COOPERATIVE CONTROL METHODS FOR THE WEAPON TARGET ASSIGNMENT PROBLEM

A Dissertation
Presented to
The Academic Faculty

By

Kyle Volle

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Robotics

Georgia Institute of Technology

May 2018

COOPERATIVE CONTROL METHODS FOR THE WEAPON TARGET ASSIGNMENT PROBLEM

Approved by:

Dr. Jonathan Rogers, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical & Computer Engineering
Georgia Institute of Technology

Dr. Frank Hammond
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Eric Johnson
School of Aerospace Engineering
Pennsylvania State University

Dr. Kevin Brink
Senior Research Engineer
Air Force Research Laboratory

Date Approved: January 10, 2018

ACKNOWLEDGEMENTS

Thank you more than I can say to my parents, Richard and Carrie Volle, for their bottomless love and support. I couldn't have done anything without you.

My sister and best friend, Katie Volle, has always been there for me with a listening ear and usually better advice than I give her credit for. Thanks, Sis.

I am grateful to the Air Force for their funding of the research presented in this dissertation and in particular to Dr. Kevin Brink for his mentorship in the Air Force Research Lab Scholars program.

Thank you to Dr. Jonathan Rogers for your guidance and advice throughout the past few years.

I will always be grateful to family and friends, nearby and far away, for their support and many fond memories.

Last, but not least, I'm thankful for all of the people I've been able to meet and work with in the Intelligent Robotics and Emergent Automation Lab, you're all going to do great things.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

SUMMARY

Weapon target assignment (WTA) is a combinatorial optimization problem in which a set of weapons must selectively engage a set of targets in order to minimize the expected survival value of the targets. In its distributed form, it is also an important problem in autonomous, multi-agent robotics. In this work, distributed methods are explored for a modified weapon target assignment problem in which weapons seek to achieve a specified probability of kill on each target. Three novel cost functions are proposed which, in cases with low agent-to-target ratios, induce behaviors which may be preferable to the behaviors induced by classical cost functions. The performance of these proposed cost functions is explored in simulation of both homogeneous and heterogeneous engagement scenarios using, as an example, airborne autonomous weapons. Simulation results demonstrate that the proposed cost functions achieve the specified desired behaviors in cases with low agent-to-target ratios where efficient use of weapons is particularly important.

Additionally, a multi-objective version of the WTA problem is considered in which the quality of an assignment is dependent on both the total effectiveness of the weapons assigned to each target, and the relative timing of agents' arrival at their targets. Such timing constraints may be important in real-world scenarios where a mission planner wishes to enforce an element of surprise on each target. A fourth cost function is presented which couples weapon effectiveness and timing metrics into a combined cost. In cases where weapon-target closing speeds are limited to a certain range, this combined cost allows the inclusion of arrival time constraints in the assignment decision process. The performance of this new cost function is demonstrated through theoretical analysis and simulation. Results show that the proposed cost function balances the dual goals of optimizing effectiveness and arrival time considerations under closing speed limitations, and that a user-defined tuning parameter can be used to adjust the priority of the dual goals of sequenced arrival and achieving the desired probability of kill.

CHAPTER 1
INTRODUCTION AND BACKGROUND

As robotic swarm technology matures, there is increasing potential for its use as an offensive weapon system. Such robots may be airborne [1, 2], seaborne [3], or terrestrial [4], and could conceivably be tasked to engage a set of targets which are predefined or discovered online. These swarm weapons systems will be required to make targeting decisions with minimal or no operator oversight, will be expected to adapt to changing circumstances or agent attrition, and will need to intelligently balance the multi-variate mission objectives with which they are presented. This will require high degrees of autonomy and, likely, inter-agent coordination. In the scenarios considered here a set of potentially heterogeneous, autonomous weapons (also referred to as agents) engage a set of pre-identified targets. Each agent selects one, and only one, target from the set to engage. For the purposes of this thesis, the weapons are assumed to be air-to-ground munitions, but the algorithms, cost functions, and overall ideas presented here are generally applicable to seaborne, terrestrial, hybrid, or any other mobile robotic system as well.

