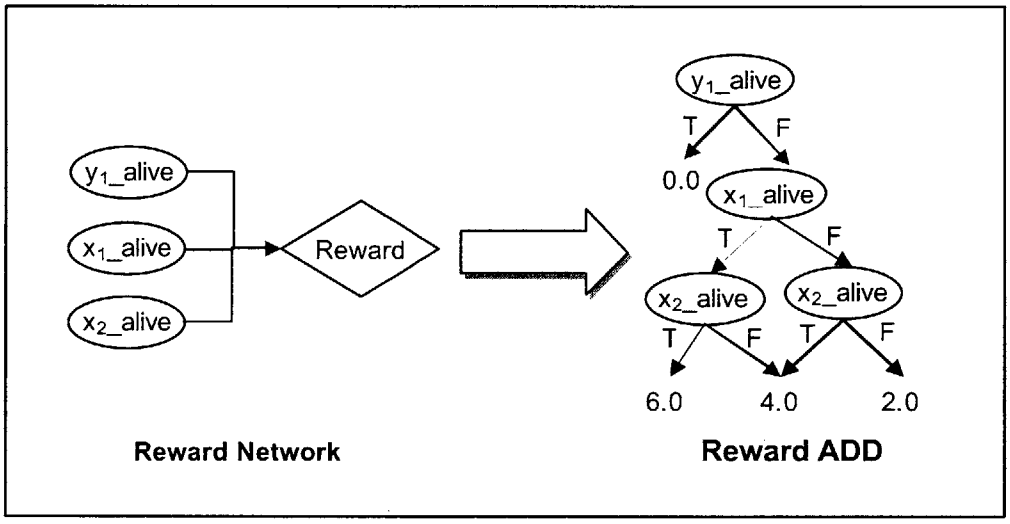

Figure 3-4 Example reward network and ADD for the simulated UAV mission

3.2.4.3 Policy representation

SPUDD adheres to the MDP optimal value iteration algorithm discussed in Section 3.2.3. SPUDD reduces the expected number of value iterations by exploiting regularities in the ADD action and reward networks and aggregating states of equivalent value during dynamic programming computation. Regularities in the action and reward networks are used to discover regularities in the search for an optimal value function. This approach avoids explicit enumeration of the entire state space, and yields significant savings in computational time and space [22]. Figure 3-5 shows the value function ADD generated by SPUDD for a system model that includes action and reward ADDs that follow from

Figure 3-3 and Figure 3-4. SPUDD computes the expected values of possible states based on state transition probabilities, action costs, and reward valuations.

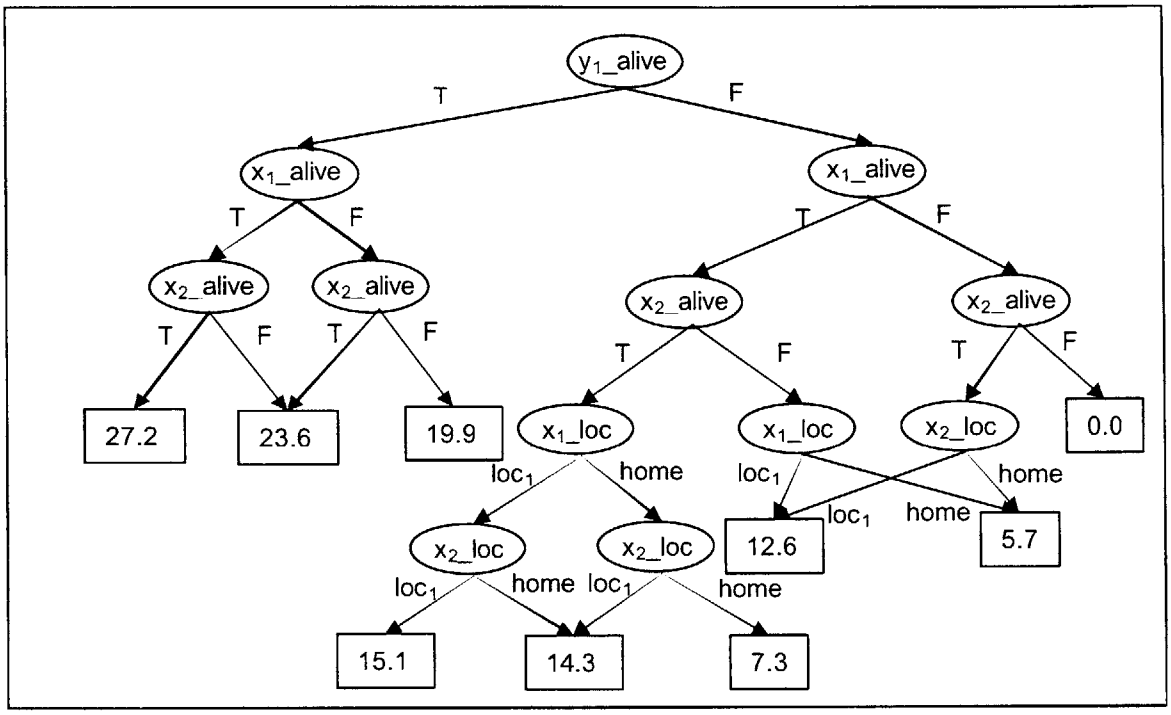

Figure 3-5 Example value function ADD for the simulated UAV mission

SPUDD constructs an optimal policy from its derived value function. The policy provides actions of maximum utility that are contingent on a global set of possible states. In this example, the battlefield is comprised of two UAVs x_1 and x_2, two locations *home* and loc_1, and one target site y_1. Participating UAVs can perform strike (x_j_*strike*) operations at a location, transits (x_j_*move*) to the target site loc_1 from their initial position *home*, or terminate the mission (*stop*).

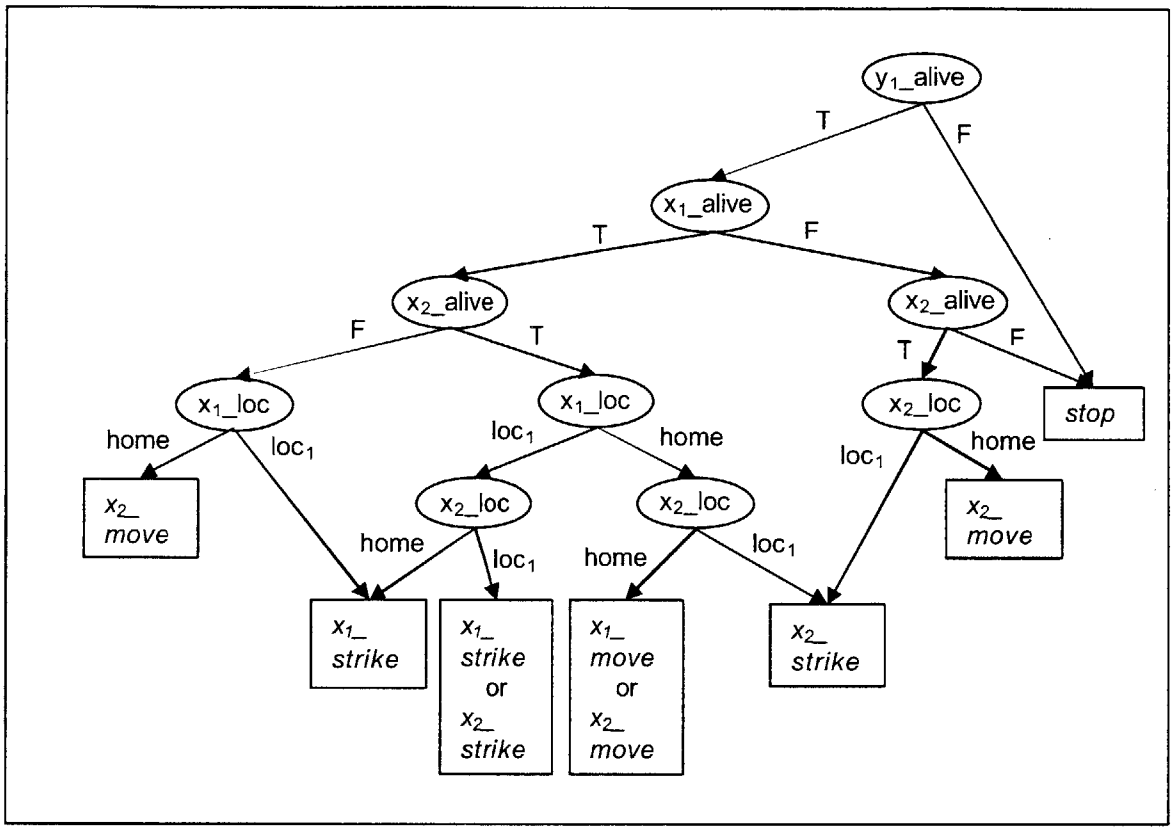

Figure 3-6 Example policy ADD for the simulated UAV mission

ADD representations not only simplify evaluation of the system model, but also make the dynamics of the system more comprehensible. That is, ADDs indicate the conditional commonalities of states that share the same value and/or plan in a flow-down format that appears easier to read than a tabular chart. Still, human planners are confronted with the difficult task of accurately representing their "best guess" of the actual system's dynamics. Complete and fully-observable planner models match the system's true nature by definition. In the real world, a planner must possess robustness to negotiate scenarios where its system model might be incomplete or inaccurate. The interrelationships of SPUDD's model parameters and its "optimal" policy are considered in the next section.

3.2.5 Factors affecting SPUDD's optimal policy

The SPUDD planner determines an optimal policy based on its model of the system. Operating on a fully-observable battlefield, the locations and states of the UAVs and the target sites are accurately known in real-time.

For the cooperative UAV mission, SPUDD's system model includes the costs for move and strike operations, stochastic distributions for successful strikes and UAV attrition, reward valuations, and a discount factor for future rewards. Indeed, the interdependency of these variables is critical to the behavior exhibited by the UAVs. For instance, although the cost of transiting between sites is correlated to distance, specific cost assignments should be proportional to the other factors in SPUDD's system model to prevent biasing.

The relationship of SPUDD's model parameters to its optimal policy is determined by its optimization equation. SPUDD generates an optimal policy through value iteration by computing the expected utility or total discounted reward for a global set of possible states. As described in Section 3.2.3, the expected total discounted reward of a policy p for an initial state $\mathbf{S}_i$ is computed as

$$V_{\pi}(\mathbf{S}_i) = R(\mathbf{S}_i) + \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \pi(\mathbf{S}_i)) \cdot V_{\pi}(\mathbf{S}_j)$$

3.2.5.1 Expected utility computations for one-UAV, two-target scenario

The advantage of following a particular plan is determined by computing the expected utilities of possible actions. For a very simple scenario, this section evaluates the relationship of SPUDD's model parameters to valuations of actions.

Consider a battlefield with a UAV x_1 and two targets y_0 and y_1. Rewards are accumulated by destroying target sites. The probability of a successful strike is maintained constant over multiple attempts. The model assumes the UAV has enough fuel to complete its mission and is indestructible. The expected utilities of rewards obtained in the future are discounted in time, as noted in each computation. At the time of optimization, the UAV is at position *loc0*. As shown in Figure 3-7, the UAV's next

possible action is either to strike its current target y_0 (*strike_loc0*) or to move to the neighboring target y_1 (*move_loc0_loc1*).

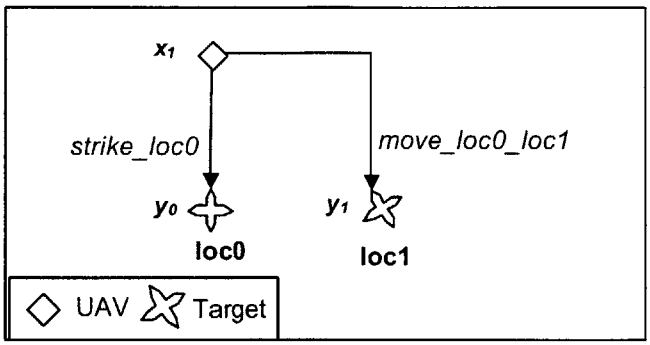

Figure 3-7 Canonical one-UAV, two-target scenario

Following the value iteration paradigm, SPUDD computes the expected utilities of both alternatives. The mission behavior exhibited by the UAVs in simulation can be related to these quantifiable, expected utilities. In this section, the expected utilities of the *strike_loc0* and *move_loc0_loc1* actions are compared for varying parameters of SPUDD's model. The expected value difference of the two actions represents the subtraction of the expected utility of *move_loc0_loc1* action from the expected utility of the *strike_loc0*. Therefore, positive expected value differences are associated with a greater utility for striking, whereas negative differences correlate to a higher expected utility for the move action.

The expected utilities are analytically computed by considering possible action sequences for the UAV over a value-iteration horizon of six time steps. This horizon is the number of time steps to look-ahead from the current state at time $t = 0$. For instance, Figure 3-8 depicts the scenario's possible action sequences for a horizon of three time steps. As described by Equation 3.1, the expected utilities of possible next actions are related to the system states that subsequent action sequences yield.

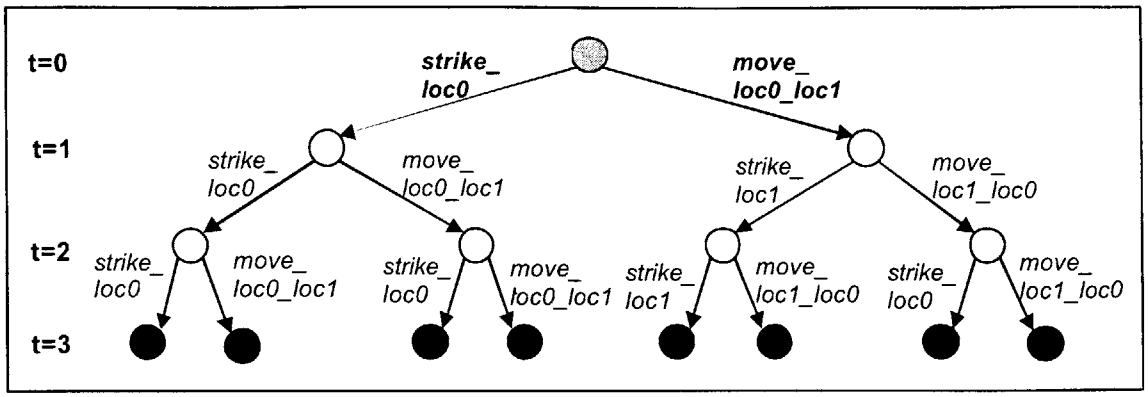

Figure 3-8 : Action tree for the one-UAV, two-target scenario for a three time-step horizon

3.2.5.1.1 Effects of target reward valuations

Figure 3-9 depicts the expected value difference of the two possible next actions with respect to the reward valuations of the targets. The system model includes a discount factor of 0.8, move and strike action costs of 1 unit, and the UAV's strike success probability of 0.5. The reward for destroying target y_0 is fixed at 2, and the reward for destroying target y_1 is varied to observe the affect of the targets' reward ratio (*loc1*-to-*loc0*) on the expected value difference. The cross-over target rewards ratio indicates the particular reward valuations of the targets where the expected utility of striking target y_0 becomes higher than moving to y_1. Namely, this cross-over point occurs at a target rewards ratio of 5, where the reward for destroying targets y_0 and y_1 are 2 and 10, respectively. The subsequent sections aim to highlight the dependence of this specific cross-over point on other parameters of system model, including discount factor, action costs, and strike success probability.

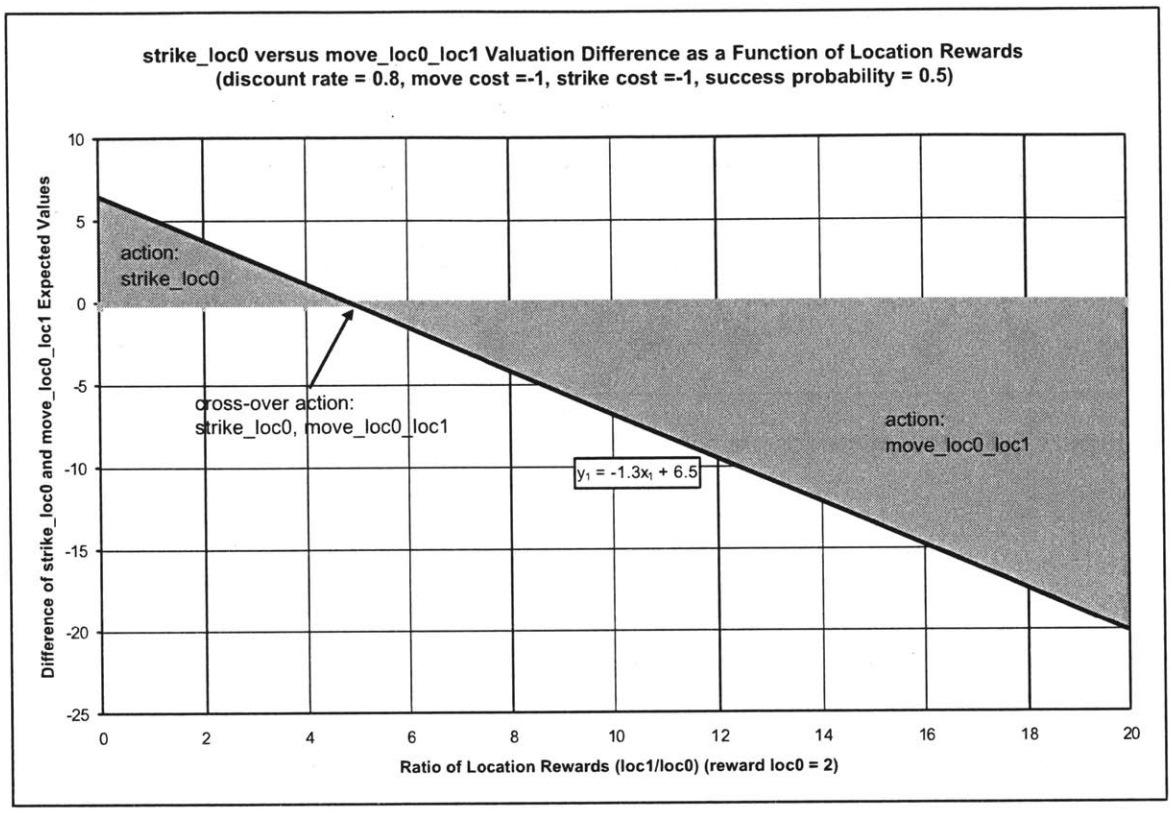

Figure 3-9 *strike_loc0* versus *move_loc0_loc1* valuation difference as a function of target rewards

3.2.5.1.2 Effects of action costs

Figure 3-10 generalizes the influence of target reward valuations on the expected utilities of the strike and move actions over a spectrum of action cost assignments. In this example, the discount factor is 0.8 and the UAV has a strike success probability of 0.5. The expected utility of striking target y_0 increases as the cost of moving to y_1 rises. Likewise, the expected utility of moving to y_1 increases as the cost of striking y_0 rises. UAV x_1 is attracted to the more valuable target for expensive strike operations. For instance, UAV x_1 passes target y_0 for y_1 in scenarios where the ratio of the target rewards (*loc1*-to-*loc0*) is large. The cross-over target rewards ratio increases for greater move action costs, and reduces for higher strike costs. That is, the *move_loc0_loc1* action has a higher expected utility if target y_1 offers a higher reward than y_0 and the *strike_loc0* action is expensive.

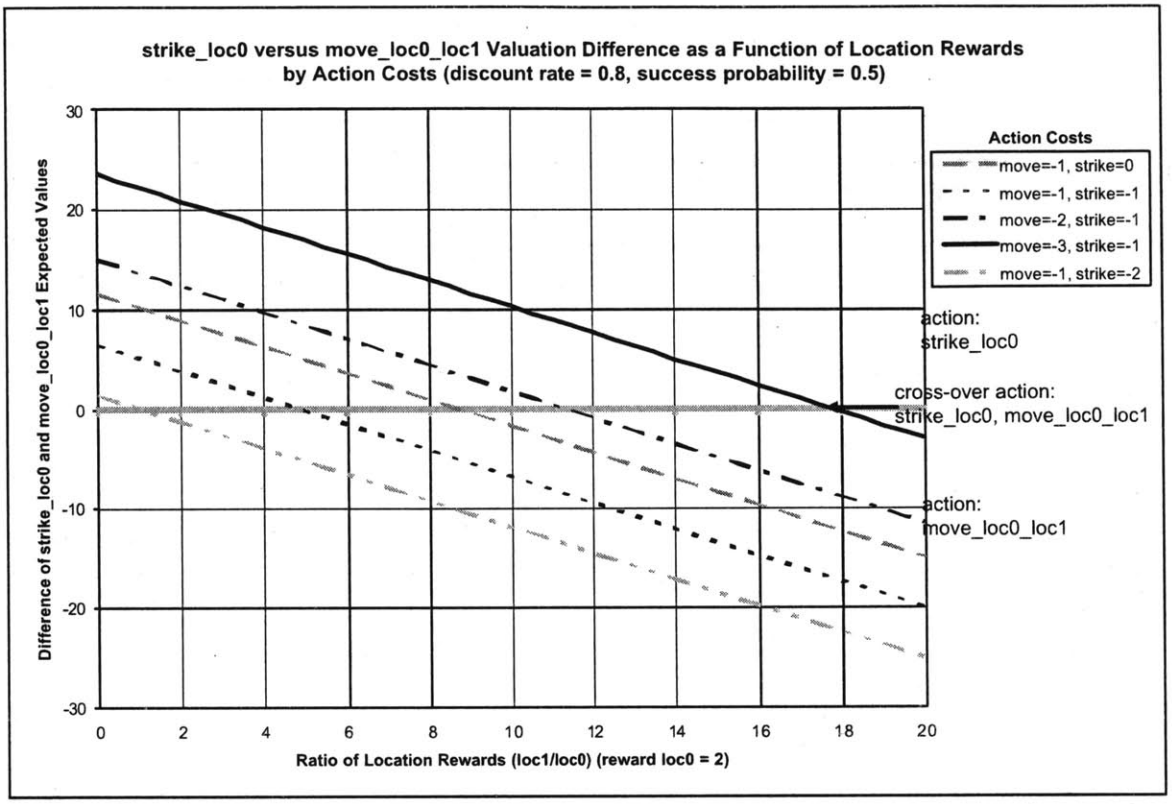

Figure 3-10 *strike_loc0* versus *move_loc0_loc1* valuation difference as a function of target rewards by action costs

Figure 3-11 considers the relationship of action costs to the expected value difference of the move and strike actions with a discount factor of 0.8, UAV strike success probability of 0.5, and a fixed reward of 2 and 8 for destroying targets y_0 and y_1, respectively. Here, action costs are a dominating factor in the expected utility calculations. Even though the target rewards ratio *loc1*-to-*loc0* is 4, the strike action has a greater expected utility for a majority of action cost ratios. The strike action has greater utility where the move action is costlier, and vice-versa. For instance, if the *move_loc0_loc1* action costs less than *strike_loc0* (i.e. *move_loc0_loc1*-to-*strike_loc0* action cost ratios below one), target y_1's larger reward supports a higher expected utility for the *move_loc0_loc1* action. Oppositely, a cost increase in the *move_loc0_loc1* action relative to the *strike_loc0* disproportionately improves the utility of the strike action – the first strike of target y_1 incurs both the cost of transiting to its location *loc1* (*move_loc0_loc1*) and the cost to

strike it (*strike_loc1*). In contrast, a strike at target y_0 only incurs the cost of the *strike_loc0* operation. As a result, the *strike_loc0* action has a greater expected utility.

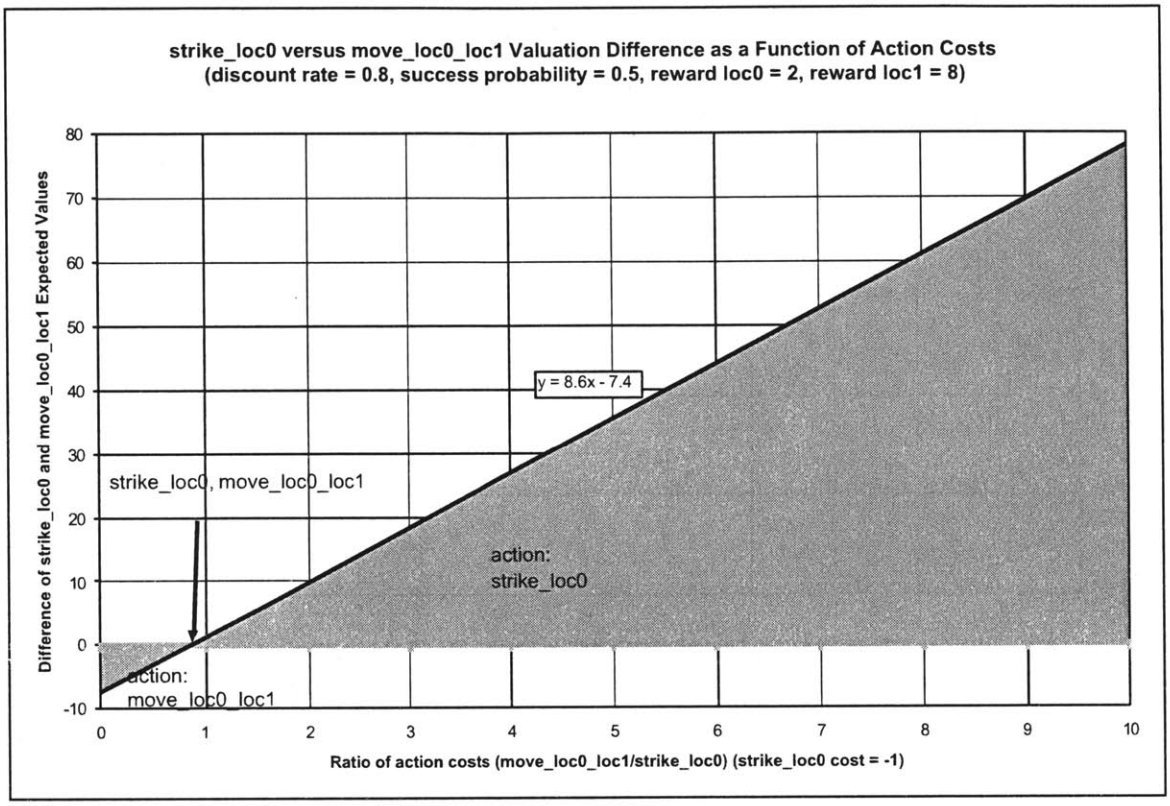

Figure 3-11 *strike_loc0* versus *move_loc0_loc1* valuation difference as function of the ratio of action costs (*move_loc0_loc1*-to-*strike_loc0*)

3.2.5.1.3 Effects of discount factor

Each of the preceding SPUDD models included a discount factor of 0.8. The discount factor $\beta \in [0,1]$ controls the effect of future rewards on the optimal policy. Future rewards are decayed less for large discount factors, and decayed more for small discount factors. Figure 3-12 plots the effect of the discount factor on the expected utilities of the strike and move actions for various target rewards settings. In this example, move and strike costs are both 1 unit and the UAV's strike success probability is 0.5.

For low discount factors, SPUDD greedily prefers actions that maximize immediate reward. Since the UAV is already at location *loc0*, low discount factors give

the *strike_loc0* action a higher expected utility. At higher discount factors, which are close to one, the reward valuations of the targets have a greater influence on the expected utility computations. The *strike_loc0* action has a higher expected utility if target y_0 has a larger or comparable reward to y_1, and the *move_loc0_loc1* action has a higher expected utility if target y_1 has a significantly larger reward. As noted in Section 3.2.5.1.2, this disproportionate discrepancy in behavior is related to the additional cost incurred for moving to target y_1 before striking it.

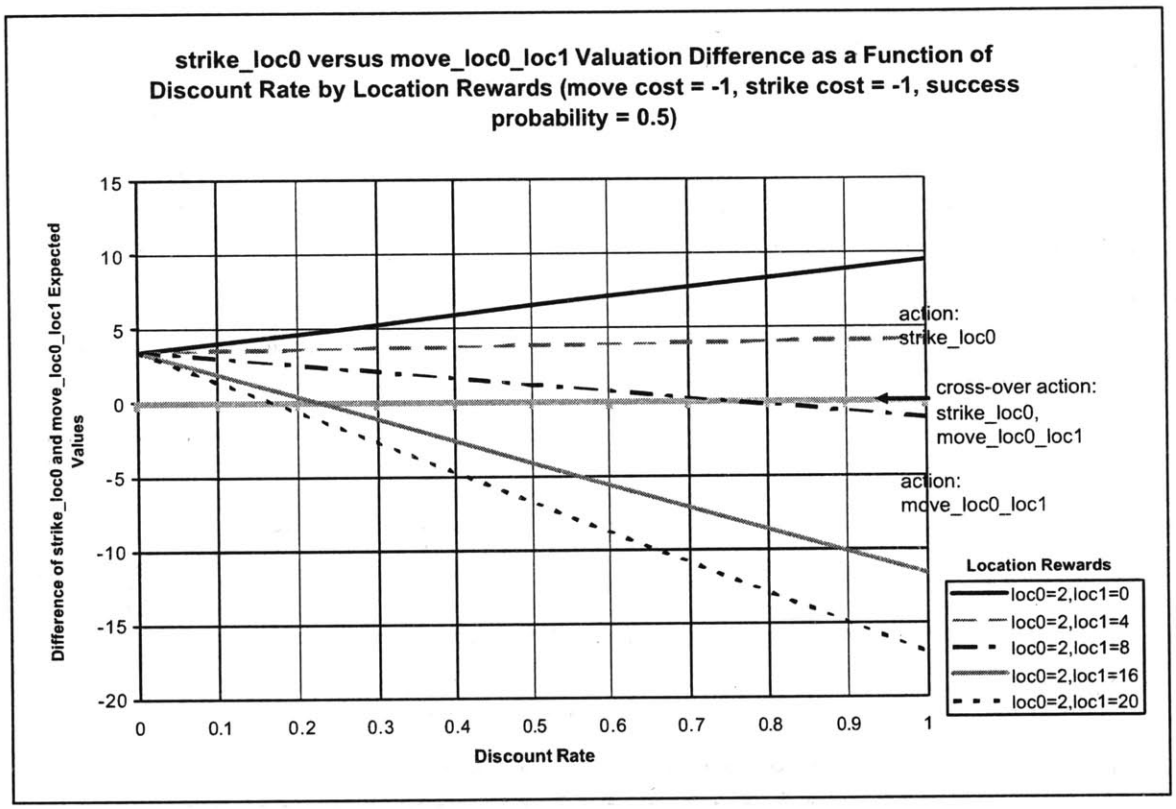

Figure 3-12 *strike_loc0* versus *move_loc0_loc1* valuation difference as a function of discount rate

3.2.5.1.4 Effects of strike success probability

Strike success probability correlates to the UAV's ability to acquire the reward it seeks. For a given ratio of target rewards, Figure 3-13 shows that the expected utilities of both the *strike_loc0* or *move_loc0_loc1* actions increase with higher strike success rates.

Accordingly, the expected value difference of the two actions crosses-over at a lower target rewards ratio for higher strike success probabilities. The expected utility of the *strike_loc0* action increases with strike success probability where the reward at target y_0 is comparable to y_1. Similarly, for larger target y_1 rewards, higher strike success probabilities increase the utility of the *move_loc0_loc1* action (i.e. yielding a negative expected value difference with respect to the *strike_loc0* action). The planner passes over target y_0 for y_1, which offers a high probability of obtaining a greater reward.

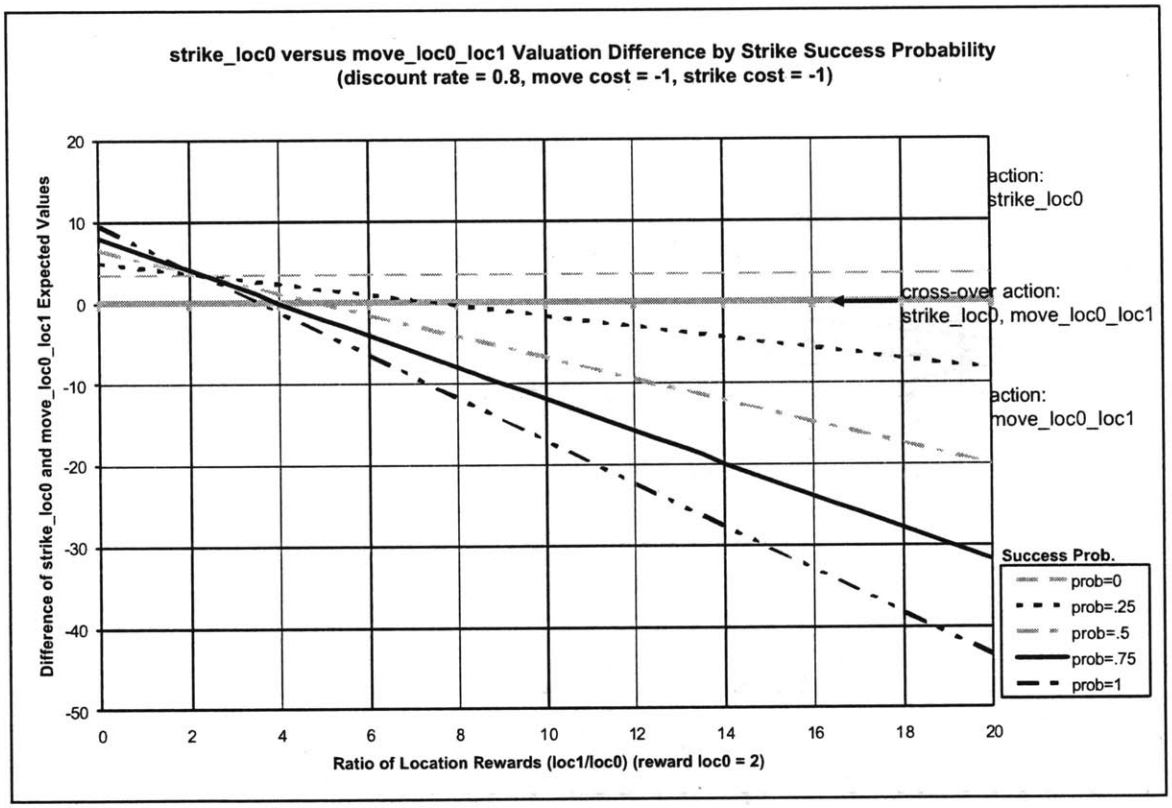

Figure 3-13 *strike_loc0* versus *move_loc0_loc1* valuation difference by strike success probability

3.2.5.2 Expected utility computation for one-UAV, three-target scenario

To confirm the computed trends, the expected utility calculations were extended to the more complex scenario depicted in Figure 3-14a, which includes one UAV and three targets. At the current time step, the planner must decide whether to strike the target y_0

(*strike_loc0*), move to y_1's site *loc1* (*move_loc0_loc1*), or move to y_2's site *loc2* (*move_loc0_loc2*). Transiting between target sites costs 2 units and each strike attempt costs 1 unit. The probability of a strike's success is 0.5, and the discount factor is 0.8. Like the one-UAV, two-target scenario, the reward valuations of the target sites significantly contribute to the expected utilities of the possible next actions. The expected valuations of the *move_loc0_loc1* and *move_loc0_loc2* actions are compared with the *strike_loc0* action over a horizon of 4 time steps in Figure 3-14b. Striking y_0 has a higher utility where the rewards of target y_1 and y_2 do not appreciably differ from the reward at y_0. The *move_loc0_loc1* and *move_loc0_loc2* actions have higher expected utilities (i.e. negative expected value difference with respect to *strike_loc0*) for greater rewards valuations of target y_1 and y_2, respectively. At the intersection of the *move_loc0_loc1* and *move_loc0_loc2* valuation planes, where the rewards for targets y_1 and y_2 are equal, the *strike_loc0* action has a higher utility for target rewards ratios below 3.

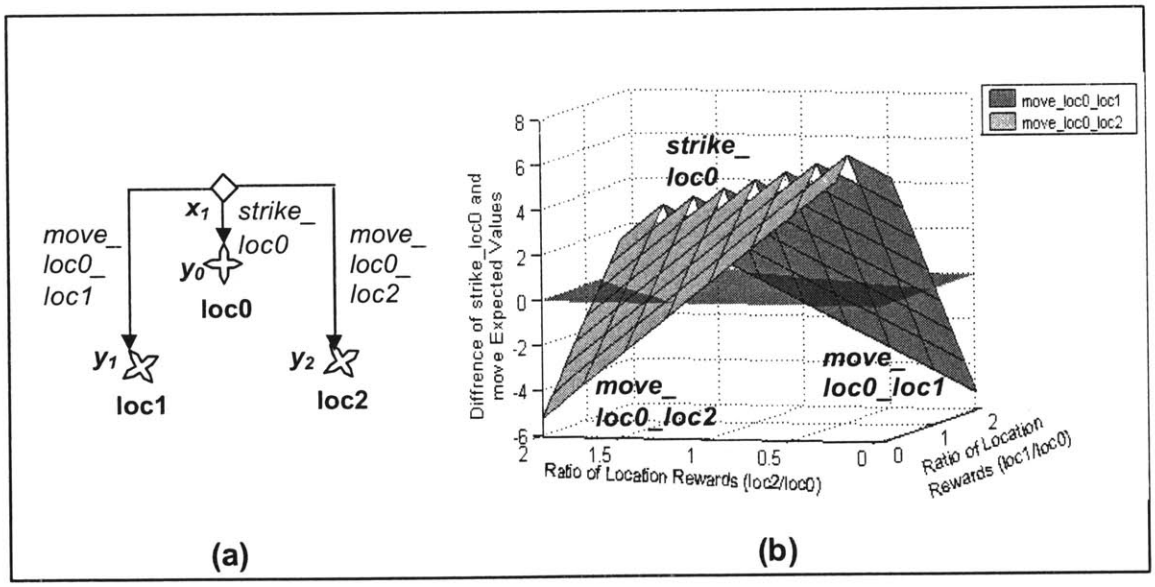

Figure 3-14 (a) One-UAV, three-target example scenario, (b) expected utility computation for varying target reward valuations (reward at loc0 = 6)

The computation confirms expectations by following the results obtained from the one-UAV, two targets example. Both Figure 3-9 and Figure 3-14b show that the expected

utility for striking target y_0 linearly decreases with increasing reward valuations of other targets in the scenarios.

3.2.5.3 Dimensionality of expected utility computations

For the simple scenarios considered, the preceding analysis provides a basic overview of the parameter interrelationships in SPUDD's model that influence the UAVs' observed behavior. The utilities of possible next actions form the basis of SPUDD's value iteration algorithm, which ultimately forms an optimal policy. This policy maximizes the expected utility of actions, using model parameters to determine contingencies for uncertain events.

Specific expected valuations and cross-over points are particular to the scenario representations. The results depict the trends that govern the interdependencies of parameter values in this problem.

Behavior exhibited in simulation can be quantifiably understood by computing the utility of possible actions, though this analysis increases in complexity with the size of the problem. These relatively small examples demonstrate the exploding complexity of the optimal, contingency planning problem as the numbers of UAVs and targets rise. Indeed, Figure 3-9 and Figure 3-14b illustrate the substantial increase in dimensionality of the expected utility computation from the one-UAV, two-target scenario to the one-UAV, three-target scenario.

3.3 Receding Horizon

The intractability of dynamic programming-based approaches, such as MDP solvers, in large domains has lead researchers to consider other methods of addressing complex stochastic optimal control problems [12]. Research has focused on decomposing the complexity of the overall problem into hierarchical levels – from high-level path planning and assignment of UAVs to target sites to detailed vehicle motion control [10]. For instance, Castañón addressed aspects of dynamic resource allocation [12] and How framed the cooperative UAV path planning problem as a mixed-integer linear program, incorporating task timing and vehicle capability constraints and including the presence of

obstacles [3]. At the level of detailing the UAVs' trajectories, planning issues include multi-vehicle formation control, obstacle avoidance, and stabilization [17, 24, 29].

3.3.1 Background

An alternative to these functional approaches is time decomposition. Receding horizon (RH) controllers maximize the total expected reward accumulated by the team over a given time horizon and periodically move this horizon forward in time [28]. RH schemes, which are associated with model-predictive control, have been used successfully for optimal control problems that do not have simple feedback solutions [11, 36]. Cassandras and Wei Li proposed a RH controller that dynamically selects UAV trajectories by sequentially optimizing over a planning horizon and executing decisions over a shorter action horizon [28]. This approach, shown in Figure 3-15, integrates vehicle assignment, sequencing, and routing construction into a single-tier, real time controller.

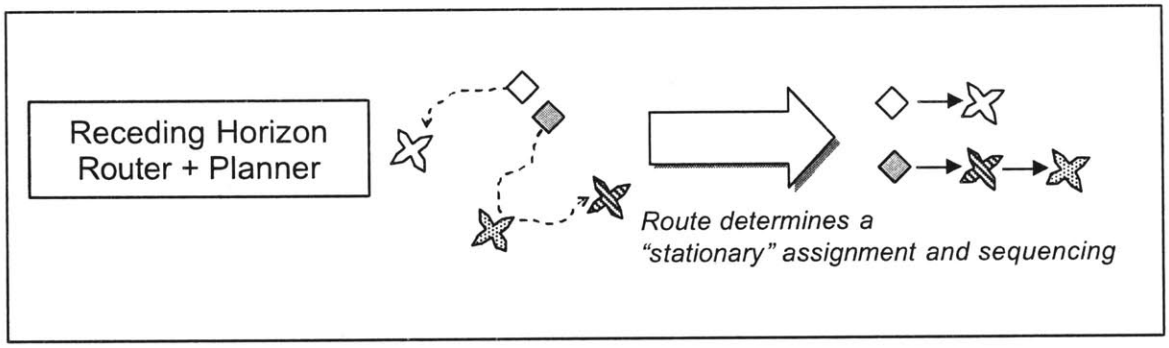

Figure 3-15 Receding horizon controller's integrated planning approach

3.3.2 Control scheme

Cassandras' RH scheme solves a nonlinear optimization problem that selects vehicle headings to place the UAVs in a position of maximum expected reward at the end of each planning horizon. Importantly, the controller somewhat simplifies the optimization problem by not attempting to make explicit vehicle-to-target assignments. To the RH

controller, a potential field overlays the region within each planning horizon. The UAVs are drawn to attack the most rewarding target sites in this space.

As described later, the headings chosen by RH have a convergence property that ensures vehicles are ultimately assigned to target sites. Cassandras and Li analytically proved that a stationary policy of vehicle-to-target assignment was obtained for one-vehicle, *M*-target [27] and two-vehicle, *M*-target [28] scenarios.

3.3.2.1 Transformation of the mission problem

The RH controller's model captures aspects of the two-dimensional battlefield with N UAVs and M fixed, target sites. The ith target's location belongs to a set of target sites $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_M\}$, and the jth UAV at time t has a position $\mathbf{x}_j(t) \in \Re^2$ in the set $\mathbf{X}(t)$. Cassandras uses vehicles' headings as the control variable in his RH model. The vehicle heading for the jth UAV at time t is given by $u_j(t) \in (0,360]$. The UAVs travel at the same, constant velocity C such that

$$\dot{x}_j(t) = C \begin{bmatrix} \cos u_j(t) \\ \sin u_j(t) \end{bmatrix} \tag{3.9}$$

As before, UAVs may only strike targets $\mathbf{Y}$ within constraints of fuel (D), munitions (O), and strike attempts (*max_strikes*). Following SPUDD's reward function $R(\mathbf{S}_k) = \sum_{i=1}^{M} T_i W_i$, the value received for successfully destroying target i is given by W_i.