The problem above, known as the canonical weapon target assignment (WTA) problem was first described by Manne in 1957 [5] and seeks to find an optimal (possibly unique) mapping Q of weapons to targets of known value v_j so as to minimize the expected value of targets surviving the engagement. The probability of target j being destroyed by weapon i (if targeted by it) is denoted $Pk_{i,j}$ and is referred to as the weapon effectiveness. For the remainder of this thesis, Pk is used as an abbreviation of "probability of kill". Assumptions about the nature of Pk and how it combines when agents cooperate are detailed in Ch. 2.

The WTA problem has two notable differences from the generalized assignment problem. The first is that zero, one, or many agents can be assigned the same target, whereas in the generalized assignment problem, each target would be assigned to exactly

one agent[6]. The second is that the utility functions used in the generalized assignment problem are summations over values associated with each agent/target pair, but in the weapon target assignment problem there is a probabilistic element that introduces multiplication into the utility functions.

Many surveys have been done on the different approaches taken to this and related problems. A commonly cited survey by Matlin [7] starts off by defining a taxonomy for different models of the problem. While the paper itself is dated, the taxonomy it introduces is useful for categorizing WTA problems. The weapons and targets are each defined based on three criteria. For Matlin these criteria are scope, reach and commitment. Scope is the number of different weapon types, where engagements with a scope of one are referred to as pair-wise homogeneous if the target scope is also one and agent-wise homogeneous otherwise. Engagements with a weapon scope greater than one are referred to as heterogeneous, regardless of the target scope. Reach, referred to here as the agent's domain, is the set of targets that are valid assignments for each weapon. The three general cases specified by Matlin are all targets being reachable by all agents, only some targets being reachable by each agent, and the reachability not being explicitly known. As detailed in the modeling section, the scenarios considered here fall into the third category of reach. Commitment is more frequently referred to in the literature as static versus dynamic WTA. Committed (or static) problems assume that once launched weapons cannot be reassigned, either due to the nature of the weapons or due to the lack of sufficiently fast battle damage assessment (BDA) to evaluate the evolving situation. The scenarios considered here are assumed to be dynamic.

The target criteria are target type, valuation and defenses. Target types are classified as either point targets or area targets with the distinction that a point target can be destroyed by a single agent whereas area targets will need multiple weapons to destroy them. In this work, exclusively point targets are considered as it is assumed that all weapons have a non-zero probability of destroying any given target. The second criteria is valuation for

which four categories are defined. In model A all targets are of equal value. Model B assumes a known ranking of target importance but no associated numeric values. Model C is concerned with targets having numerical values on a single scale, allowing for like-to-like comparison. Finally, model D considers intrinsic as well as extrinsic value. An example of a model D scenario would be an anti-aircraft battery protecting a high-value target —there might be no value to destroying the battery aside from making the main target more vulnerable. Model C is used to describe the value of the targets in this work.

Another useful survey can be found in [8] which lists several of the most common solution techniques for the WTA problem as well as the drawbacks of each. The first is dynamic programming which has been used many times [9, 10] but suffers from Bellman's curse of dimensionality [8, 11]. The next approach is Lagrange Multiplier methods such as those implemented in [12, 13]. The disadvantage of Lagrange Multiplier methods is that they require differentiable cost functions and so aren't always applicable depending on the problem model. Game theory has been applied to WTA problems with some success [14–16]. This approach, while requiring knowledge of the strategy models for all agents (or targets if applicable), has favorable scalability properties to large problems. Neural networks have seen success at solving WTA problems [17, 18], but the need for training may make them unsuitable for online WTA problems. Genetic/stochastic algorithms [19–22] are useful due to their ability to escape local optima, but it can be difficult to evaluate their performance or provide convergence guarantees due to their stochastic nature. Simulated annealing [23], as described in Chapter 3, falls into this category of algorithm. Many other classes of algorithms have been used on WTA problems as well such as auction [24, 25], fuzzy decision making [26], Tabu Search [27], and branch and bound [28]. Additionally, branch and bound has been applied to generalized assignment problems [29], mixed-integer problems [30], and constraint optimization problems [31], all of which are generalizations of the WTA problem.