The RH planner solves optimization problems over a sequence of planning horizons. In mission time, these optimizations $[RP_1, \ldots, RP_P]$ occur at time points $[t_1, \ldots, t_P]$. Considering the kth optimization problem at time t_k, RP_k solves for the control vector $\mathbf{u}_k = [u_1(t_k),\ldots, u_N(t_k)]$. At time t_k, suppose that the N UAVs are assigned the headings $u_1(t_k),\ldots, u_N(t_k)$, which are intended to be maintained for a planning horizon denoted by H_k. Then, the position of a vehicle j at time $t_k + H_k$ is given by

$$\mathbf{x}_j(t_k + H_k) = \mathbf{x}_j(t_k) + C\begin{bmatrix}\cos u_j(t)\\ \sin u_j(t)\end{bmatrix}H_k$$
$$\Rightarrow \mathbf{x}_j(t_k + H_k) = \mathbf{x}_j(t_k) + \dot{\mathbf{x}}_j(t_k)H_k \tag{3.10}$$

The earliest time that vehicle j could reach a target i, starting at a time t_k with a heading assignment $u_j(t_k) \in \mathbf{u}_k$ and moving directly to target site i from the point $\mathbf{x}_j(t_k + H_k)$ is given by

$$\tau_{ij}(t_k, \mathbf{u}_k) = (t_k + H_k) + \frac{\left\|\mathbf{x}_j(t_k + H_k) - \mathbf{y}_i\right\|}{C} \tag{3.11}$$

$\|\cdot\|$ denotes the Euclidean norm. The optimization problem is to maximize the reward obtained by vehicle j when it reaches target i at time $\tau_{ij}(t_k, \mathbf{u}_k)$ given a heading vector $\mathbf{u}_k$.

3.3.2.2 Formulation of the optimization problem

The optimization problem incorporates reward valuations and an assignment probability function. As noted earlier, W_i values the reward for the successful destruction of target i.

Next, for N>1, a relative distance function $\delta_{ij}(\mathbf{X}(t_k))$ gives the proximity of vehicle j to target i's position y_i in relation to the positions of the UAVs $\mathbf{X}(t_k) = \{\mathbf{x}_1(t_k),...,\mathbf{x}_N(t_k)\}$ at time t_k. This relative distance function is defined as

$$\delta_{ij}(\mathbf{X}(t_k)) = \frac{\left\|\mathbf{x}_j(t_k) - \mathbf{y}_i\right\|}{\sum_{l=1}^{N}\left\|\mathbf{x}_l(t_k) - \mathbf{y}_i\right\|} \tag{3.12}$$

For N UAVs and M targets, a normalized relative proximity function $q_{ij}(\delta_{ij})$ is any monotonically nonincreasing function of δ_{ij} such that

$$q_{ij}(0)=1,\ q_{ij}\left(\frac{1}{N}\right)=\frac{1}{N},\ \lim_{\delta_{ij}\to 1} q_{ij}(\delta_{ij})=0 \tag{3.13}$$

The normalized relative proximity function $q_{ij}(\delta_{ij})$ can be interpreted as the probability that target i is assigned to vehicle j at a time t [27]. If the Euclidean distance from a particular target i to each of the N vehicles is the same, $q_{ij}\left(\frac{1}{N}\right)=\frac{1}{N}$ ensures that the vehicles have an equal probability of assignment.

As relative distances vary during the mission, UAVs are attracted to the vicinity of target sites and are eventually assigned to them by virtue of their proximity [10]. For a particular UAV, the relationship of a target's attractive weighting to its proximity is determined by the normalized relative distance function $q_{ij}(\delta_{ij})$. In a two vehicle (N=2) scenario, a convenient $q_{ij}(\delta_{ij})$, which satisfies the conditions set in Equation 3.13 and was used in [27], is

$$q_{ij}(\delta_{ij})=\begin{cases}1 & \text{if } \delta_{ij}\le\Delta_i\\ \dfrac{1}{1-2\Delta_i}[(1-\Delta_i)-\delta_{ij}] & \text{if } \Delta_i<\delta_{ij}\le 1-\Delta_i\\ 0 & \text{if } \delta_{ij}>1-\Delta_i\end{cases} \tag{3.14}$$

where $\Delta_i\in[0,\frac{1}{2})$ is an adjustable threshold that can be interpreted as target i's "capture radius": if a vehicle j is close enough to i as to satisfy $\delta_{ij}\le\Delta_i$, then it is committed to visit i [10]. For simplicity, assume $\Delta_i=\Delta$ for all targets i.

The value of the normalized relative proximity function at the end of the planning horizon (i.e. at $t = t_k + H_k$) for a particular heading control vector $\mathbf{u}_k$ is defined as

$$\tilde{q}_{ij}(\mathbf{u}_k,\mathbf{X}(t_k))=q_{ij}\left[\delta_{ij}(\mathbf{X}(t_k+H_k))\right] \tag{3.15}$$

where $\mathbf{X}(t_k+H_k)$ is given by Equation 3.10.

Next, for UAV j at time t_k, define the available fuel as $f_j(t_k)$ and available munitions as $g_j(t_k)$. Assuming that fuel is allotted in time increments and is consumed at a constant rate of 1 fuel unit per time unit, UAV j's remaining fuel after a planning horizon H_k is $f_j(t_k) - 1 \cdot H_k$, which follows the correlation of SPUDD's move action costs to distance. Each UAV is initialized with a fuel quantity D, such that $f_j(t_0) = D$.

At the end of the planning horizon H_k, the UAV's remaining munitions are $g_j(t_k) - g_j(t_k + H_k)$. The available munitions for vehicle j at time t_k are computed as $g_j(t_k) = O - \sum_{i=1}^{M} z_{ij}(t_k)$ where O is the initial number of munitions allocated each UAV and $z_{ij}(t_k)$ gives the number of strikes that has been made by UAV j on target i by time t_k. For the RH controller examined in this thesis, the number of strikes executed on a particular target i by vehicle j is the minimum of the maximum strike attempts *max_strikes* and the available munitions of vehicle j (i.e. $z_{ij}(t_k) = \min\{ max\_strikes, g_j(t_k) \}$). The optimization problem RP_k that solves for the UAV heading vector $\mathbf{u}_k$ at time t_k is

$$\begin{aligned} &\max_{\mathbf{u}_k} \sum_{i=1}^{M} \sum_{j=1}^{N} \left[c_1 \cdot W_i \cdot \tilde{q}_{ij}(\mathbf{u}_k, \mathbf{X}_k) + c_2 \cdot f_j(t_k + H_k) + c_3 \cdot g_j(t_k + H_k) \right] \\ &\text{such that} \quad f_j(t_k) - 1 \cdot H_k \geq 0, \quad g_j(t_k) - g_j(t_k + H_k) \geq 0 \end{aligned} \tag{3.16}$$

where the normalized relative proximity function is given by $\tilde{q}_{ij}(\mathbf{u}_k, \mathbf{X}_k)$, fuel of vehicle j is given by $f_j(t_k + H_k)$, and munitions of vehicle j is given by $g_j(t_k + H_k)$. The parameters are weighted by factors c_1, c_2, and c_3, respectively. In addition, the optimization is subject to constraints that ensure the vehicles have sufficient fuel and munitions.

Based on available state information, an optimal $\mathbf{u}_k$ is derived for Equation 3.16. The N UAVs follow this control for an action horizon h_k. The value of h_k is determined either by the occurrence of an unexpected event at time $t_e \in (t_k, t_k + h_k)$, which sets h_k =

$t_e - t_k$, or by simply updating the control at predefined intervals. As a result, the times of planning optimization $[t_1, \ldots, t_P]$ can be a random sequence.

The optimization problem $\overline{RP}_k$ is fully specified by selecting the functions W_i and $q_{ij}(\delta_{ij})$ along with H_k to define $\tau_{ij}(t_k, \mathbf{u}_k)$ and h_k. The next section discusses the criticality of the planning horizon H_k to obtain desirable properties for this RH controller. $\overline{RP}_k$ is solved through standard nonlinear programming techniques in which multiple local optima may generally exist. As an approximation, the possible heading assignments of the vehicles $u_j(t) \in (0,360]$ are discretized into 10^o increments. The problem is solved as an on-line control that responds to stochastic events, such as the elimination of UAVs or targets; however, the problem setting can also be used for a priori planning of vehicle trajectories [27].

3.3.2.3 Stationary vehicle-to-target assignments

The optimization problem shown in Equation 3.16 solves for the joint-headings $\mathbf{u_k}$ of the participating UAVs at time t_k. This joint-heading assignment does not explicitly require the resultant trajectories to be stationary or characterized by an ultimate assignment of vehicles-to-targets. Indeed, there is no express constraint imposed on $\overline{RP}_k$ to assign a vehicle to a target site of the form $\mathbf{x}_j(t_k + H_k) \in \mathbf{Y}_k$ or $\frac{\mathbf{y}_i - \mathbf{x}_j(t_k)}{\left\|\mathbf{y}_i - \mathbf{x}_j(t_k)\right\|} = \frac{\mathbf{x}_j(t_k + H_k) - \mathbf{x}_j(t_k)}{\left\|\mathbf{x}_j(t_k + H_k) - \mathbf{x}_j(t_k)\right\|}$, which force a vehicle to either be at a particular point y_i by a certain time or to set a heading to it [28].

Cassandras and Li showed that the length of the planning horizon H_k determines the stationary properties of the receding horizon controller [10, 27]. They found that the length of the planning horizon should be based on the shortest distance of UAV-target pairs. Since UAVs might be lost and targets might be destroyed over the course of the mission, $\mathbf{X}_k$ and $\mathbf{Y}_k$ reflect, respectively, the participating UAVs and targets at time t_k. Assuming the UAVs travel at a constant velocity C, the planning horizon H_k is

$$H_k = \min_{j \in \mathbf{X}_k, i \in \mathbf{Y}_k} \left\{ \frac{\left\| \mathbf{y}_i - \mathbf{x}_j(t_k) \right\|}{C} \right\} \tag{3.17}$$

The RH controller provides stationary vehicle-to-target assignments for planning horizons that follow Equation 3.17 without explicit enforcement of such a constraint.

3.3.2.3 Cooperative assignment for a two-UAV, two-target scenario

This section analytically examines the RH controller's behavior in an example mission. Consider the system shown in Figure 3-16 with two UAVs initialized at the coordinates (0,0) and two target sites at the coordinates (10,0) and (0,10). Suppose the UAVs are tasked to visit the two target sites with the least amount of fuel. The controller has three options: (a) assign both UAVs to a heading of 360, routing them together to target y_1, (b) direct UAV x_1 and x_2 to the headings 360 and 090, respectively, or (c) assign both UAVs to a heading of 090, routing them together to target y_2. Options (a) and (c) are equivalent because of the scenario's symmetry, so case (c) is disregarded in the proceeding analysis. Notice that heading assignment (b) ultimately achieves the mission faster than option (a) by cooperatively dispatching vehicle x_1 to target y_1 and vehicle x_2 to target y_2.

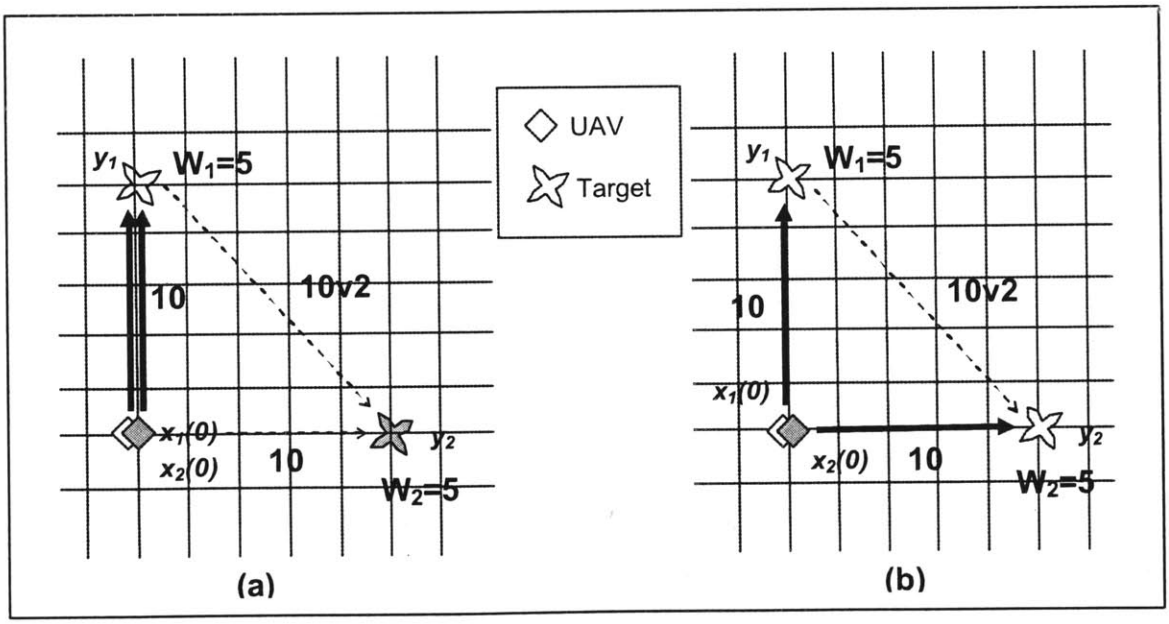

Figure 3-16 Two possible heading assignments for a two-UAV, two-target scenario

The RH controller selects a joint-heading assignment that maximizes the expected reward over the planning horizon. For scenarios with two UAVs and M targets, Li showed that the maximization problem in Equation 3.16 is equivalent to the minimization [27]:

$$\min_{\mathbf{u}_k} \sum_{i=1}^{N} W_i \left(\left\| \mathbf{x}_1(t_k + H_k) - \mathbf{y}_i \right\| q_{i1} + \left\| \mathbf{x}_2(t_k + H_k) - \mathbf{y}_i \right\| q_{i2} \right) \tag{3.18}$$

Assuming the UAVs travel at velocity C of one distance unit per time unit; the planning horizon H_k is ten time units (based on the shortest distance between vehicle-target pairs). The first UAV to visit target i obtains a reward $W_i = 5$. Using a normalized relative proximity function $q_{ij}(\delta_{ij})$ that satisfies Equation 3.13 [27], the value of the trajectory assignment illustrated in Figure 3-16a is

$$\begin{aligned}
[u_1 = 360, u_2 = 360] &= W_1 \left(\left\| \mathbf{x}_1(t_k + H_k) - \mathbf{y}_1 \right\| q_{11} + \left\| \mathbf{x}_2(t_k + H_k) - \mathbf{y}_1 \right\| q_{12} \right) \\
&\quad \ldots + W_2 \left(\left\| \mathbf{x}_1(t_k + H_k) - \mathbf{y}_1 \right\| q_{21} + \left\| \mathbf{x}_2(t_k + H_k) - \mathbf{y}_1 \right\| q_{22} \right) \\
&= 5\left((0.0)(0.5) + (0.0)(0.5)\right) + 5\left((14.1)(0.5) + (14.1)(0.5)\right) = \underline{\underline{70.5}}
\end{aligned}$$

Similarly, the value of the joint-vehicle heading assignment depicted in Figure 3-16b is

$$\begin{aligned}
[u_1 = 360, u_2 = 090] &= W_1 \left(\left\| x_1(t_k + H_k) - y_1 \right\| q_{11} + \left\| x_2(t_k + H_k) - y_1 \right\| q_{12} \right) \\
&\quad \ldots + W_2 \left(\left\| x_1(t_k + H_k) - y_1 \right\| q_{21} + \left\| x_2(t_k + H_k) - y_1 \right\| q_{22} \right) \\
&= 5\left((0.0)(1.0) + (14.1)(0.0)\right) + 5\left((14.1)(0.0) + (0.0)(1.0)\right) = \underline{\underline{0.0}}
\end{aligned}$$

The receding horizon controller selects the optimal joint-vehicle heading assignment that minimizes the optimization problem of Equation 3.18. For this example, the minimum value of UAV heading assignments over the initial planning horizon is zero. The RH controller selects the heading assignment $\mathbf{u}_0$ = {360,090}, which ultimately directs vehicle x_1 to target y_1 and vehicle x_2 to target y_2.

This assignment reveals the controller's cooperative task distribution. Cooperation enhances the behavioral performance of the UAVs. Positive cooperative behavior is defined by improving some performance characteristic with each additional participant. Clustering both UAVs to a single target provides no advantage to an analogous one-UAV scenario. By assigning the UAVs to separate headings, the two vehicles are able to cooperatively complete their task in less time than possible with a single vehicle.

3.4 Relationship of SPUDD and RH planners

3.4.1 Problems of optimization

Two critical characteristics that distinguish the RH controller from the SPUDD planner are its (1) on-line and (2) greedy design. The cooperative behavior engendered by each planner in the form of vehicle-to-target assignments and action sequencing affect mission performance, which includes rewards accumulated for destroying targets and costs incurred for losing UAVs. The dissimilar approaches also affect computational performance measures, including planning time and memory consumption.

The SPUDD and RH planners solve planning problems that differ in the scope of optimization. Whereas the SPUDD planner searches for an optimal policy with contingences for a global state space, the RH controller sequentially selects actions that provide a maximum expected reward within each localized planning horizon. These optimization disparities induce the SPUDD planner to construct its global policy prior to mission execution, and the RH controller to react to battlefield events in real-time. As noted in Section 3.3.1, the RH scheme dynamically optimizes over a sequence of planning horizons and executes decisions over shorter action horizons, which ultimately result in stationary vehicle-to-target assignments and sequencing. On the other hand, the SPUDD planner embeds these attributes into its policy before the simulated mission has even begun.

Both the SPUDD and RH controllers construct plans for a fully-observable battlefield in which states of system variables, such as a UAV's remaining fuel, are accurately known. The RH optimization described by Equation 3.16 though is a

reduction of the problem considered by SPUDD. The system components represented in the models of the SPUDD and RH controllers are compared in Figure 3-17. SPUDD relies on a system model that captures the costs for transits and strikes, rewards for satisfying mission objectives, UAV strike success capability, and UAV attrition probability. Indeed, detailed, stochastic state transition models permit SPUDD to discover contingencies of maximum expected utility through the global state space. The RH controller does not benefit from such a complete model of the battlefield system. Although the RH controller equivalently models cost and reward valuations, it is not provided probabilistic representations of the UAVs' capabilities and attrition risks. These shortcomings and its myopic planning horizon induce the RH controller to behave greedily.

The differences in the planners' system models lead to distinctive mission behaviors and computation requirements. Although contingency planning benefits mission performance, the practicality of SPUDD's optimal policy search is cursed by increasing system dimensionality, as described in Section 3.2.3.4. Oppositely, the RH controller divides SPUDD's global computation into a sequence of optimizations over localized planning horizons. The RH controller's reduced system model and narrowed horizon streamline planning computation, but trade behavioral optimality for approximation.

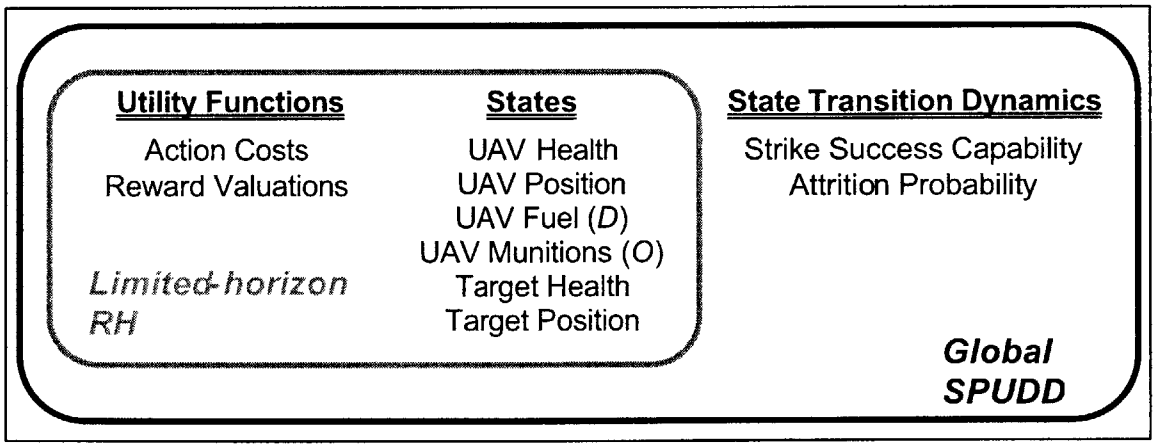

Figure 3-17 System model comparison of the SPUDD planner and RH controller

To evaluate the robustness of these algorithms, performance is evaluated in scenarios where certain model parameters (of Figure 3-17) inaccurately represent the actual system. Indeed, a model's level of accuracy and/or specification impacts the value of contingencies that a planner can provide.

3.4.2 Optimal versus greedy controllers

The preceding section showed that the SPUDD and RH planners solve differing problems of optimization.

Greedy algorithms sometimes perform optimally in a subset of systems, but may behave poorly in others. In addition, they tend to require fewer computational resources than elaborate exact approaches. SPUDD guarantees an optimal policy; however, the planner is plagued by the curse of dimensionality. Although the SPUDD planner utilizes efficient model representations, it still searches and provides contingencies for a universal state space. Oppositely, the RH controller performs a sequence of optimizations over shorter planning horizons and ignores aspects of the system's dynamics. Indeed, the RH controller's abridged system model and myopic horizon truncate the planning problem.

The following section briefly describes the quantitative relationship of optimal MDP-based controllers and heuristic greedy controllers from the work of [19, 37].

3.4.2.1 Valuation of plans

As described in Section 3.2.2, MDP-based planners use the expected total discounted reward as a basis for optimality to compare possible policies. The value function $V_\pi \in \Re^N$ gives this quantity for following a policy p from an initial state $\mathbf{S}_i$. The expected value $V_\pi(\mathbf{S}_i)$ of a policy p for an initial state $\mathbf{S}_i$ satisfies [33]:

$$V_\pi(\mathbf{S}_i) = R(\mathbf{S}_i) + \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \pi(\mathbf{S}_i)) \cdot V_\pi(\mathbf{S}_j)$$

Greedy controllers seek actions that maximize this expected value. The greedy valuation function is defined as [19]:

$$\textbf{Greedy}\left[V_\pi(\mathbf{S}_i)\right] = \arg\max_{\mathbf{A}_k \in \mathbf{A}} \left\{ R(\mathbf{S}_i) + \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \pi(\mathbf{S}_i)) \cdot V_\pi(\mathbf{S}_j) \right\} \tag{3.18}$$

To an extent, SPUDD is greedy. The SPUDD planner finds an optimal policy p^* that is a greedy selection of actions with respect to the optimal value function V^*. The planner seeks an optimal policy with contingencies for a global state space with a maximum expected discounted reward. That is,

$$\pi^* = \textbf{Greedy}[V^*] \tag{3.19}$$

For RH, the targets (or rewards) visible in a particular optimization is dependent on the size of the planning horizon. The limited planning horizon and reduced system dynamics model afford an approximate value function $\hat{V}_\pi$, instead of the optimal value function V^*. The RH controller chooses a joint course of action that offers an immediate reward over each planning horizon without regard to the UAVs' strike success capabilities or attrition risks. As a result, the greedy plan $\hat{\pi} = \textbf{Greedy}[\hat{V}_\pi]$ might be suboptimal with respect to SPUDD's optimal policy. The deficiency sustained by following a suboptimal policy $\hat{\pi}$ instead of p^* is bounded by the error on the approximate valuation function $\hat{V}_\pi$ [37]. This approximation or Bellman error E for a particular value function V_π that may be suboptimal is [38]

$$E(V_\pi) = \max_{\mathbf{S}_i \in \mathbf{S}} \left| V_\pi(\mathbf{S}_i) - \max_{\mathbf{A}_k \in \mathbf{A}} \left\{ R(\mathbf{S}_i) + \beta \sum_{\mathbf{S}_j \in \mathbf{S}} P(\mathbf{S}_j \mid \mathbf{S}_i, \mathbf{A}_k) \cdot V_\pi(\mathbf{S}_j) \right\} \right| \tag{3.20}$$

The optimal value function V^* has a Bellman error of zero. Williams and Baird bounded the loss of following a greedy, suboptimal policy $\hat{\pi}$ rather than an optimal policy p^* as [37]:

$$V^*(\mathbf{S}_i) - V_{\hat{\pi}}(\mathbf{S}_i) \leq \frac{2\beta \cdot E(\hat{V}_\pi)}{1-\beta}, \forall \mathbf{S}_i \in \mathbf{S} \qquad (3.21)$$

where $\hat{V}_\pi$ is the approximate value function, V^* is the optimal value function for the policy p^*, and $V_{\hat{\pi}}(\mathbf{S}_i)$ is the true value of the suboptimal policy $\hat{\pi}$. The bound depends on the approximation quality of the value function and the discount factor β.

Cassandras showed that the RH controller's solutions can match a reward upper bound. Considering a fully deterministic environment, the upper bound is provided by an exhaustive search over all possible trajectories from given initial vehicle positions, assuming straight-line paths between target sites [10]. That is, vehicle j's trajectory is specified as a sequence of targets to be visited. In a two-UAV, six-target scenario, RH sometimes yields rewards that are equivalent to the exhaustive search, however, the controller performs suboptimally in the presence of multiple local optima and instabilities in the form of oscillating heading vectors [10].

[*Except for this sentence, this page intentionally left blank*]

4 Results

"Planning without action is futile, action without planning is fatal"

– Unknown

4.1 Overview

This thesis evaluates the SPUDD and RH planning algorithms in a representative set of scenarios. As described in Chapter 3, each algorithm has extensive customizable attributes. Although the results are limited to specific problem instances, they provide characterizations of each algorithm's capabilities and limitations.

4.2 Test cases

The scenarios emulate visit-and-destroy, cooperative UAV missions. Scenarios vary numbers and positions of UAVs and targets, rewards obtainable for destroying targets, available fuel and munitions, and probabilities of UAV strike successes and attrition. These test cases experimentally reveal the strengths and weakness of the algorithms. As noted in Section 3.4.1, the SPUDD and RH planners utilize differing system models. The SPUDD planner searches for an optimal policy with contingencies for a global state space, while the RH controller sequentially re-plans over more localized regions.

Dissimilar system representations and optimization horizons contribute to particular mission and computational performances. For each scenario, mission success is assessed on the basis of rewards accumulated for destroying targets and costs incurred for UAV losses. Planning time is the primary metric for describing the computational requirements of each planner. Monte-Carlo simulations evaluate the planners' average-case performances in the presence of stochastic system components. For the examined scenarios, the SPUDD planner generates policies with a discount factor of 0.8 to moderate its bias on mission behavior, as described in Section 3.2.5.

Extending nominal tests, the planners are examined in situations where system models are inaccurate or incomplete representations of the true battlefield. These sensitivity studies indicate each algorithm's robustness to real world conditions, where a planner's model is typically an imprecise estimate of reality. The test scenarios also suggest each planner's scalability to larger missions that include greater numbers of UAVs and targets.

4.2.1 Scenario 1

This scenario captures the assignment and sequencing behavior of the SPUDD and RH controllers in a straightforward, cooperative mission. The initial system is comprised of two UAVs and eight targets at the relative positions shown in Figure 4-1. The targets are valued equally, and the fuel cost for moving between target sites is proportional to distance. Each UAV is initialized with 7 fuel units and 6 ordnances. UAVs are invincible; however, the probability of a successful strike degrades from 0.5 to 0.2 to 0.1 with each strike attempt on a particular target. The *max_strikes* constraint limits the number of strike attempts allowable on a single target to three. Figure 4-1 highlights the nominal path of the two UAVs. The mission concludes when every target site has been destroyed or all the UAVs have expended their fuel and/or munitions. In nominal conditions, the UAVs have sufficient fuel and munitions to visit their all of the target sites and perform an average of 1.5 strikes at each site.

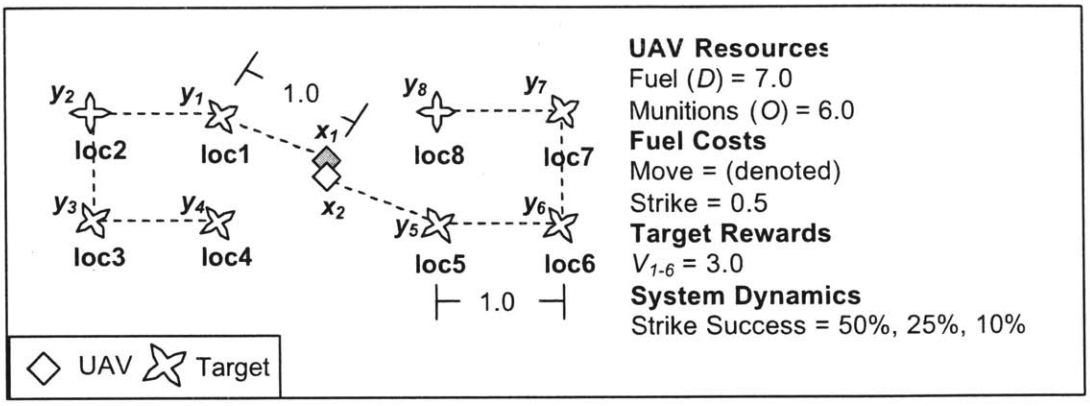

Figure 4-1 Scenario 1: hitial system with two UAVs and eight targets. Dotted lines denote the nominal target assignment sequence for both SPUDD and RH controllers

4.2.1.1 SPUDD and RH comparison

Figure 4-2 shows typical strike sequences of the two UAVs on a per planner basis. Both the SPUDD and RH controllers select identical vehicle-to-target assignments. In average conditions, the planners assign UAV x_1 to targets y_1, y_2, y_3, y_4 and x_2 to y_5, y_6, y_7, y_8. The algorithms critically differ in the number of strikes enacted on a particular target. Guided by the SPUDD planner, the UAVs execute a maximum of two strike attempts on each target. SPUDD's strategy seeks to obtain rewards from the target sites at the higher 0.5 and 0.2 strike success probabilities. Rather than strike at a 0.1 success probability, the planner advances the UAVs to other more promising target sites. On the other hand, the RH controller's model disregards state transition dynamics, which include the UAVs' strike capabilities. As a result, the RH planner repeatedly strikes each target until either a reward is acquired or constrained by *max_strikes*.

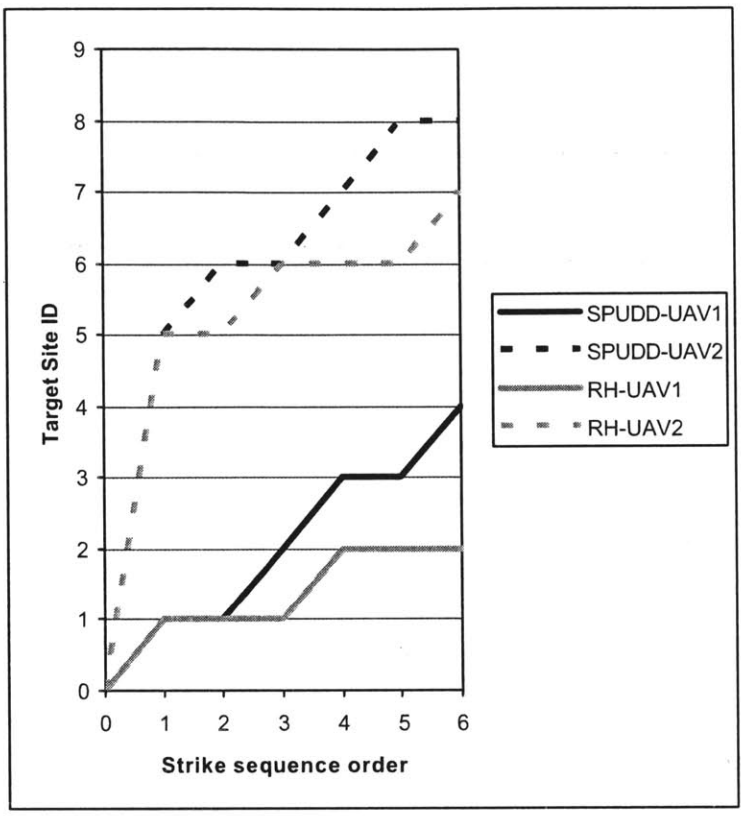

Figure 4-2 Scenario 1: Typical target sequencing for each UAV on a per planner basis

The probabilistic-nature of strike successes introduces uncertainty into the system. This stochastic component causes variation in possible vehicle-to-target strike sequences. Monte-Carlo simulations provide average-case results over five hundred samples. Figure 4-3 depicts the number of times each target was visited, struck, and destroyed by the SPUDD planner. The plot indicates that SPUDD's global policy balances its strikes amongst the eight, equally-valued targets. Per iteration, each target is visited 1.0 times, struck about 1.5 times, and destroyed 0.6 times. This behavior indicates the strength of SPUDD's complete and fully-observable system model, which permits the construction of a policy with contingencies to manage uncertainty. The UAVs destroy 4.72±0.06 targets and accumulate a reward of 14.16±0.18, per iteration.

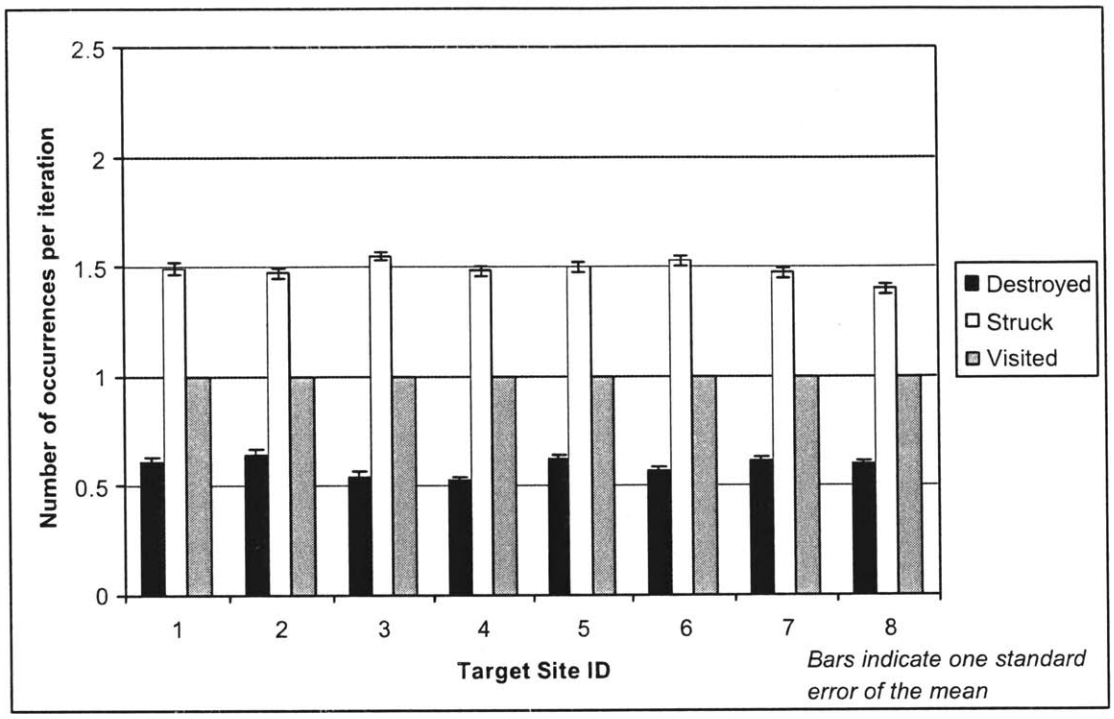

Figure 4-3 Scenario 1: Mission statistics for SPUDD policy over a Monte-Carlo simulation with five hundred samples

The mission performance of the RH controller was similarly measure by Monte-Carlo simulation. The results, shown in Figure 4-4, expose the greedy nature of the algorithm. The UAVs repeatedly strike target sites until successful or constrained by *max_strikes*. Because of fuel and munitions limitations, a UAV that performs more than two strikes on a single target is unable to attack all of its potential targets. UAVs x_1 and x_2 strike their first targets y_1 and y_5, respectively, about 2.25 times per iteration. By expending extra resources on these targets, UAVs x_1 and x_2 visit y_4 and y_8, respectively, only about 50 percent of the time. The RH scheme destroys 4.30±0.08 targets and claims a reward of 12.90±0.24, per iteration. A two-sample, one-tailed Student's t-Test indicates that more targets were destroyed with the SPUDD planner than the RH controller ($p<0.05$). SPUDD effectively manages the constraints of fuel and munitions and the uncertainty associated with strike successes by generating a policy with contingencies for a universal state space. Indeed, RH's strategy performs worse than SPUDD because resources are wasted on excessive strike attempts that have a low probability of success.

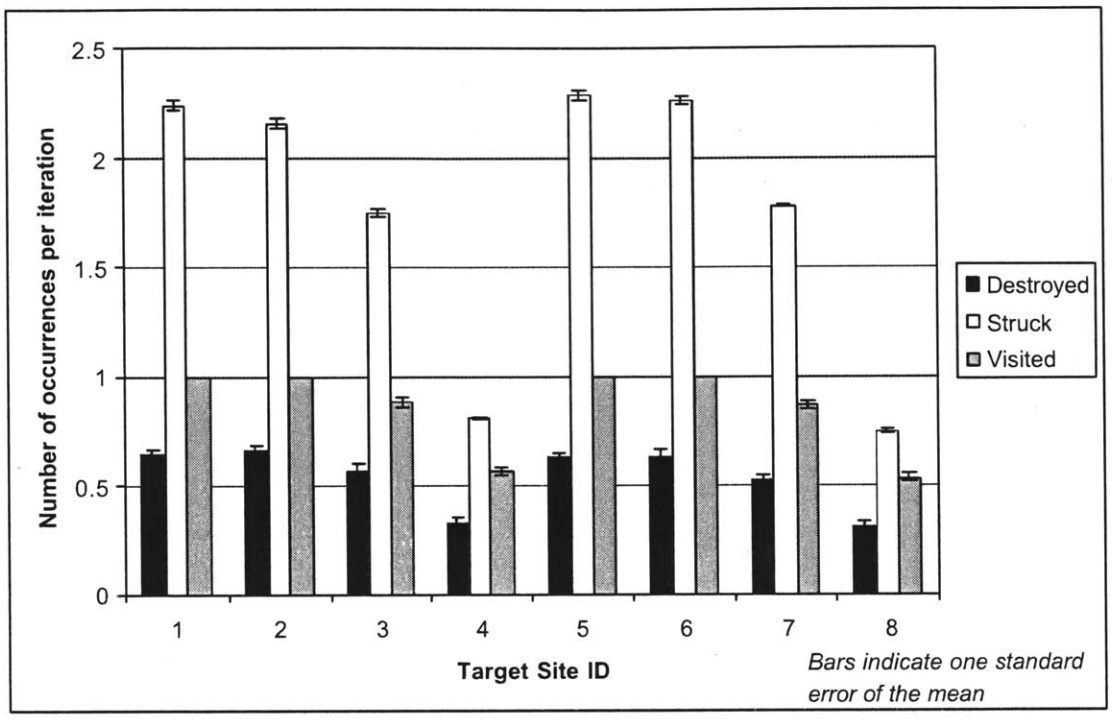

Figure 4-4 Scenario 1: Mission statistics for receding horizon strategy over a Monte-Carlo simulation (n=500)

4.2.1.2 SPUDD sensitivity to UAV capability model

Unlike the RH controller, the SPUDD planner models the uncertainty associated with the system's dynamics. In particular, SPUDD's model represents the UAVs' degradation in strike success probability as reducing from 0.5 to 0.2 to 0.1 over three strike attempts. Suppose the SPUDD planner inaccurately estimates the success probability of the first strike attempt. For example, a first strike success transition probability error of -0.3 represents the UAV strike capability degradation as 0.2 to 0.2 to 0.1.

SPUDD's sensitivity to such errors is shown in Figure 4-5. Note that these results are dependent on the attributes of this problem instance, including reward valuations, action costs, fuel, munitions, etc. The planner formulates three types of policies depending on whether the first strike success probability is underestimated, comparable, or overestimated with respect to its true value of 0.5. Figure 4-5 depicts typical strike sequences induced by each policy.

Policy 1: Underestimate. For first strike transition probability errors of -0.2 and -0.3, the UAVs destroy about 4.3 targets per iteration. Since the probabilities of the UAVs' first and second strike attempts are similar, SPUDD formulates a somewhat greedy policy. Exhibiting the same performance as RH, the UAVs repeatedly strike each target until successful or constrained by *max_strikes*. UAVs x_1 and x_2 strike their first targets (y_1 and y_5, respectively) about 2.25 times, their second targets (y_2 and y_6, respectively) about 2.20 times, their third targets (y_3 and y_7, respectively) about 1.77 times, and their fourth targets (y_4 and y_8, respectively) about 0.78 times per iteration. The underestimated first strike success rate favors striking at least two times, rather than advancing to another site, because both options appear to offer similar success probabilities and the move action is costlier. As a result, however, the UAVs strike targets at the lower 0.2 and 0.1 strike success probabilities and lack sufficient resources to attack targets that have a greater probability of destruction.

Policy 2: Comparable. For first strike transition probability errors of -0.1 and +0.1, the UAVs destroy about 4.7 targets per iteration. SPUDD shows robustness by performing equivalently to these small model errors as with a true model. SPUDD's global policy balances its strikes amongst the eight equally-valued targets within the constraints of fuel and munitions, and each target is struck on average about 1.5 times per iteration.

Policy 3: Overestimate. For first strike transition probability errors of +0.2, +0.3, and +0.4, the seemingly large disparity between first and second strike success rates induces the UAVs to advance to their next targets after only one strike attempt. In these scenarios where its model overestimates the UAVs' first strike success probability, SPUDD destroys about 4.5 targets per iteration. Mission performance is better than that of underestimated capability models because the UAVs strike each target at least once. Still, the UAVs expend considerable resources while transiting between target sites. Instead of optimizing strike attempts, SPUDD directs the UAVs to strike each of their targets once on the first pass and attempt a second wave of attacks if resources permit. Typically, in the first pass, UAV x_1 strikes each target y_1, y_2, y_3, y_4 once and x_2 strikes each target y_5, y_6, y_7, y_8 once. In the second attack, UAV x_1 may revisit targets that were not destroyed in the order y_4, y_3, y_2, y_1, and UAV x_2 may revisit targets that were not

destroyed in the order y_8, y_7, y_6, y_5. Consequently, targets y_1 and y_5 are struck about 1.07 times, y_2 and y_6 are struck about 1.12 times, targets y_3 and y_7 are stuck about 1.30 times, and y_4 and y_8 are struck about 1.70 times, per iteration.

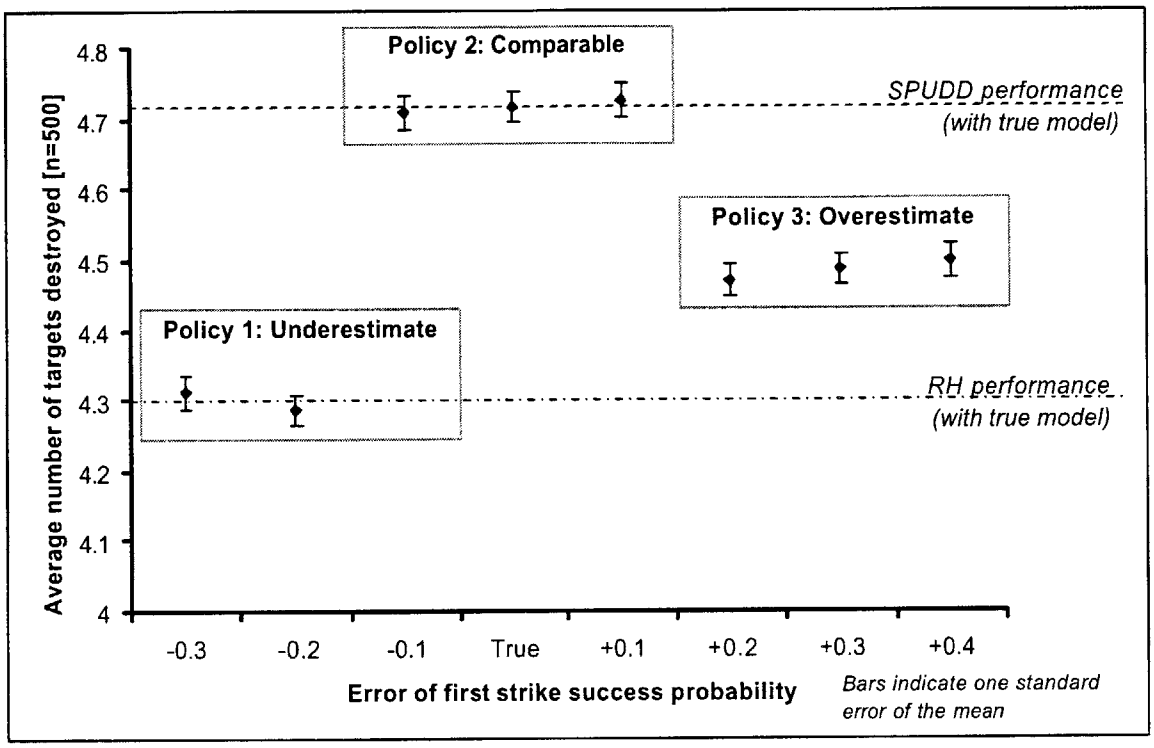

Figure 4-5 Scenario 1: SPUDD's sensitivity to the accuracy of the UAVs' first strike probability

4.2.1.3 SPUDD and RH sensitivity to partially-observable munitions

In nominal missions, the state of each UAV's available munitions is fully-observable to both the SPUDD and RH controllers. Consider a scenario in which the ordnance monitoring systems fail on the UAVs. As a result, the number of munitions perceived to be available for strike operations is inaccurate. Although the evaluated planner implementations do not explicitly handle partially-observable states, this test case considers each algorithm's robustness to such a situation. The models of both planners assume each UAV is initialized with six strike ordnances, even though each is actually mounted only with three. As before, the simulation ends when every target site has been destroyed or the UAVs have expended either their fuel or munitions.