These approaches have all been applied to the dynamic WTA problem as well as to the

static WTA problem. In the dynamic problem the optimal assignment changes as the situation evolves which allows for increased effectiveness from a given weapon set [32]. For instance, if a target is destroyed then the other agents targeting it will switch to new targets if possible. If a new solution can be calculated much faster than the time scale on which the agents can act then the dynamic problem is reduced to continuously solving the static problem at new initial conditions. Additionally, the WTA problem has been considered in both centralized and distributed settings [33]. In a distributed setting the agents must share information in order to achieve common objectives, which introduces elements from the field of cooperative control[34]. Some authors (for instance [35]) simply employ agents as centralized planners which broadcast their plans to other agents, after which auction or negotiation mechanisms resolve disparities. This approach becomes problematic if communication is limited, expensive, or unreliable, or if there is a large number of agents.

The research described here differs from most previous investigations of WTA in that the primary goal is not to minimize the expected value of the targets directly but instead to induce the agents to achieve a specified total kill probability $Pk_{\mathrm{des},j}$ on each target. However, this variant is not unprecedented. In [22], a stochastic algorithm, specifically an "ant colony algorithm", was used to meet specified kill probabilities. The most prominent difference between the work presented here and the traditional WTA problem is that in the form considered here there is no added benefit to engaging a target past the point where its specified Pk has been met. This is referred to as the *modified* WTA problem. In the case where all desired Pk's are set to be 100%, the modified weapon target assignment problem reduces to the traditional problem. If there are sufficient agents to meet all goals then the specific assignment is not of great importance and traditional WTA solution algorithms applied to the modified WTA problem perform quite well. However, of particular interest in this work is the case in which there are insufficient agents to meet the desired Pk's on all targets. In this case there are many partial assignment solutions, some of which may

be more desirable than others. In some instances it may be a secondary goal to balance achieving the desired Pk's with engaging as many targets as possible, while in other cases there should be no engagement of lower priority targets until the desired Pk's ($Pk_{\mathrm{des},j}$) on all higher priority targets have been met. It is possible that in some situations it is desirable simply to meet as many $Pk_{\mathrm{des},j}$'s as possible, without regard to target priority. It will be shown that with proper selection of a cost function, the same optimization algorithms can serve to induce all of these desired behaviors as well as solving the traditional WTA problem.

Additionally, previous work on the WTA problem has largely focused solely on the expected survival value of the targets. However, in a practical context, it may be important to consider not only the expected value but aspects such as the arrival timing of the agents. It is for this reason that in Chapter 4 a multi-objective cost function is introduced. Multi-objective forms of the weapon target assignment problem have also been previously studied. Gelenbe *et. al.* [17] included a cost to deploying a weapon so that an agent wouldn't be assigned to a target if the improvement in the objective function didn't balance the cost associated with engaging that target. Multi-objective assignment can be divided into two general categories. In the first category, all objectives are considered when making the assignment. Examples include [17] and the current work in Chapter 3. Alternatively some formulations, such as [36], will find an assignment based on some objectives and then optimize additional parameters for the remaining objectives given the selected assignment. In this thesis, a linear scalarization is explored to allow single-objective solution algorithms to solve the multi-objective problem. In particular a weighted-sum scalarization [37] is used as it is not computationally complex and allows for an operator to express the relative priority of objectives.

As an example of additional objectives, it may be highly undesirable for one agent to reach the target well in advance of the remaining agents, as this may trigger defensive measures that reduce the effectiveness of subsequent agents. Alternatively, it may be

desired that agents arrive at regularly-spaced time intervals to allow for battle damage assessment between agent arrivals so as to prevent agents from being expended unnecessarily. If the arrival times of each agent can be adjusted without bound, then the agent arrival times can be coordinated on a per-target basis and the assignment and arrival time problems become completely decoupled. However, if the agents can only adjust their arrival times within certain bounds, then a given assignment may restrict the ability to achieve the desired degree of simultaneity or sequenced arrival. In this case, the assignment and arrival time optimization problems become coupled, and the WTA problem becomes a more complex multi-objective optimization problem [38]. It is important to note that in any practical scenario, agents are subject to arrival time bounds due to factors such as limited onboard fuel and minimum and maximum speed constraints.