The performance statistics for the SPUDD and RH controllers are shown in Figure 4-6 and Figure 4-7, respectively. The two UAVs have a total of six strike ordnances to use in the mission. Munitions, not fuel, are the overwhelming limitation in this scenario. The optimal strategy for such a munitions constraint is for each UAV to attempt only one strike on each target at the highest success probability (i.e. 0.5).

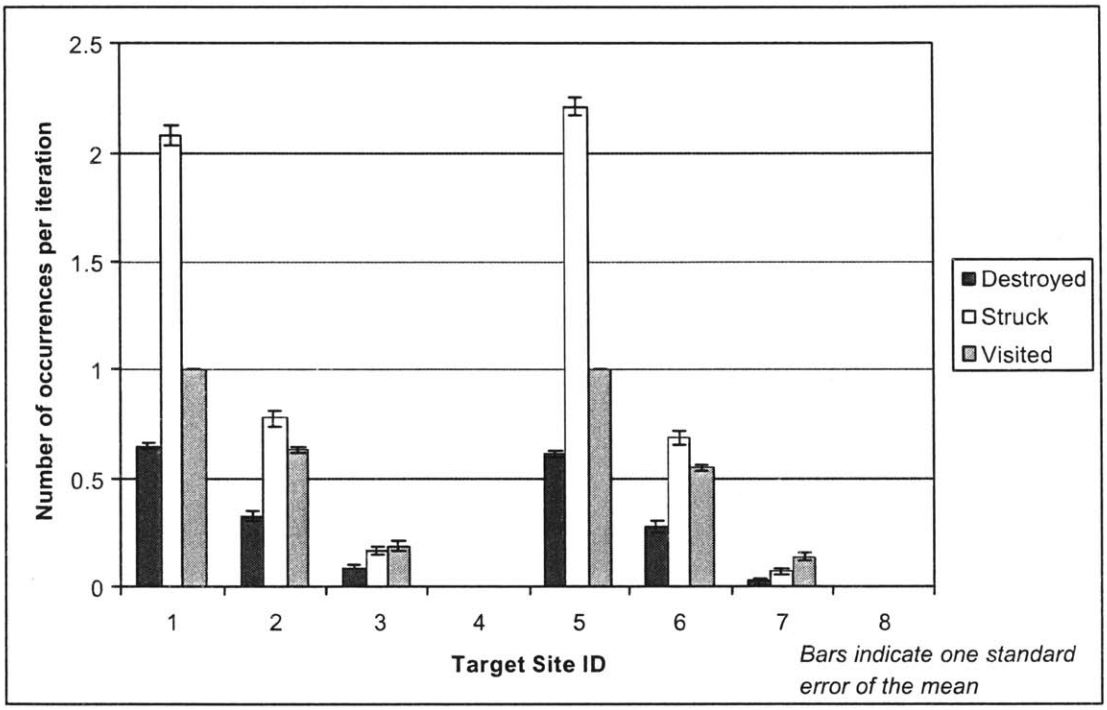

Figure 4-6 Scenario 1: Mission statistics for SPUDD policy with constraining, partially observable munitions states over a Monte-Carlo simulation (n=500)

With partially-observable munitions states, the SPUDD planner assumes the two UAVs possess a total of twelve ordnances. Consequently, the planner generates a policy equivalent to that of Section 4.2.1.1. SPUDD only guarantees the generation of an exact policy for the system model that it receives. Indeed, SPUDD's policy provides suboptimal contingencies because the UAVs have fewer munitions than modeled. Like before, the SPUDD policy directs the UAVs to attempt about 1.5 strikes on each target per iteration. The expenditure of munitions early in this mission; however, reduces the ordnances available to strike targets in the future. Consequently, targets y_1 and y_5 are

struck about 1.5 times, y_2 and y_6 are struck 1.3 times, and y_3 and y_7 are struck 0.3 times, per iteration. Targets y_4 and y_8 are not struck at all. Over a five hundred sample Monte-Carlo simulation, the SPUDD planner destroys an average of 2.58±0.06 targets and collects a reward of 7.74±0.18, per iteration.

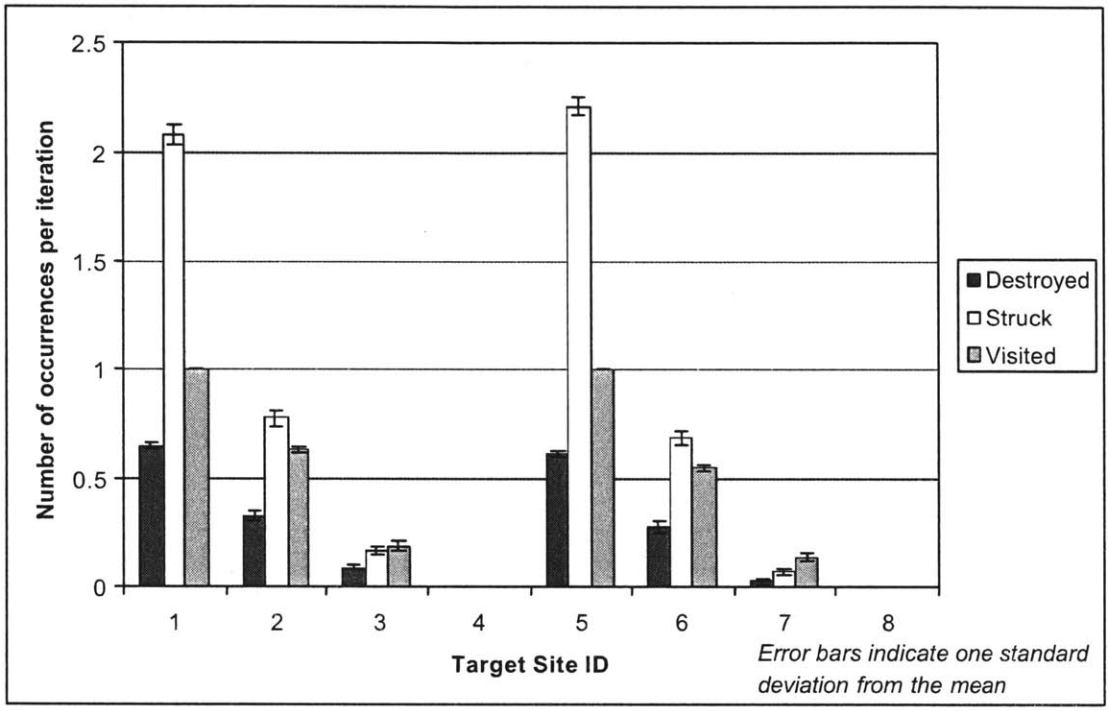

Figure 4-7 Scenario 1: Mission statistics for receding horizon strategy with constraining, partially observable munitions states over a Monte-Carlo simulation (n=500)

The receding horizon controller similarly suffers from the partially-observable, reduced supply of munitions. The number of ordnances available for later targets is even lower; however, because the RH planner greedily strikes until a UAV's target is destroyed or limited by *max_strikes*. The strikes attempted per iteration decline rapidly across the targets from about 2.1 times at y_1 and y_5 to 0.8 times at y_2 and y_6 to 0.1 times at y_3 and y_7 to 0.0 times at y_4 and y_8. The controller destroys 2.05±0.06 targets and collects a reward of 6.15±0.18, per iteration. A Student t-Test indicates that the SPUDD planner performs statistically better than the RH controller for restrictive, partially-observable munitions (p<0.05).

The SPUDD planner successfully destroyed more targets than the RH controller in the fully-observable missions (see Section 4.2.1.1) and the robustness study with partially-observable munitions states (see Section 4.2.1.3). Still, mission performance of both the SPUDD and RH planners significantly degrades with reductions in system observability or model completeness. These performance reductions are pronounced for the RH controller, which lacks contingency planning capability. Furthermore, the sensitivity trials (see Section 4.2.1.2) show that SPUDD's mission performance is adversely affected by inaccuracies in the uncertainty associated with UAV strike capabilities. Interestingly, while SPUDD destroys the greatest number of targets with a true system model, the planner performs better with an overestimation than an underestimation of the UAVs' first strike success probability ($p < 0.05$). The average targets destroyed per iteration for each test case are summarized in Figure 4-8.

	Targets Destroyed per Iteration
SPUDD	4.72±0.06
RH	4.30±0.08
SPUDD (partially-observable munitions)	2.58±0.06
RH (partially-observable munitions)	2.05±0.06

Figure 4-8 Scenario 1: Summary of average targets destroyed for each test case

SPUDD's mission performance advantages entail substantial computational resources. Whereas the RH controller's average, cumulative planning time is 43±5 seconds, SPUDD consumes 1.01 hours to construct a policy for the two-UAV, eight-target scenario. SPUDD-guided UAVs accomplish statistically more mission objectives than the RH controller ($p<0.05$); however, SPUDD's planning computation is about 100 times longer. The benefits of SPUDD's policy can be realized only if sufficient time exists for off-line, pre-mission planning. SPUDD's policy construction consumes a maximum of 233 MB of memory, while iterating through a possible action and reward space of 168,400 nodes. Still, SPUDD's ADD abstractions considerably condense the search space of classical value-iteration methods, which would require 112,037,630 equivalent nodes.

4.2.2 Scenario 2

This scenario extends the battlefield of Scenario 1 to three UAVs and nine targets. The initial, relative positions of the vehicles and targets are shown in Figure 4-9. Like Scenario 1, the targets are equally valued, and the cost for moving between target sites is proportional to distance. Additionally, the maximum number of strike attempts *max_strikes* on a single target is limited to three, and the UAVs' strike success probability degrades from 0.5 to 0.2 to 0.1 with each strike on a particular site.

In real world battlefields, UAVs may be lost due to malfunction, damage, maintenance, weather, etc. These losses are modeled as probability distributions on each UAV's health during operation. Namely, UAVs are lost 10 percent of the time in transits between sites and 25 percent of the time in strike executions. UAV attrition is assumed to occur *after* an action has been committed. For instance, the resultant system state after a strike action might include the destruction of the attacking UAV, struck target, or both.

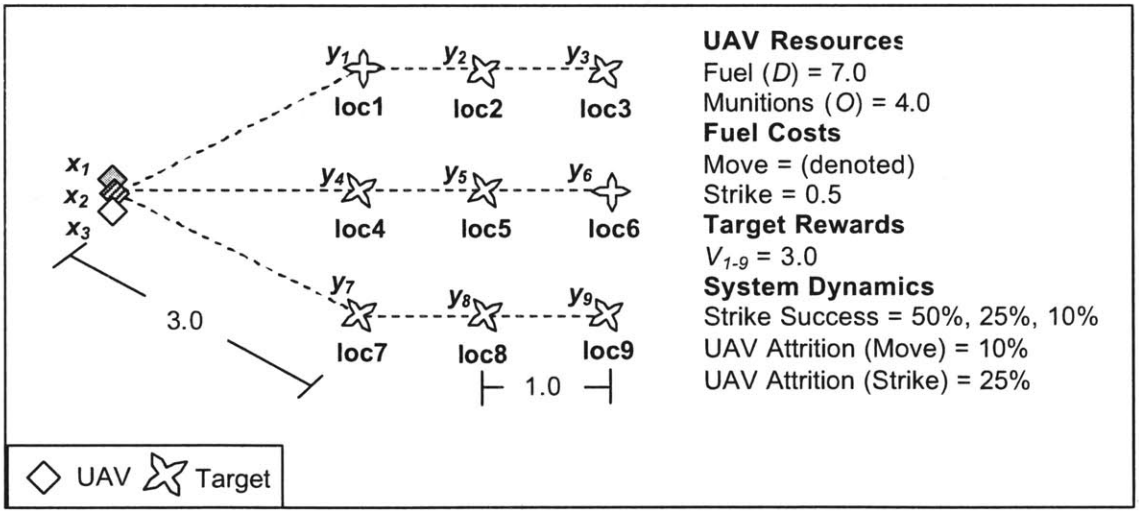

Figure 4-9 Scenario 2: Initial system with three UAVs and nine targets. Dotted lines denote the nominal target assignment sequence for both SPUDD and RH controllers

4.2.2.1 SPUDD and RH comparison

Figure 4-10 shows typical strike sequences of the two UAVs on a per planner basis. Both planners select identical vehicle-to-target assignments. In average conditions, the planners ultimately assign UAV x_1 to targets y_1, y_2, y_3; x_2 to y_4, y_5, y_6; and x_3 to y_7, y_8, y_9. SPUDD's policy directs the UAVs to execute a fewer number of strikes at each site to maximize the total number of sites struck. With complete models of the UAVs' strike success and attrition distributions, SPUDD's contingency-based policy manages uncertainty by capitalizing strikes on targets at high success probabilities, while mitigating risk to the UAVs. Oppositely, the RH controller neither models the UAVs' strike success capability nor loss probability. As a result, the RH planner greedily strikes targets until either a reward is acquired or limited by *max_strikes*.

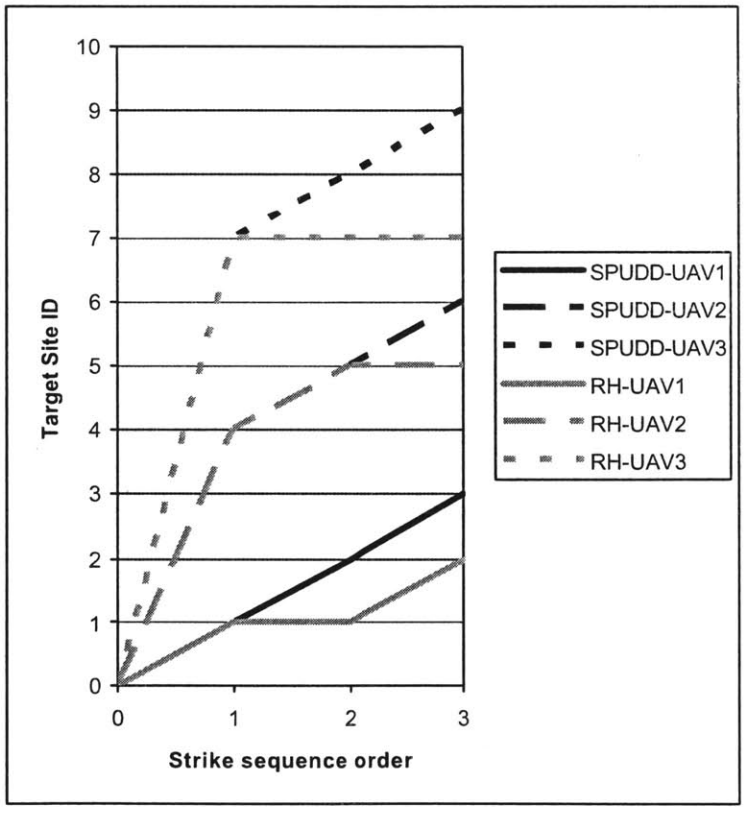

Figure 4-10 Scenario 2: Typical target sequencing for each UAV on a per planner basis

Figure 4-11 depicts the average number of times the UAVs visited, struck, and destroyed each target with the SPUDD planner. UAV attrition causes the targets to be unevenly struck. The policy directs the UAVs to strike once at each target site; however, UAVs are quickly lost through attrition during move and strike operations. On average, a reduced number of UAVs survive to attack their second target assignment and fewer yet to their third. Consequently, UAV x_1 visits target y_1 100 percent of the time, y_2 about 60 percent of the time, and y_3 40 percent of the time. Over a five hundred sample Monte-Carlo simulation, an average of 2.17±0.03 UAVs are lost and 2.68±0.06 targets are destroyed, per iteration. Although these values appear similar, a one-tailed, unpaired Student t-Test shows that more targets are destroyed than UAVs lost ($p<0.05$).

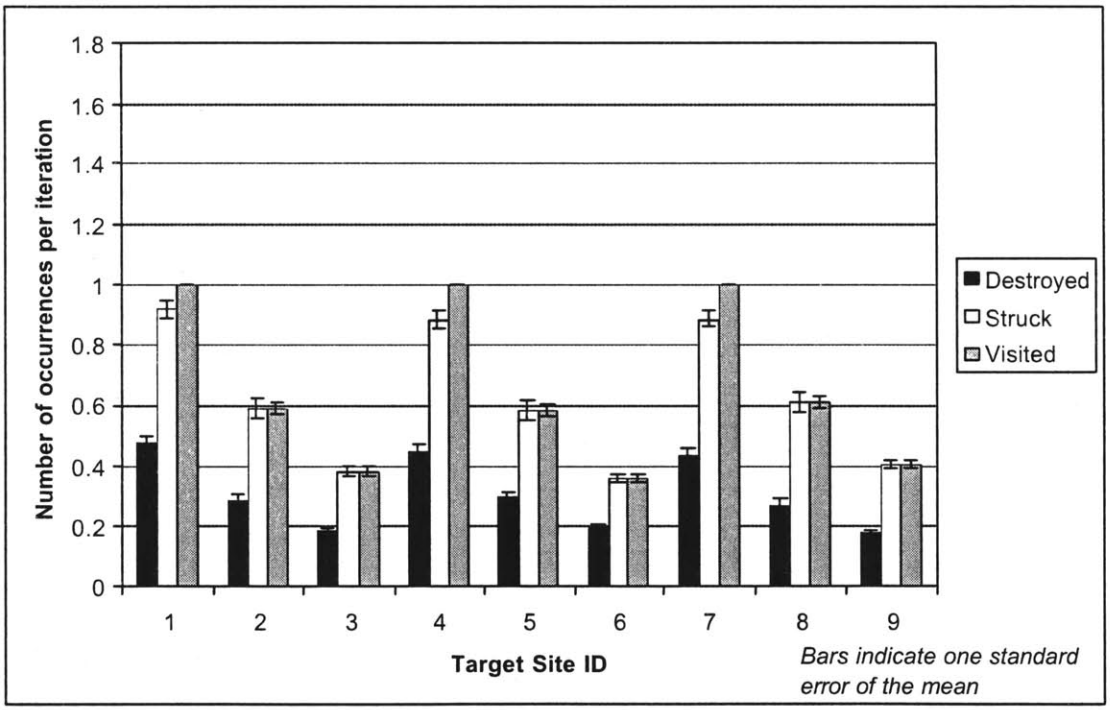

Figure 4-11 Scenario 2: Mission statistics for SPUDD policy over a Monte-Carlo simulation (n=500)

An equivalent Monte-Carlo simulation was performed to assess the mission performance of the RH controller in an identical scenario. The UAVs' mission performance, as shown in Figure 4-12, indicates the greediness of the controller's plans. As observed in previous

test cases, the UAVs repeatedly strike each target site until successful or constrained by *max_strikes*. The RH planner suffers without stochastic models of risk and success. The planner is unable to recognize the danger of vehicle attrition and diminished success probabilities for multiple strike attempts on a single target. As a result, even fewer UAVs survive to strike their targets. For the RH controller, UAV x_1 survives to visit target y_1 100 percent of the time, y_2 about 35 percent of the time, and y_3 about 10 percent of the time. The method loses 2.16±0.03 UAVs and destroys 2.40±0.07 targets, per iteration. Like SPUDD, more targets were destroyed than UAVs lost ($p<0.05$).

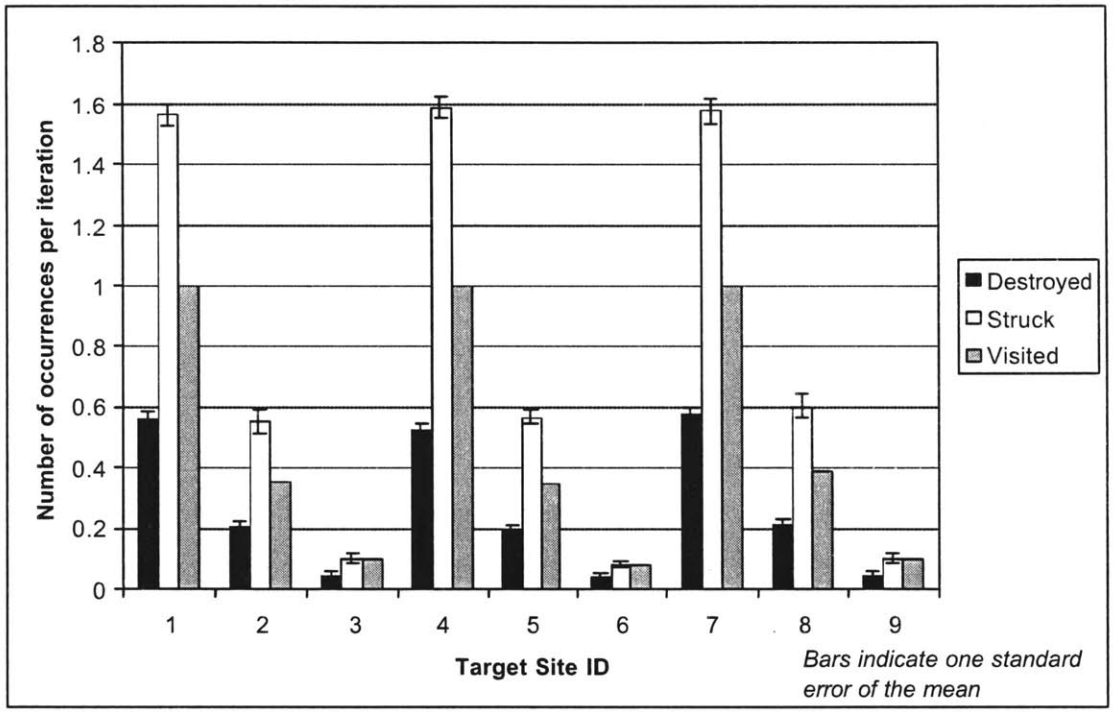

Figure 4-12 Scenario 2: Mission statistics for receding horizon strategy over a Monte-Carlo simulation (n=500)

UAV attrition severely impacts the mission performance of both the SPUDD and RH planners. The SPUDD planner destroys more targets than the RH controller ($p<0.05$), however, no statistical difference exists between the numbers of UAVs lost by the two methods. The SPUDD policy's bias towards destroying more targets at the expense of vehicle attrition exposes partiality in the planner's optimization and reward functions.

SPUDD's contingency-based policy inherently rejects actions with high loss risk and low success probability to extract the greatest cumulative reward over the mission duration. Indeed, SPUDD's reward function $\sum_{i=1}^{M} T_i W_i$, where T_i is indicates whether target i is destroyed and W_i is its value, represents the cumulative reward obtained from the M targets. As noted in Section 2.3, the optimization functions of neither SPUDD nor RH explicitly represent the cost of vehicle losses because such additions substantially increase the computational requirements of the SPUDD planner and limit the sizes of missions that can be reasonably examined.

Although SPUDD and RH suffer similar UAV losses, the disparity in computational performance is great. RH spends an average total of 81±7 seconds planning, while SPUDD forms its policy in 8.73 hours. The SPUDD planner destroys more targets than the RH controller with a universal contingency-based policy, but takes 500 times longer to compute ($p<0.05$). Construction of SPUDD's policy consumes a maximum of 558 MB of memory, while iterating through 377,712 nodes. For a classical MDP approach, this search space is equivalent to 2,553,809,902 nodes.

Although SPUDD substantially compresses the search space, it still suffers from the curse of dimensionality described in Section 3.2.3.4. Essentially, one vehicle and one target have been added to Scenario 1. The corresponding additions to the action and reward ADDs explode SPUDD's policy construction time by nearly eight hours and double its maximum memory usage. In contrast, the execution time for the RH controller remains within two minutes. These results highlight the computational differences of global contingency planning and localized, sequential re-planning. The SPUDD planner's computational requirements weigh heavily on its performance advantages. A considerable amount of pre-mission time is required to realize the performance benefits of SPUDD.

4.2.2.2 SPUDD and RH performance gain with fuel

In the preceding test scenarios, UAVs were initialized with a predetermined quantity of fuel. This amount was maintained constant across the preceding trials to fairly compare

the planners. Fuel though is a limiting constraint that directly affects mission performance. To examine the influence of fuel in Scenario 2, the numbers of targets destroyed and vehicles lost are measured for varying quantities of initial UAV fuel. For each fuel level, the planners are evaluated in a five hundred sample Monte-Carlo simulation. The difference in performance metrics for the two planners is shown in Figure 4-13.

For scenarios in which each UAV is initialized with fewer than three fuel units, SPUDD's contingency-based policy exhibits comparable behavior to the RH controller's sequential approximations. Greater fuel quantities, however, allow the UAVs to advance farther into the mission and intensify the two planners' performance differences. As the mission extends, more targets are destroyed and more UAVs are lost. Generating contingences based on its capabilities and risks model, the SPUDD planner destroys more targets and loses fewer UAVs than the RH controller. For UAVs initialized with over nine fuel units, the mission performances of the SPUDD and RH planners appear bounded. These bounds reflect other constraints in the system, including the number of targets and UAVs, munitions, *max_strikes*, strike success capabilities, and attrition rates. In this performance-bounded region, the SPUDD planner destroys about 0.4 more targets and loses 0.2 less UAVs than the RH controller, per iteration. Variance in the mission performance differences of these planners produces uncertainty in the statistical significance of these results. One-tailed, unpaired Student t-Tests suggest that the SPUDD controller destroys more targets for fuel settings above seven increments and loses fewer vehicles above six increments ($p<0.05$).

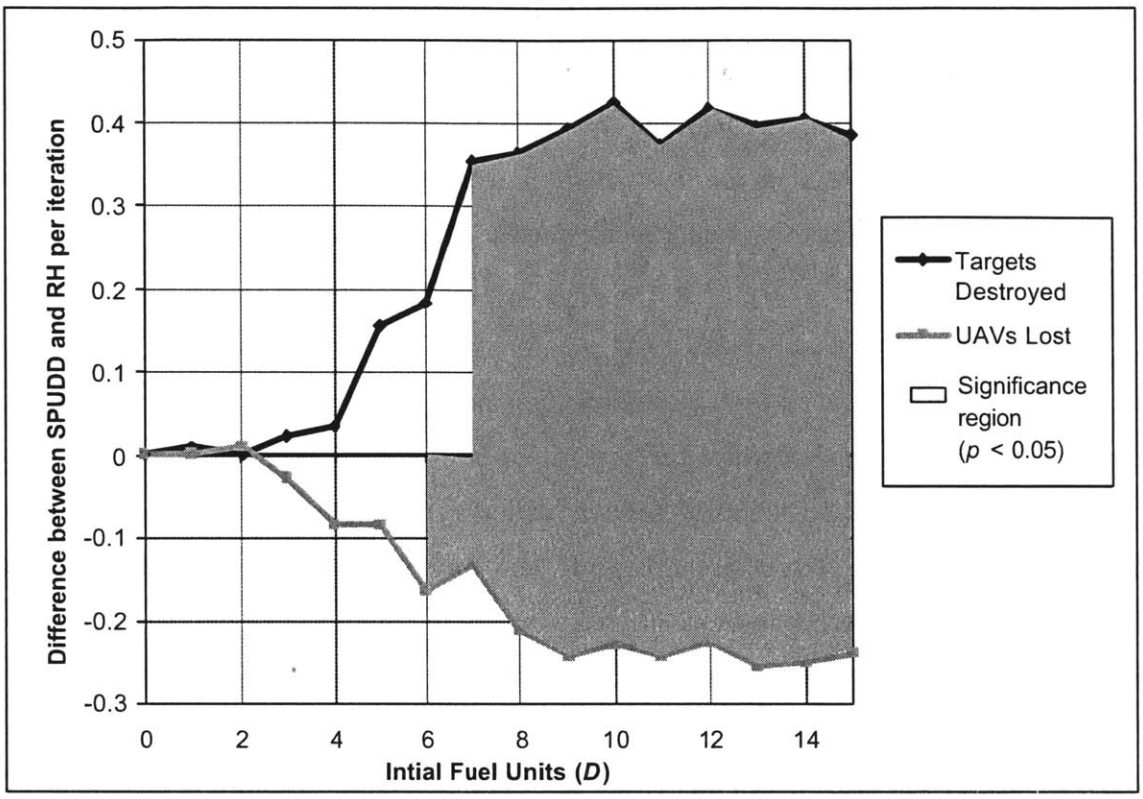

Figure 4-13 Scenario 2: Mission performance differences of SPUDD and RH planners as a function of UAV fuel over Monte-Carlo simulations (n=500). Highlighted significance region computed by a series of one-tailed, unpaired Student t-Tests (p<0.05)

4.2.3 Scenario 3

Scenarios 1 and 2 revealed some of the UAV behavioral differences produced by the SPUDD and RH planners. These scenarios, however, only focused on strike sequences as a mission performance discriminator (i.e. both planners chose identical vehicle-to-target assignments). Scenario 3 extends the investigation to measure the controllers' performance attributes for dissimilar vehicle-to-target assignments *and* strike sequences.

The battlefield is initialized with three UAVs and eight targets, as shown in Figure 4-14. Like Scenarios 1 and 2, *max_strikes* limits the number of strike attempts on a target to three, and the UAVs' strike success probability degrades from 0.5 to 0.2 to 0.1 with each strike on a particular site. Additionally, UAVs may be lost 10 percent of the time during transits between sites and 25 percent of the time during strike executions.

The scenario includes seven targets y_1, y_2, y_3, y_4, y_5, y_6, y_8, which offer a reward of three, and one highly-prized target y_7, which has a value of twelve. Perhaps, target y_7 is a key weapons depot and the others are less-critical, communication infrastructure targets.

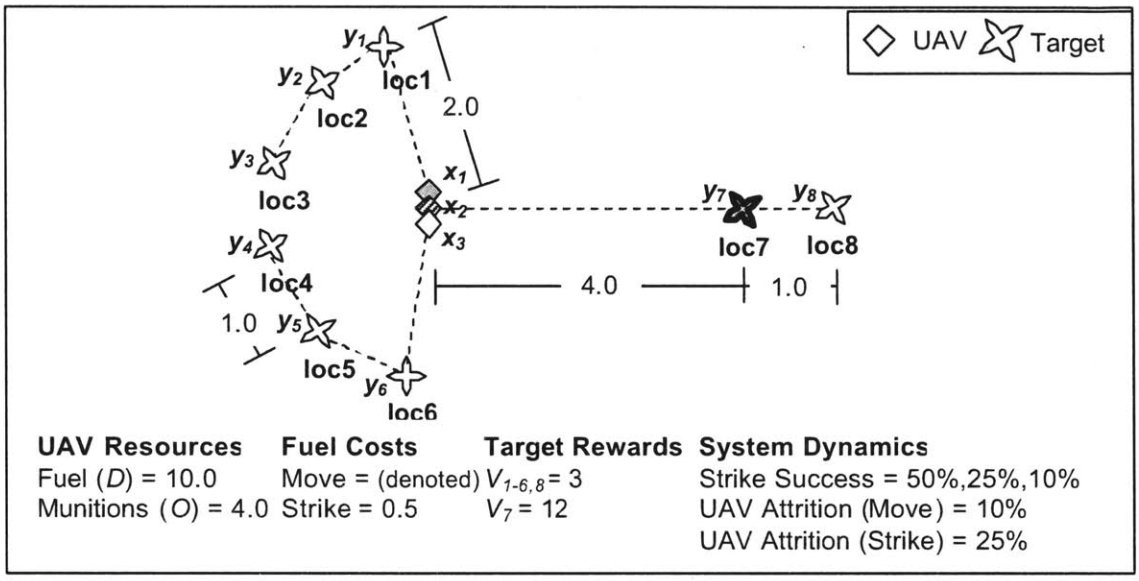

Figure 4-14 Scenario 3: Initial system with three UAVs and eight targets. Dotted lines denote the nominal target assignment sequence for the SPUDD controller

4.2.3.1 SPUDD and RH comparison

Figure 4-14 depicts the nominal vehicle-to-target assignment determined by the SPUDD policy. In average conditions, x_1 is assigned to targets y_1, y_2, y_3; x_2 to y_7, y_8; and x_3 to y_4, y_5, y_6. Since the SPUDD planner performs a global search for an optimal policy, UAV x_2 is immediately dispatched to strike high-value target y_7. Even though UAV x_2 is in closer proximity to other targets, the reward at target y_7 dominates SPUDD's expected discounted reward computation. SPUDD's policy selects contingent actions that maximize this expected reward in the presence of uncertainty. As a result, either UAV x_1 or x_3 is rerouted to target y_7 if x_2 is lost. Indeed, SPUDD's policy directs UAVs x_1 and x_3 to attempt only one strike on each of their targets until y_7 is destroyed, to maintain vehicles for backup. After target y_7 is eliminated, the UAVs follow a course of action that mirrors that observed in Scenario 2. Success probability, risk aversion, and fuel

constraints contribute to SPUDD's conservative strategy for striking lower-valued targets. Unless a target's reward is exceptionally high, the planner tends not to assign multiple UAVs to a single target because the *max_strikes* constraint permits each target site to be struck a maximum of three times.

Figure 4-15 depicts the number of times each target was visited, struck, and destroyed with the SPUDD planner. Highly-prized target y_7 is destroyed most often at 0.47±0.02 times per iteration. UAV attrition, during move and strike operations, substantially reduces the number strikes performed on targets later in mission. For a five hundred sample Monte-Carlo simulation, an average of 2.07±0.03 UAVs are lost and 2.70±0.07 targets are destroyed, per iteration. The UAVs collect an average reward of 12.33±0.45 per iteration.

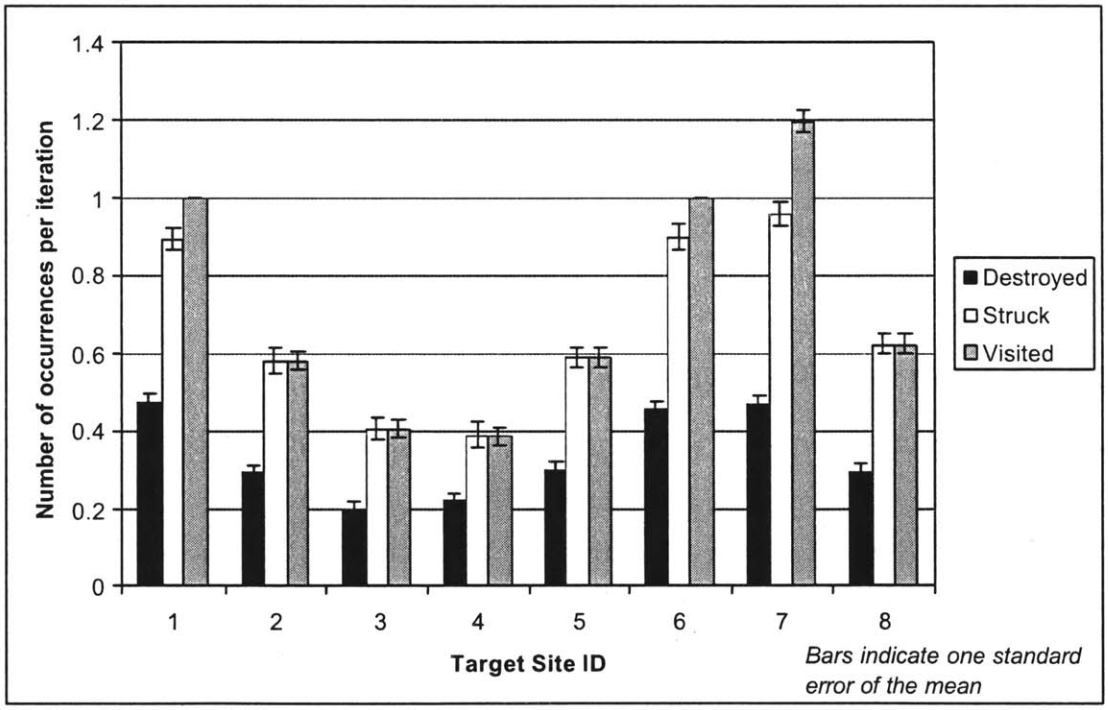

Figure 4-15 Scenario 3: Mission statistics for SPUDD policy over a Monte-Carlo simulation (n=500)

Unlike the SPUDD planner, the RH controller cannot build contingencies for stochastic events with its incomplete model of the system. RH's model neither represents the

UAVs' strike capability nor attrition probability. Additionally, the controller's visibility is limited by its localized planning horizon H_k. As discussed in Section 3.3.2.3, Cassandras and Li showed that their RH controller requires H_k to be based on the shortest distance between any vehicle and any target at time t_k to guarantee stationary trajectories. Figure 4-16 indicates that high-value target y_7 is excluded from the controller's initial planning horizon H_0 at time t_0. Nominally, the UAVs first destroy targets within this horizon ($y_1 \ldots y_6$) before attacking targets y_7 and y_8.

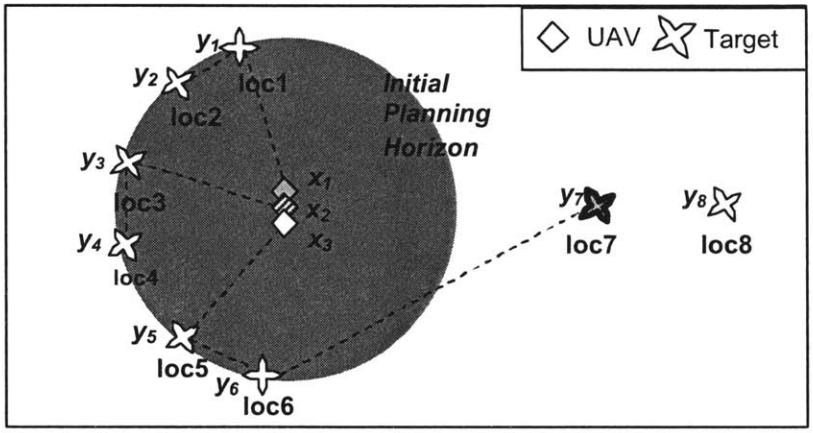

Figure 4-16 Scenario 3: Nominal vehicle-to-target assignment sequence for the RH controller

The RH planner's vehicle-to-target assignment and sequencing adheres to its greedy design. That is, the planner focuses on generating trajectories to targets within this horizon-limited area. Targets farther than H_k are largely ignored until the horizon expands. With a sizeable portion of targets in close proximity, the vehicles are attracted to a locally maximum region. Additionally, the controller's strike sequencing follows that of Scenarios 1 and 2. Specifically, the UAVs repeatedly strike target sites until successful or constrained by *max_strikes*. Figure 4-17 compares typical vehicle-to-target assignments and strike sequencing of both the SPUDD and RH controllers on a per UAV basis.

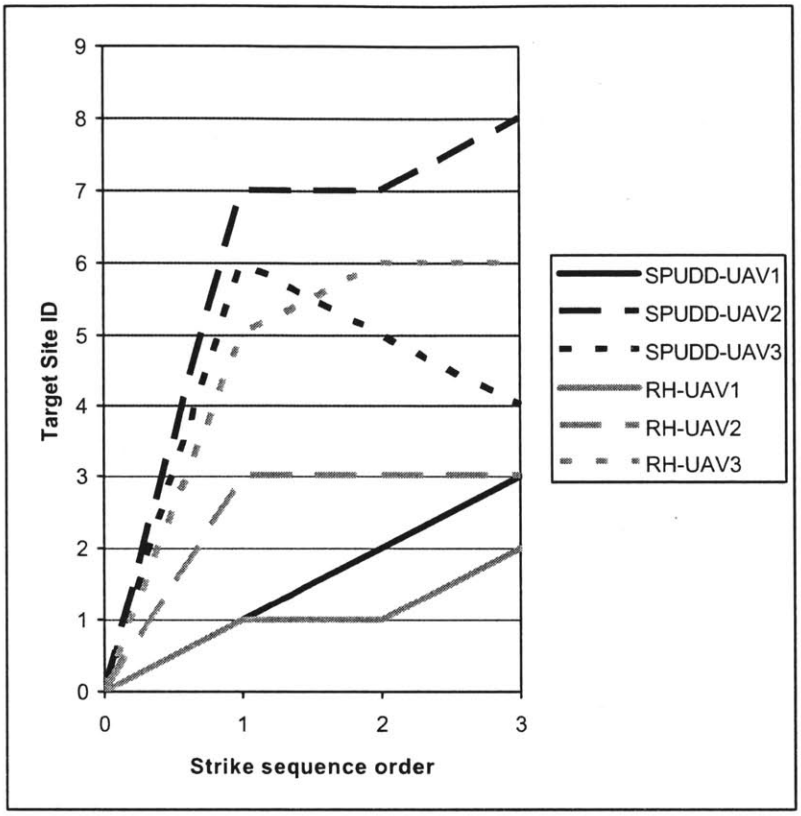

Figure 4-17 Scenario 3: Typical target sequencing for each UAV on a per planner basis

The RH planner's performance suffers from the lack of a complete system dynamics model and a myopic planning horizon. As shown in Figure 4-18, the UAVs concentrate strike attempts on nearby targets. Consequently, targets y_7 and y_8 are rarely visited because of fuel and/or munitions exhaustion and UAV attrition. For example, target y_1 is struck about 1.3 times, y_2 is struck 0.2 times, and y_7 is struck 0.1 times, per iteration. Target y_8 is visited 0.04 times per iteration, but never struck because all vehicles that moved to its site were subsequently lost. For a five hundred sample Monte-Carlo simulation, an average of 2.61±0.03 UAVs are lost and 2.14±0.03 targets are destroyed, per iteration. While SPUDD destroys more targets than it loses UAVs, RH loses more UAVs than it destroys targets (p<0.05). Furthermore, RH-directed UAVs destroy high-value target y_7 only 3 percent of the time, claiming a significantly lower reward of 6.69±0.33 per iteration.