Outside of the WTA context, time-domain coordination of multi-agent systems has been studied in great detail and is typically referred to as the rendezvous problem and is a subproblem of cooperative control[39]. Several methods have been proposed for solving rendezvous problems including optimal path-planning through the discretized state space [40, 41] and use of control Lyapunov functions [42]. One example of coupled tasking and timing in a WTA context [43] investigated auction algorithms as well as the proper way of framing such a problem. In a scenario in which agents have to search for a known number of targets before engaging them [44], a two-stage approach was taken in which agents narrow the search space by generating individual plans that are satisfactory to themselves and then selecting the minimum global cost from the unions of these plans. Many of the same approaches that have been implemented for the traditional WTA problem have also been applied to assignment with timing considerations such as genetic algorithms [45] and mixed-integer linear program solvers [46].

A final aspect of the WTA problem explored in this thesis stems from the dynamic scenario in which agents may arrive at, and destroy, a target before others pursuing the same target arrive. It was observed in early simulation experiments in the current work that

if the arrival of the agents are spread out through time and weapons are allowed to update their assignments throughout the engagement, then the actual achieved Pk on each target computed at the end of the scenario is greater than or equal to the calculated Pk on each target at any point during engagement. Consider that, for all but one agent engaging a given target, there exists a non-zero probability that its target will be destroyed before it arrives. Assuming that the agents are able to modify their target selection at any time, there is a non-zero probability that these agents will ultimately engage a different target. Accounting for this possibility allows for a more accurate measure of combined effectiveness. If the assignments are made based on an inaccurate measure of the combined effectiveness, then it is possible that they are making suboptimal decisions. While traditionally neglected in academic work, contingency targets and retargeting behavior have been considered by military researchers [47–49].

To address this issue, the selection of multiple targets is investigated in the final chapter of this thesis. Specifically, each weapon will select a reserve target and a flight path to reach the primary and secondary targets. By computing the probability that each weapon will become redundant on its primary target, a more accurate Pk can be calculated recursively. Assigning multiple targets to a single weapon system is not novel as can be seen in a paper by Rosenberger [50]. However, in that work the systems are capable of engaging multiple targets in series rather than being able to only engage the first undestroyed target it reaches. Despite this difference, [50] presents a good explanation of applying branch and bound methods for this type of problem. Typically branch and bound algorithms are implemented in a centralized manner, however there has been some work, for example by Bader [51], on parallel methods. While Bader's work is in large part focused on modeling of various computer architectures, it also contains sound insight into solving assignment problems on distributed memory systems using branch and bound techniques.

1.1 Contributions

This thesis makes contributions to several aspects of the weapon target assignment problem, as listed below.

1. A primary contribution of this work lies in demonstrating how cost functions can be designed to elicit desired emergent behaviors to address specific tactical priorities. While the traditional weapon target assignment problem has generated a substantial amount of academic literature, the majority of solution algorithms cannot be directly applied to military scenarios. For example, in military engagements, targets typically have a prespecified desired probability of kill ($Pk_{des,j}$) that participants in the engagement seek to meet, rather than a value expressed in arbitrary units. By framing the WTA problem in terms of the Pk parameters that define military engagement scenarios, and by defining the modified WTA problem, this work allows WTA to be solved in a highly relevant and practical context.

2. Another contribution of this thesis lies in the extension of cost functions to include goals that are coupled with, but separate from, the achieved Pk. An example of this is presented in the form of a cost function that seeks to achieve both the desired Pk objectives and to enforce operator-specified arrival time intervals. The desired interval, $S \geq 0$, represents the elapsed time between successive agents engaging the same target. This may, for instance, allow for battle damage assessment (BDA) to be performed. Alternatively, the operator may specify $S = 0$ so as to induce simultaneous arrival for tactical reasons. This further allows the WTA algorithms developed here to move closer to practical use in military scenarios. It is shown through analysis and simulation that agents can find assignments that balance objectives of different types even when those objectives potentially conflict with each other.