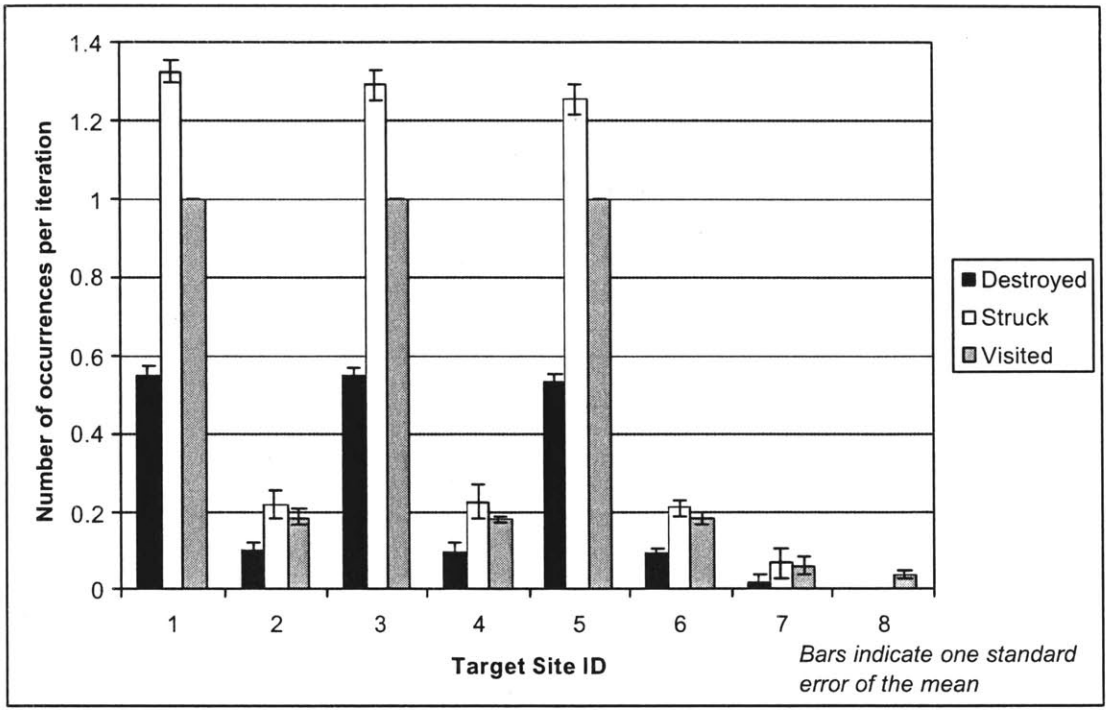

Figure 4-18 Scenario 3: Mission statistics for receding horizon strategy over a Monte-Carlo simulation (n=500)

4.2.3.2 SPUDD sensitivity to reward valuations

In part, the RH planner performs poorly because its local maximization strategy ignores the distant, but more valuable target, y_7. Prior to mission execution, suppose the SPUDD planner was unaware of target y_7's high-value. Perhaps, intelligence reports indicated that all eight targets were equally-important infrastructure targets. During post-mission analysis, military strategists may realize that a high-value, critical weapons depot existed at y_7.

This sensitivity study compares SPUDD's mission and computational performance in a scenario where its reward function is true to a distortion. The SPUDD planner must negotiate a reward valuation model in which all eight targets have a reward valuation of three, even though target y_7 will later be known to have a value of twelve. The mission performance of the SPUDD planner with such distorted reward valuations is shown in Figure 4-19. The reward model engenders a policy that does not actively focus on target y_7. Vehicle-to-target assignments follow that of the greedy RH controller;

however, SPUDD's complete model of UAV strike capabilities and attrition rates support a cautionary, contingency-based policy. Indeed, strikes are only executed at high of confidence levels to mitigate the effects of attrition. Tempering the RH controller's sharp performance decline over multiple targets, the UAVs strike y_1, y_2, y_3 about 0.9 times; y_4, y_5, y_6 0.6 times; y_7 0.4 times; and y_8 0.3 times, per iteration. With a distorted reward model, SPUDD's policy loses 2.20±0.03 UAVs and destroys 2.72±0.07 targets, per iteration. While the imprecise model values the accumulated rewards as 8.16±0.21 per iteration, the UAVs actually collect a reward of 10.37±0.45 per iteration by inclusion of target y_7's high-value. In comparison to its performance with a true reward model, SPUDD destroys a comparable number of targets, but accumulates less reward by not aggressively pursuing critical target y_7.

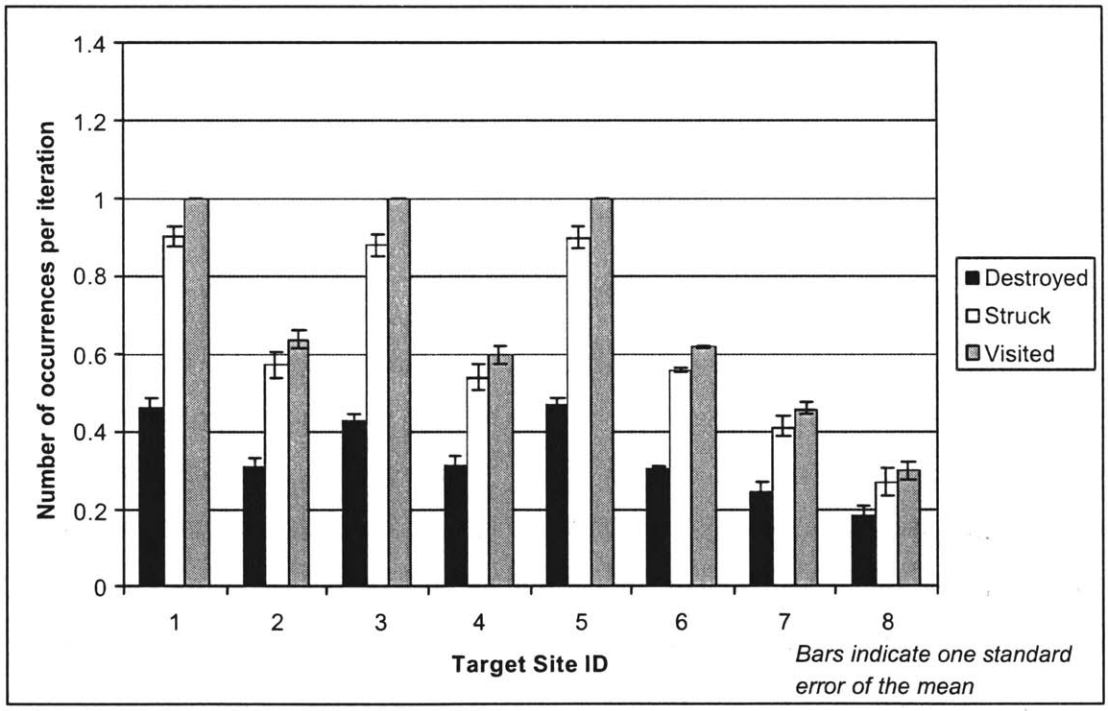

Figure 4-19 Scenario 3: Mission statistics for SPUDD policy with distorted reward function over a Monte-Carlo simulation (n=500)

Still, the SPUDD planner destroys more targets than the RH controller in both the fully-observable and sensitivity trial missions. The performance metrics for each test case are summarized in Figure 4-20.

	UAVs Lost per Iteration	Targets Destroyed per Iteration	Target 7 Destroyed per Iteration	Reward per Iteration
SPUDD	2.07±0.03	2.70±0.07	0.47±0.02	12.33±0.45
RH	2.61±0.03	2.14±0.03	0.03±0.02	6.69±0.33
SPUDD (distorted reward model)	2.20±0.03	2.72±0.07	0.25±0.02	10.37±0.45

Figure 4-20 Scenario 3: Mission performance summary of test cases

The planning time of the RH controller averages 72.42±3.21 seconds per iteration. Although plans are quickly generated, the RH approach loses more UAVs than it destroys targets. The SPUDD planner performs better than the RH controller; however, sufficient time and memory must exist for pre-mission planning. SPUDD's policy is generated in 6.6 hours, consuming 526 MB of memory for 351,170 nodes.

4.2.5 Scalability

Mission performance and computational cost seem to follow each other. Complete and accurate system models tend to provide high mission success; however, the preceding case studies have also shown they beget increasing computational costs.

These computational requirements are inherently intertwined with the complexity of the planning task. Section 3.4 described the optimization and representation differences of the SPUDD and RH controllers. Whereas the RH controller optimizes over a sequence of local horizons, the SPUDD planner searches for an optimal policy with contingencies for a global state space. The RH controller models utility functions and state variables, while the SPUDD planner represents the complete system, including state transition dynamics.

This section evaluates the ability of the planners to scale to systems of greater dimensionality. Mission and computational performance is examined over a series of scenarios, which vary system attributes modeled by both controllers. Specifically,

scalability is revealed for missions that include increasing numbers of UAVs and targets. The standardized test scenarios include up to three UAVs and nine targets at positions which avoid the special-case behavior observed in Scenario 3. As shown in Figure 4-21, UAVs begin their mission from a common, initial position. Targets are positioned along evenly-spaced rows and columns, where the number of columns equals the number of UAVs.

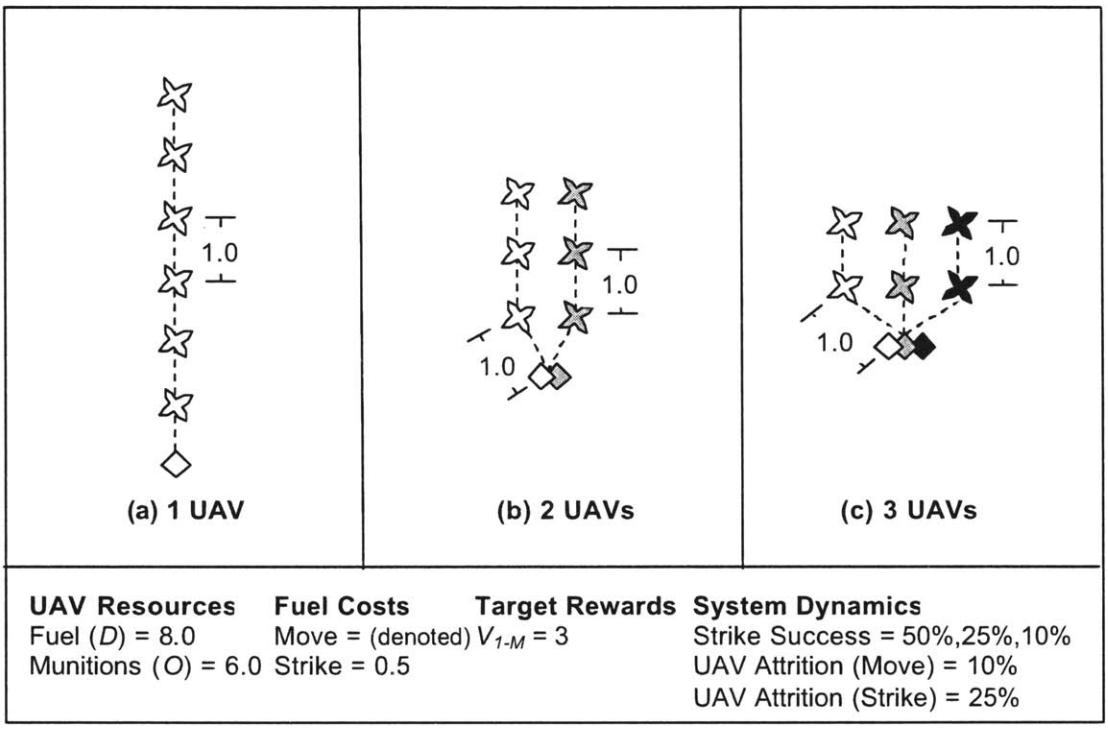

Figure 4-21 Standardized scenario examples with six targets and varying numbers of UAVs. Dotted lines denote the nominal target assignment sequence for both SPUDD and RH controllers

In each of the 27 standardized scenarios, the UAVs are initialized with the same fuel and munitions. Targets are valued equally, and the cost for transiting between target sites is proportional to distance. Strike attempts on a single target site are limited to three by *max_strikes*, and strike success probability degrades from 0.5 to 0.2 to 0.1 with each attempt to destroy a particular target. UAVs are lost 10 percent of the time during transits between sites and 25 percent of the time during strike executions. Five hundred

sample Monte-Carlo simulations of each scenario provide the average-case behavior of the planners. Pre-mission policy construction establishes the SPUDD algorithm's planning time, and the sum of the RH controller's on-line optimizations determines its time.

The planning times of the SPUDD and RH controllers in each of these scenarios are shown in Figure 4-22. SPUDD generates policies about an order faster than the RH controller for scenarios that include less than a total of 4 UAVs and targets. For greater numbers of UAVs and targets, the curse of dimensionality that plagues SPUDD worsens its computational performance. Each additional target site lengthens SPUDD's policy construction time by an average of 1,532 seconds. Oppositely, the RH controller's planning time increases at a more restrained rate of about 6 seconds per target. The RH controller avoids some of the complexity of larger state and action spaces by disregarding certain system attributes and sequentially optimizing over localized spaces. With each additional UAV, SPUDD's planning time increased by an average of 4,200 seconds, while the RH controller gained an average of 12 seconds. The accelerated growth of SPUDD's execution time eases moderately in scenarios with seven or more targets. The difference of the two planners' execution times is most pronounced in these relatively large scenarios. In the three-UAV, nine-target scenario, SPUDD spends nearly 400 times longer to compute its policy than the summed execution time of RH's optimizations. These planning time differences represent significant disadvantages in practicality. Unless sufficient pre-mission time exists for SPUDD policy generation, the possible mission benefits of contingency planning cannot be realized.

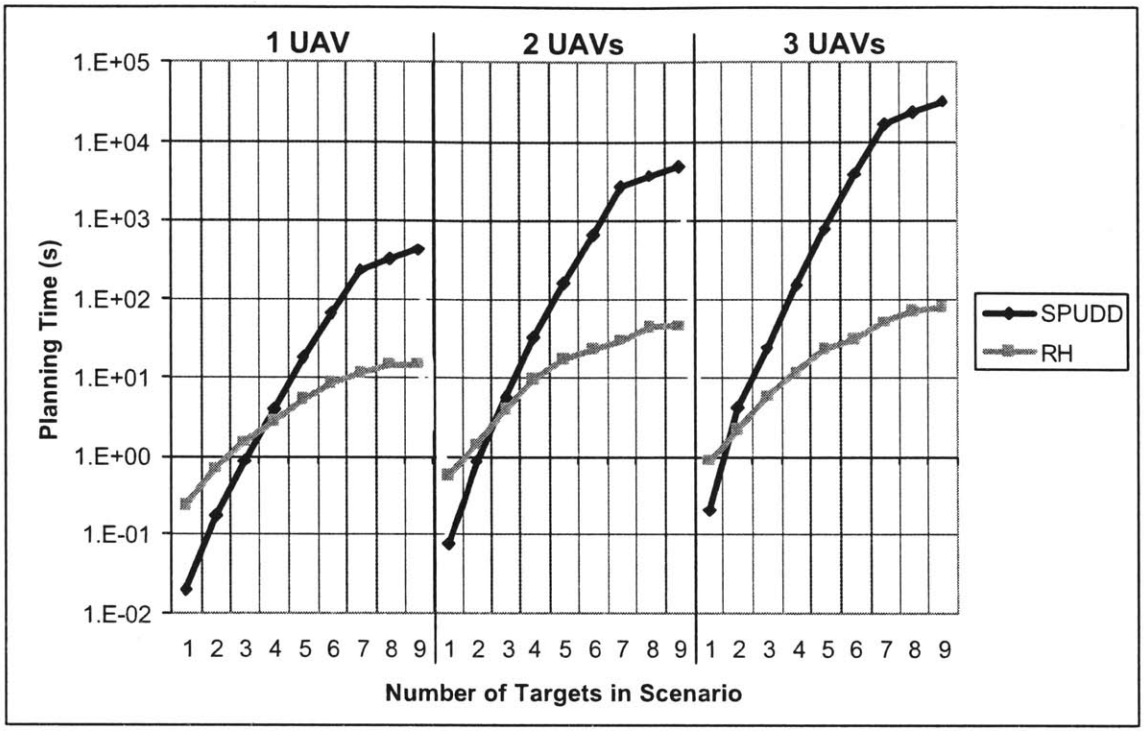

Figure 4-22 Planning times of SPUDD and RH controllers for standardized scenarios

SPUDD's optimal policy generation consumes both time and memory. As shown in Figure 4-23, SPUDD's memory requirements depend on the number of UAVs and targets included in the mission. The plot shows limited growth in maximum memory utilization for scenarios that include less than a total of seven UAVs and targets. Memory consumption accelerates in larger scenarios; however, like the trend of SPUDD's execution time, its rate diminishes in scenarios that include seven or more targets. In the three-UAV scenarios, maximum memory usages range from 12 MB with one target to 560 MB with nine targets.

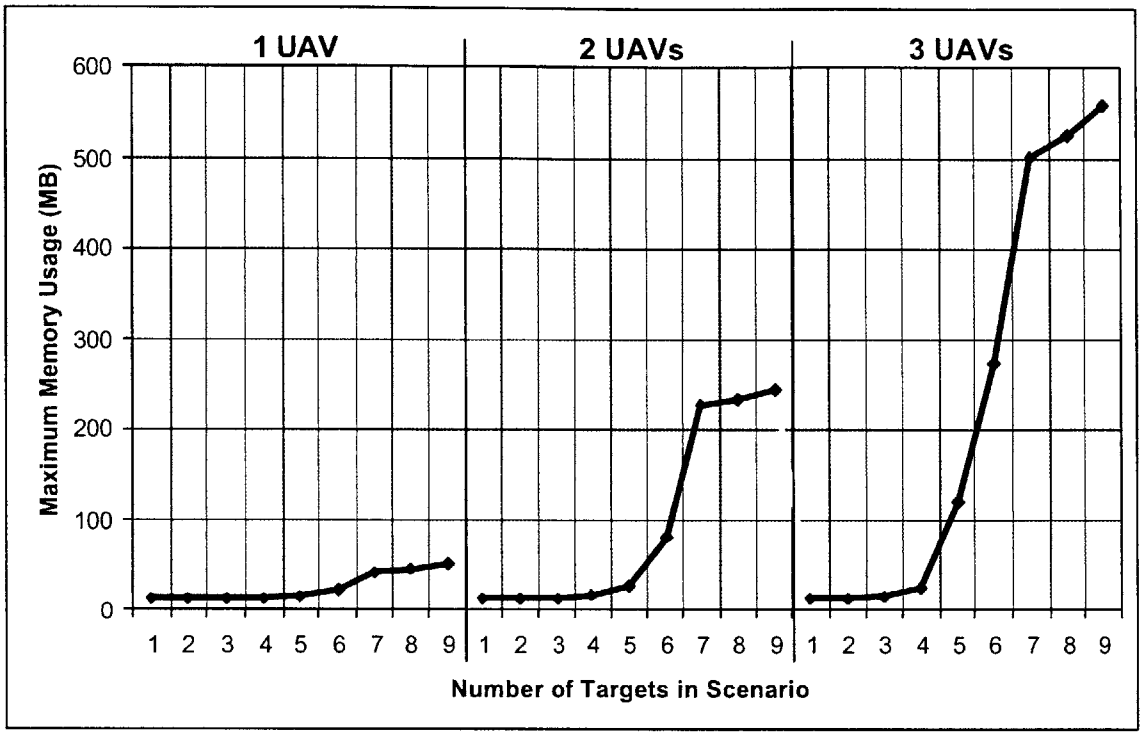

Figure 4-23 Maximum memory usage of SPUDD and RH controllers for standardized scenarios

The SPUDD planner's maximum memory consumption is determined by the size of its model-dependent search space. As described in Section 3.2.4, SPUDD uses compact ADD action and reward representations that avoid tabulations. Aggregating multiple subtrees of the same value, the ADD formulation compresses the search space. The numbers of action and value nodes represented by ADDs and an analogous tabular-based MDP structure are shown in Figure 4-24. The orders of difference between the two representations intensify for increasing dimensionality. Like the trends of SPUDD's planning time and memory usage, the growth in ADD nodes decelerates in scenarios that include more than seven targets. The equivalent tree's expansion does not decrease as substantially. In scenarios with three UAVs, the numbers of ADD and equivalent tree nodes ranged from 104 to 377,712 and from 476 to 2,553,809,902, respectively. On average, the number of ADD nodes increased by 24,941 with each additional target, while the equivalent tree grew at a rate of 111,777,704 nodes per target.

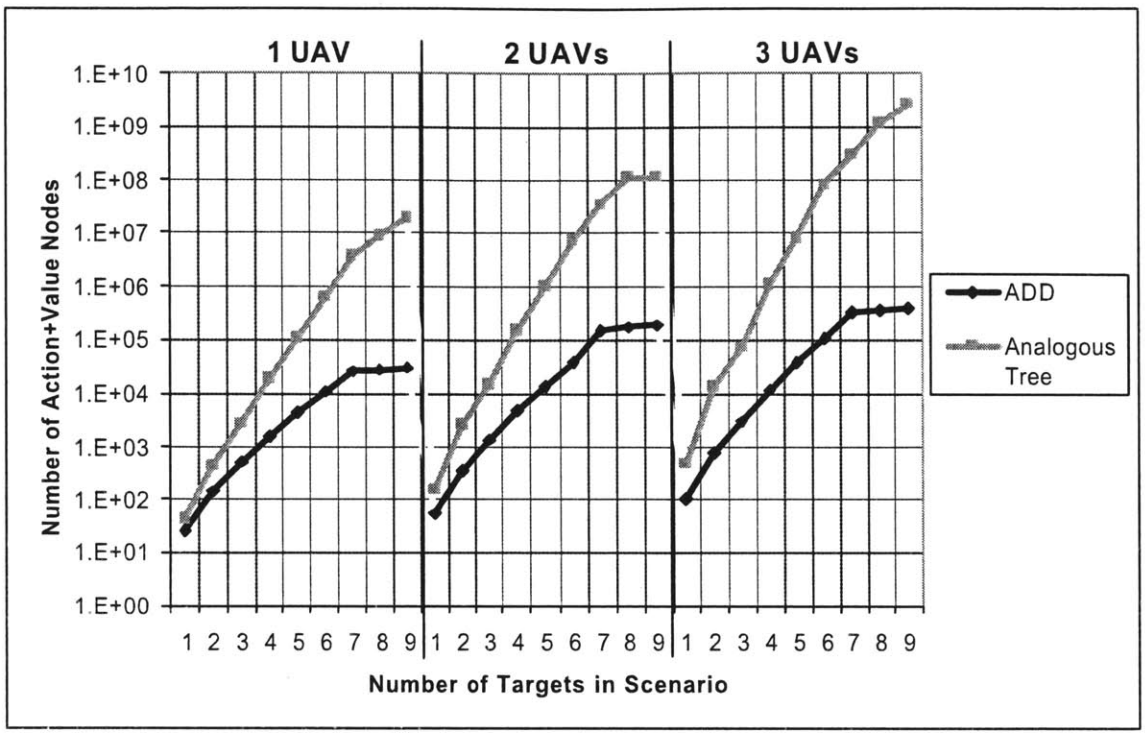

Figure 4-24 Representation sizes of SPUDD's ADD and an analogous tabular-based MDP structure

Although SPUDD shrinks the search space with efficient ADDs, increases in dimensionality curse the method's computational feasibility. Each vehicle and target adds considerable model complexity. Elaborate system models result in large memory consumptions and long planning times. While hardware costs and sizes miniaturize on an annual basis, the value of time continues to increase – especially in military operations. The RH controller demonstrates a more graceful scaling of computational requirements with increasing numbers of UAVs and targets. On the other hand, the RH approach does not guarantee an optimal course of action. The RH controller exhibits better computational scalability precisely because it sequentially negotiates localized approximations of the system.

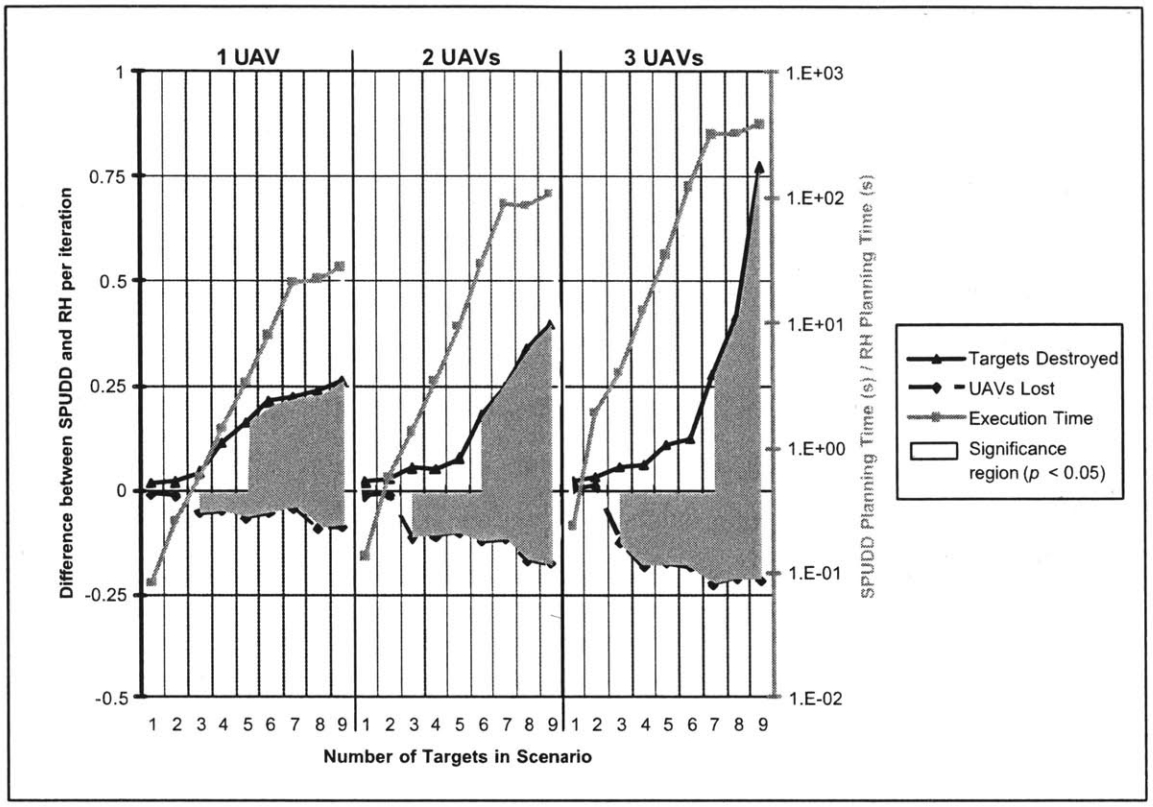

Figure 4-25 Mission and computational performance differences of SPUDD and RH planners in standardized scenarios over multiple Monte-Carlo simulations (n=500). Highlighted significance region computed by a series of one-tailed, unpaired Student t-Tests (p<0.05)

Figure 4-25 compares the behavioral and computational trade-offs exhibited by the two planners. Each planner was examined in five hundred sample Monte-Carlo simulations of the 27 standardized scenarios. This two y-axes plot shows the explosive increase in execution time of the SPUDD planner in relation to the RH controller. SPUDD's policy generation time tends to be shorter than the RH controller for scenarios with less than a total of 4 UAVs and targets, and decelerates in growth for scenarios that include more than seven targets. Figure 4-25 highlights scenarios in which SPUDD showed better mission performance based on the results of one-tailed, unpaired Student t-Tests (p<0.05).

SPUDD loses fewer UAVs in scenarios that include more than three targets, though neither planner explicitly represents loss-of-vehicle costs (p<0.05). Unlike the

RH controller, however, SPUDD's model does represent UAV attrition probabilities. To extract the highest cumulative reward, SPUDD's contingency-based policy cautiously avoids actions that are associated with expectations of high loss and low gain. Oppositely, the RH controller directs each UAV to repeatedly strike its targets until successful or constrained by *max_strikes*. Because UAVs attrite 25 percent of the time during strike operations, the RH controller's greedy approach loses more UAVs ($p<0.05$).

As denoted in Figure 4-21, the battlefield's grid-like design induces nominal vehicle assignments along target columns. In small scenarios that include a comparable number of targets and UAVs, each UAV ultimately visits few target sites. Consequently, the two planners produce similar courses of action and exhibit comparable mission performances because the UAVs have sufficient supplies of fuel and munitions to fully strike every target to its *max_strikes* limit. In larger scenarios where the targets outnumber the UAVs, SPUDD's conservative strike sequencing limits UAV losses and destroys more targets than the RH controller ($p<0.05$). This advantage is apparent in the one-UAV, five-target; two-UAV, six-target; three-UAV, seven-target battlefields and larger.

Furthermore, the one-UAV mission performance trends indicate a bound for both the SPUDD and RH planners in scenarios that have low vehicle-to-target ratios (e.g. scenarios that include one-UAV and six targets or more). These bounds reflect constraints in the system, including fuel (D), munitions (O), *max_strikes*, strike success capabilities, and attrition rates.

Across the 27 test scenarios, SPUDD's planning time is an average factor of 60 times greater than the RH controller. For each target added to the standardized scenarios, on average the SPUDD planner has a 22 times longer execution time, destroys 0.06 more targets, and loses 0.02 fewer UAVs than the RH controller. Similarly, for each additional UAV, on average SPUDD takes 62 times longer, destroys 0.03 more targets, and loses 0.05 fewer UAVs than RH. The considerable increase of SPUDD's policy generation time may be prohibitive; however, its gain in mission performance may be worthwhile. The partially decelerated increase of computational resources in scenarios involving more than seven targets might offer some relief for large-scale mission planning. Still, sufficient pre-mission time must exist for SPUDD's optimal policy construction.

5 Conclusions

"I will never believe that god plays dice with the universe"

– Albert Einstein

This thesis examines two planning algorithms that coordinate collaborative agents to accomplish shared mission objectives in the presence of action outcome uncertainty. On a fully-observable battlefield, the planners coordinate a team of unmanned aerial vehicles (UAVs) to obtain a maximum reward by destroying targets (Chapter 2). Stochastic components, including UAV capability and attrition, represent uncertainty in the simulated missions.

5.1 Summary

Planning algorithms can be generally classified into two categories: exact and heuristic. In this thesis, an exact stochastic planning using decision diagrams planner (SPUDD) and a heuristic, receding horizon controller (RH) are evaluated in typical planning problems. SPUDD searches for an optimal policy with global contingencies, while RH sequentially selects approximate plans over more localized horizons.

Generally, the two planners trade mission and computational performance. Although the results are limited to specific problem instances, they provide characterizations of the algorithms' capabilities and limitations. The SPUDD planner provides optimal courses of action for all possible conditions over the mission duration; however, the algorithm consumes substantial computational resources. On the other hand, the RH approach does not guarantee optimality, but may form worthy plans without evaluating every contingency.

5.2 Capabilities and Limitations

The SPUDD and RH planners have an expansive set of customizable attributes. The accuracy, observability, and interrelationships of these parameters, such as discount factors, action costs, rewards valuations, success capabilities, and attrition probabilities, directly impact both mission and computational performances (Chapter 3). Indeed, the SPUDD planner's more complete and truthful model of the system exhibits statistically better mission performance at substantially higher computational cost in comparison to the heuristic, RH controller (Chapter 4).

Sensitivity trials capture each algorithm's robustness to real world planning environments where planners must negotiate incomplete or inaccurate system models. A model's completeness and correctness tends to correlate to a planner's computational complexity and mission performance. Indeed, the SPUDD and RH planners endure high UAV attrition because neither explicitly represents the cost of vehicle losses. The mission performances of both methods decay as the quality of their system model worsens.

Scalability studies assess the trends of the mission and computational performance metrics for both approaches in larger scenarios that include increasing numbers of UAVs and targets. For relatively small scenarios, the RH controller provides statistically similar mission performance to the SPUDD planner in considerably less time. In scenarios with greater dimensionality, however, SPUDD's mission advantages and computational weaknesses intensify in relation to the RH controller. For long missions where the UAVs are initialized with large quantities of fuel (D) or that include many

more targets than vehicles, SPUDD destroys more targets and loses fewer UAVs than RH (p<0.05). In certain cases, the mission performances of both planners is bounded by other constraints in the system, including munitions (O), maximum strike attempts (*max_strikes*), strike success capabilities, and attrition rates.

Although SPUDD reduces the computational requirements of tabular MDP value iteration with efficient algebraic decision diagram (ADD) representations, its state, action, and outcome spaces are afflicted by the curse of dimensionality in large scenarios. Oppositely, the RH controller's abridged system model and myopic horizon truncate the planning problem. The RH method suffers mission performance shortcomings because its model does not represent the uncertainties associated with the UAVs' capabilities and risks. Instead of formulating contingencies for possible states over the mission duration, RH sequentially re-plans over shorter horizons that steer the UAVs towards local maxima. Consequently, the RH controller is characterized by shorter planning times and poorer mission performance than the SPUDD planner.

In nearly every scenario, statistical tests show that SPUDD's policy destroys more targets and loses fewer UAVs than the RH controller (p<0.05). The merits of this statistical significance depend on one's perspective. SPUDD's mission advantages can be realized only with sufficient availability of pre-mission planning time. While hardware costs and sizes miniaturize on an annual basis, the value of time continues to increase – especially in military operations. These concerns are particularly relevant for responding to unexpected events. Both SPUDD and RH are designed for fully-observable systems; however, the real world is difficult to completely estimate and quantify. The sensitivity studies show that the planners' performances deteriorate with partially-observable and imprecise systems models. Consequently, re-planning may be necessary. Whereas re-planning is an integral feature of the RH controller, the computational requirements of reconstructing SPUDD's optimal policy is prohibitive. For example, if targets need to be destroyed in a particular order, the RH controller simply ignores sites that need to be attacked later in the mission. Conversely, SPUDD incorporates this ordering into its model, but cannot immediately respond to unanticipated changes during mission execution.

5.3 Future directions

This thesis formally measures the mission and computational performance trade-offs of the SPUDD and RH planners over a spectrum of scenarios. This objective suggests three opportunities for future work: (1) performance metrics, (2) planning models, and (3) scenarios.

5.3.1 Performance metrics

5.3.1.1 Mission measures

Mission performance is assessed based on vehicle-to-target assignments and sequencing, accumulated rewards, number of targets destroyed, and number of UAVs lost. These metrics provide a high-level glimpse into the UAV behavior induced by each planner. Vehicle-to-target assignments and sequencing give indications to the cooperativeness of a planner's objective fulfillment strategy; however, these metrics do not describe the estimated time of mission completion. Simulated mission times are a better characterization of speed.

5.3.1.2 Computation measures

Pre-mission policy construction establishes the SPUDD algorithm's planning time, and the sum of the RH controller's on-line optimizations determines its time. These measurement inconsistencies may imperfectly represent each method's true computational performance. Total planning time and maximum memory consumption are practical measures, but these can be supplemented by theoretical average- and worse-case computability analysis.

5.3.2 Planning models

5.3.2.1 Scope

The planning models incorporate basic system attributes, including action costs, reward valuations, health, fuel, munitions, positions, strike capabilities, and attrition probabilities. Actual UAV missions necessitate consideration of many other features, including heterogeneous vehicle characteristics, three-dimensional space, non-stationary targets with unique threat capabilities, time-sensitive rewards, attrition costs, etc. Such additions curse the feasibility of the SPUDD planner. SPUDD substantially compresses the evaluation state space with its aggregative ADD representations; however, context-specific planners that exploit the structure of a problem domain may exhibit better scalability to missions of greater complexity. For instance, fuel and munitions can be modeled as part of an action's cost, rather than as separate state variables. In addition, a single UAV might be modeled to represent an entire team of UAVs that follow identical plans. Simplified system models could allow SPUDD to evaluate larger scenarios, but mission performance may suffer. Oppositely, incorporating additional system components, such as UAV capabilities and threats, into the RH controller's optimization may improve mission performance at the expense of greater computation.

5.3.2.2 Robustness

Both planners perform significantly worse with system models that are partially-observable, incomplete, or inaccurate. Although neither SPUDD nor RH is intended to manage such situations, actual missions require robustness to the unexpected. To ensure stationary trajectories, the RH controller determines the length of each optimization's planning horizon based on the shortest distance of UAV-to-target pairs. These myopic horizons disadvantage the controller in scenarios that include local maxima. Other schemes for setting the controller's horizon might still enforce stationary trajectories and provide better performance.

Whereas RH trades mission performance for computational efficiency, SPUDD performs optimally at the expense of substantial computational requirements. SPUDD

manages uncertainty with a complete and accurate system model; however, policy reconstruction in response to unanticipated events requires considerable processing time. Researchers at the University of British Columbia recently upgraded their implementation of SPUDD with approximate value iteration to speed computation and partially-observable Markov decision process (POMDP) capabilities to support greater robustness.

Alternative planning approaches include decentralized MDP models and policy roll-out. Decentralized MDPs (Dec-MDPs) methods capture the decentralized nature of multi-agent systems by using each agent's partial knowledge of the overall system to choose local actions (including communication and information sharing actions) [18]. Each agent formulates a local policy for itself, which aims to maximize the expected utility for the entire team. Research is focused on finding approximate solutions to Dec-MDPs, as optimal solutions were proven to require double exponential time (2^{2^n}) [4]. Policy roll-out, which approximates MDP policy iteration, combines machine learning and simulation to iteratively improve the best policy found until improvement ceases or a time limit is exceeded [39]. This process is initially computationally intensive, like SPUDD, but its reactive policy can quickly solve future problem instances of a specific domain.

5.3.3 Scenarios

Monte-Carlo simulations provide each planner's average-case performance in the test scenarios. The robustness of the SPUDD and RH planners is examined by sensitivity and scalability studies. The curse of dimensionality that plagues SPUDD limits the feasibility of considering missions that include more than 3 UAVs and 9 targets. Alternative planning models may permit comparison of the two approaches over a greater range of scenarios.

For each mission, system and optimization parameters, including discount factor, fuel (D), munitions (O), *max_strikes*, reward valuations, action costs, strike capabilities, attrition probabilities, and vehicle and target positions, can also be varied to experimentally observe their effect on performance. For example, the number of strikes

attempts allowable on a single target (*max_strikes*) is three in the examined scenarios. As a result, unless a target's reward is exceptionally high, planners tend not to assign multiple UAVs to a single target. Generalizing the interdependency of scenario attributes could assist planners to appropriately select model parameters.

[*Except for this sentence, this page intentionally left blank*]

References

[1] Basharin, G., Langville, A., and Naumov, V., "The Life and Work of A. A. Markov," *Linear Algebra and its Applications*, Vol. 386, Addison-Wesley, New York, 2004, pp. 3-26.

[2] Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, 1957.

[3] Bellingham, J., Tillerson, M., Alighanbari, M., and How, J., "Cooperative Path Planning for Multiple UAVs in Dynamic and Uncertain Environments," *Proceedings of 41st IEEE Conference on Decision and Control*, 2002, pp. 2816-2822.

[4] Bernstein, D., Givan, R., Immerman, N., and Zilberstein, S., "The Complexity of Decentralized Control of Markov Decision Processes," *Mathematics of Operations Research*, Vol. 27, No. 4, 2002, pp. 819-840.

[5] Bertsekas, D. and Castanon, D., "Adaptive aggregation for infinite horizon dynamic programming," *IEEE Transactions on Automatic Control*, Vol. 34, 1989, pp. 589–598.

[6] Boutilier, C., Dean, T., and Hanks, S., "Decision theoretic planning: Structural assumptions and computational leverage," *Journal of Artificial Intelligence Research*, Vol. 11, 1999, pp. 1-94.

[7] Boutilier, C., "Planning, Learning, and Coordination in Multiagent Decision Processes," Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK-96), 1996, pp. 195-210.

[8] Butenko, S., Murphey, R., and Pardalos, P., editors, *Recent developments in cooperative control and optimization*, Kluwer Academic Publishers, Boston, 2004.

[9] Cassandra, A., Kaelbling, L., and Littman, M., "Acting optimally in partially observable stochastic domains," *Proceedings of the 12th National Conference on Artificial Intelligence* (AAAI), 1994, pp. 1023-928.

[10] Cassandras, C. and Li, W., "A Receding Horizon Approach for Solving Some Cooperative Control Problems," *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002, pp. 3760-3765.

[11] Cruz, J., Simaan, M., Gacic, A., and Liu, Y., "Moving Horizon Game Theoretic Approaches for Control Strategies in a Military Operation," *Proceedings of the 40th IEEE Conference on Decision and Control*, 2001, pp. 628-633.

[12] Curry, M., Wohletz, J., Castañón, J., and Cassandras, C., "Modeling and Control of a Joint Air Operations Environment with Imperfect Information," *Proceedings of the SPIE 16th Annual International Symposium*, 2002, pp. 41-51.

[13] Dean, T. and Kanazawa, K., "A model for reasoning about persistence and causation," *Computer Intelligence*, Vol. 5, No. 3, 1989, pp. 142–150.

[14] Dearden, R. and Boutilier, C., "Abstraction and approximate decision theoretic planning," *Artificial Intelligence*, Vol. 89, 1997, pp. 219–283.

[15] Gamerman, D., *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inferences*, Chapman and Hall, New York, 1997.

[16] Garey, M., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[17] Gazi, V. and Passino, K., "Stability Analysis of Social Foraging Swarms: Combined Effects of Attractant/Repellent Profiles," *Proceedings of 41st IEEE Conference on Decision and Control*, 2002, pp. 2848-2853.

[18] Goldman, C. and Zilberstein, S., "Decentralized control of cooperative systems," Technical Report 03-36, Department of Computer Science, University of Massachusetts at Amherst, 2003.

[19] Guestrin, C., "Planning under uncertainty in complex environments," PhD thesis, Department of Computer Science, Stanford University, 2003.

[20] Guestrin, C., Koller, D., and Parr, R., "Multiagent planning with factored MDPs," *Proceeding of the 14th Neural Information Processing Systems (NIPS-14)*, 2001, pp. 1523-1430.

[21] Hanks, S. and McDermott, D., "Modeling a dynamic and uncertain world I: symbolic and probabilistic reasoning about change," *Artificial Intelligence*, Vol. 66, No. 1, 1994, pp. 1-55.

[22] Hoey, J., St-Aubin, R., Hu, A., and Boutilier, C., "SPUDD: Stochastic planning using decision diagrams," *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999, pp. 279-288.

[23] Howard, R., *Dynamic programming and Markov processes*, MIT Press, Cambridge, 1960.

[24] Hu, J. and Sastry, S., "Optimal Collision Avoidance and Formation Switching on Riemannian Manifolds," *Proceedings of 40th IEEE Conference on Decision and Control*, 2002, pp. 1071-1076.

[25] Juhasz, Z., "Multi-Agent Systems," COM6405: Agent Technology, University of Exeter, 8 March 2004, http://www.dcs.ex.ac.uk/studyRes/COM6405/2004_5.ppt [cited 23 December 2004].

[26] Lovejoy, W., "A survey of algorithmic methods for partially observable Markov decision processes," *Annals of Operations Research*, Vol. 28, No. 1, 1991, pp. 47-66.

[27] Li, W. and Cassandras, C., "Stability Properties of a Cooperative Receding Horizon Controller," *Proceedings of the 42nd IEEE Conference on Decision and Control*, 2003, pp. 492-497.

[28] Li, W. and Cassandras, C., "Stability Properties of a Receding Horizon Controller for Cooperating UAVs," Department of Manufacturing Engineering, Boston University, 2004.

[29] Lian, F. and Murray, R., "Real-Time Trajectory Generation for the Cooperative Path Planning of Multi-Vehicle Systems," *Proceedings of 41st IEEE Conference on Decision and Control*, 2002, pp. 3766-3769.

[30] Littman, M., "Algorithms for Sequential Decision Making," PhD thesis, Department Computer Science, Brown University, 1996.

[31] Manne, A., "Linear programming and sequential decisions," *Management Science*, Vol. 6, No. 3, 1960, pp. 259-267.

[32] Papadimitriou, C. and Tsitsiklis, J., "The complexity of Markov decision processes," *Mathematics of Operations Research*, Vol. 12, No. 3, 1987, pp. 441-450.

[33] Puterman, M., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York, 1994.

[34] Rabiner, L. and Juang, B., "An introduction to hidden Markov models," *IEEE ASSP Magazine*, January 1986, pp. 4-15.

[35] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, 1995.

[36] Singh, L. and Fuller, J., "Trajectory Generation for a UAV in Urban Terrain Using Nonlinear MPC," *Proceedings of 2001 American Control Conference*, 2001, pp. 2301-2308.

[37] Williams, R. and Baird, L., "Tight performance bounds on greedy policies based on imperfect value functions," *Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems*, Yale University, 1994.

[38] Wingate, D. and Seppi, K., "Cache performance of priority metrics for MDP solvers," Technical Report, Department of Computer Science Department, Brigham Young University, 2004.

[39] Yoon, S., Fern, A., and Givan, R., "Learning Reactive Policies for Probabilistic Planning Domains," Technical Report, Department of Electrical and Computer Engineering, Purdue University, 2004.