3. A third contribution lies in the formulation and implementation of a branch and bound algorithm that allows for distributed computation in an efficient and provably

correct way. This algorithm allows the agents to obtain the optimal, or a near-optimal, solution to the modified WTA problem specifically considering the arrival time objective. In particular, relaxation methods are developed that allow branch and bound to be applied to the combined assignment-arrival time optimization problem.

4. The final contribution lies in an extension to the assignment optimization algorithms to account for the inclusion of multiple backup options. This allows agents to consider contingency target(s) to engage if their primary target is destroyed before the agent reaches it. The methods proposed here allow for a more accurate estimate of achieved Pk to be used by the agents throughout a dynamic engagement for assignment decision-making. By using a more accurate estimate of the achieved Pk, solution algorithms can avoid expending resources where they are not in fact needed.

CHAPTER 2
MODELING AND ENGAGEMENT SIMULATION

Formulation of WTA solution algorithms, and evaluation of algorithm performance, requires defining of agent and target models which capture dynamic properties, communication capabilities, and any underlying agent attrition process. This section details the major agent and target modeling assumptions used throughout the remainder of this thesis to define and evaluate the proposed optimization algorithms. Note that the focus of this work is on air-to-ground engagement scenarios, and thus the weapon dynamic model includes basic characteristics common to aerospace vehicles such as turn rate limits and a maximum glide ratio. In addition, weapon effectiveness and attrition rates are defined in a general manner such that both homogeneous and heterogeneous cases may be simulated. Throughout this work, these models are exercised in example and Monte Carlo simulations to evaluate performance of the proposed cost functions.

2.1 Dynamic Model

It is assumed that agents can only engage targets within a certain radius of their current position, where this radius can vary with time (or altitude). This reachability constraint can be used to model the behavior of a variety of aerospace systems including projectiles, gliding munitions, unmanned aerial vehicles, guided parachutes, or other devices.

Define an inertial reference frame using a standard flat earth approximation and a North-East-Down coordinate system. For this work, targets are represented as point locations on the ground and assumed static (although this is not a limiting assumption for the algorithms developed here). Agent dynamics are modeled as a 3-degree-of-freedom (3DOF) point mass, where a speed in the $I_I - J_I$ plane of $v_{\mathrm{h,nom}} \pm \Delta V$ is assumed where v_h is a nominal velocity and $2\Delta V$ is the width of their achievable velocity range. Each agent can vary its

velocity within its range to best achieve its desired arrival time. Thus, the resulting model is a 3DOF system in which vertical position z_i and vertical velocity $\dot{z}_i$, horizontal position and velocity v_i, and agent heading ψ_i and heading rate $\dot{\psi}_i$ define the degrees of freedom. Agents always seek to travel from their current position in a straight line path toward the target they are pursuing, and thus upon selecting target q_i a commanded heading is defined according to,

$$\psi_{i,\text{com}} = atan\left(\frac{y_j - y_i}{x_j - x_i}\right) \tag{2.1}$$

where (x_j, y_j) are the target coordinates and (x_i, j_i) are the agent coordinates. The agent is assumed to track this commanded heading according to a first order lag such that,

$$\dot{\psi}_i = k_\psi(\psi_{i,\text{com}} - \psi_i) \tag{2.2}$$

where $k_\psi > 0$. Note that $\dot{\psi}_i$ is bounded by a saturation function to a maximum magnitude of $\dot{\psi}_{max}$ to model turn rate or, to some approximation, lateral acceleration constraints. It is important to note that this straight-line travel assumption is not meant to be limiting —any path generated by a secondary path planner has an equivalent straight-line path and horizontal closing speed.

To define the vertical dynamics, it is assumed that agents can instantaneously change their descent rate. Agents select their descent rate so as to reach zero altitude at the target location, and thus,

$$\dot{z}_i = -\frac{v_i}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} z_i \tag{2.3}$$

A maximum glide ratio limitation, denoted by γ_{max}, is imposed on all agents which restricts range as a function of altitude and horizontal velocity. In particular, maximum glide is

imposed by the condition,

$$\frac{v_i}{\dot{z}} \leq \gamma_{max} \tag{2.4}$$

If reaching a target requires a descent rate slower than that achievable due to the maximum glide ratio, the agent will not consider engaging that target (i.e., it will be considered out of range and will not be an option during target selection).

2.2 Weapon Effectiveness

It is assumed that there is a known kill probability that target j will be destroyed by weapon i if targeted by it. This value is denoted as the weapon effectiveness $Pk_{i,j}$ where $0 < Pk_{i,j} < 1$. Given a set of N weapons pursuing target j, the Pk's of each agent combine as independent probabilities to form the expected Pk on target j according to Eq. (2.5). In Eq. (2.5) it is assumed that agents will reach their desired targets.

$$Pk_{\Sigma,j} = 1 - \prod_{\substack{i=1 \\ q_i=j}}^{N} (1 - Pk_{i,j}) \tag{2.5}$$

Introducing a probability of attrition $Pa_{i,j}$, which describes the likelihood that agent i will be attrited on the current path to target j, the total Pk against a target can be written as shown in Eq. (2.6). The expected value (EV) for a given assignment can then be calculated according to Eq. (2.7).

$$Pk_{\Sigma,j} = 1 - \prod_{\substack{i=1 \\ q(i)=j}}^{N} (1 - Pk_{i,j} + Pk_{i,j}Pa_{i,j}) \tag{2.6}$$

$$EV = \sum_{j=1}^{M} (1 - Pk_{\Sigma,j}) V_j \tag{2.7}$$

2.3 Engagement Simulation

Throughout this thesis it is assumed that each agent has access to certain information prior to and during the engagement. First, it is assumed that at the beginning of the engagement scenario all weapons are made aware of the weapon effectiveness values for each weapon/target pair, the desired Pk's on all targets, and the locations of all targets. This information is static and thus can be uploaded to the weapons just prior to the engagement. Furthermore, each weapon maintains a (possibly imperfect) model of the current assignment of weapons to targets Q, which is updated according to the communication model to follow. Finally, it is assumed that the survival states of all weapons and targets are known to the weapons throughout the engagement. In a realistic scenario, knowledge of target survival would require a battle damage assessment (BDA), in this case likely performed by a third party, and thus would be subject to some delay. The impact of this delay on the assignment problem is highly dependent on the specifics of the engagement scenario. A full treatment of the effects of this delay due to BDA is beyond the scope of this work and, in general, it is assumed that agents receive information about target survival immediately. However, in Chapter 4 of this thesis an assignment algorithm that incorporates sequential arrival objectives is proposed, in part to allow for BDA to take place between successive agent arrivals at a target.

The manner in which agents make decisions with respect to one another, and with respect to the rate at which they communicate, has important implications for convergence of the assignment algorithms proposed here. A possible synchronous method, called turn-taking, makes use of so-called decision rounds. In a decision round each agent makes a single target selection, taking turns in a pre-specified order and communicating the current assignment state (as described in Section 2.5) after each agent's selection. An external signal (such as a GPS clock signal) may be used as a timing reference and each agent assigned a specific set of times for target selection. Alternatively, agents may act in

a completely asynchronous manner, making new selections as desired with no *a priori* scheduling, ordering, or timing. Both methods have benefits and drawbacks with respect to the optimization algorithms proposed here. Turn-taking can provide convergence guarantees in some cases, but requires the use of some synchronization mechanism (for instance, an external clock signal) to provide a common timing reference for all agents. Asynchronous selection has the benefit of being simpler to implement, but can lead to problems with limit cycles or bandwagoning behavior [15] and does not offer the convergence guarantees provided in some cases by turn-taking. For the purposes of this thesis, it is assumed that agents use a turn-taking scheme for target selection enabled by an external timing signal for synchronization, where the ordering is prescribed prior to the engagement.

In the simulation model employed here, when an agent reaches its target, that target is destroyed with a probability equal to its weapon effectiveness. Because decision rounds are assumed to continually take place until all agents reach their targets, if a target is destroyed other agents will switch to a new target if one is reachable from their current state as the desired Pk on an already destroyed target resets to zero. Similarly, if an agent fails to destroy a target, agents engaging lower priority targets will potentially switch to that target as the current expected Pk on it will have decreased.

2.4 Attrition

There are many possible ways to model weapon attrition. For example, a simple model for attrition may assume that all weapons have a constant probability of attrition at all times. This path-independent model is actually equivalent to reducing the weapon effectiveness by the attrition probability. For the purposes of this paper, a path-dependent weapon attrition model is used such that, for every fixed distance d_a traveled, the weapon has a finite probability Pa of being killed. Thus, assuming weapon i travels to target j in (approximately) a straight line, the total distance from i to j given by

$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$ can be divided into subintervals of length d_a. Let this be given by,

$$d_{int} = d_{ij}/d_a \tag{2.8}$$

The total probability of i being attrited on its path to target j is then given as,

$$Pa_{i,j} = 1 - (1 - Pa)^{d_{int}} \tag{2.9}$$

$Pa_{i,j}$ is used during computation of the total achieved Pk by conditioning on the attrition probability as shown in equation (2.6). Also note that during simulation, agents are randomly attrited over each finite interval traveled according to the attrition model above.

A non-zero attrition rate introduces several complications into the assignment problem. In particular, even homogeneous sets of agents (having identical weapon effectiveness values) are effectively heterogeneous due to attrition if they are at different locations with respect to the targets. Additionally, the optimal assignment may change throughout the engagement as agents move and the overall weapon effectiveness (accounting for attrition probability) changes. This effect becomes particularly pronounced as the value of Pa increases.

2.5 Communication

For the purposes of this thesis, it is assumed that agents can communicate with other agents within a limited radius. As agents receive information that updates their model of the engagement state, they rebroadcast this information to other agents, forming a daisy-chained network.

Each information packet broadcast by an agent contains its current belief about which target each agent is assigned to, an estimate of the attrition probability (Pa_{i,q_i}) for each agent, an estimated arrival time for each agent at its target, and a number specifying how

many intermediate hops this information took to reach agent i (i.e., number of times it was rebroadcast). This last value is important because it provides a measure of confidence in the information provided about other agents, which becomes useful in resolving disparities if an agent receives contradictory messages about another agent. If this happens, the agent trusts the information that was broadcast more recently.

In addition to these $4N$ values (where N is the number of agents), in Ch. 4 estimated arrival times at the target also need to be communicated as well as, if the agents are utilizing the branch and bound algorithm described in Section 3.1, information necessary to reconcile local copies of the search tree as the algorithm progresses. Details regarding this information transmission are provided in the aforementioned section.

If the network is connected and message passing occurs with sufficient frequency, all agents will be able to communicate at least indirectly with all other agents. If the graph of the network becomes disconnected then there will be no way for agents to know the current target assignments of other agents in the disconnected portion. Maintenance of connectivity is not considered in this work; however, others have incorporated it into their objective functions [52]. Examples of connected and disconnected (bifurcated) networks can be seen in Figure 2.1. In this figure, the communication radius is denoted by the circle surrounding the agent location, and a graph of the network (where edges denote connectivity and nodes denote agents) is shown below. If a network bifurcates, agents will overtarget high priority targets as the agents of one sub-network will not have knowledge of the assignments of agents in the other sub-network. As a result, higher priority targets will be engaged by more agents than necessary to meet the desired Pk's. As the agents converge on their targets the network should reconnect, but there is the risk that alternative targets have become unreachable in the interim. In addition, sudden reconnection of the network can lead to limit cycles in the assignments in some cases. Thus limited communication can significantly reduce performance in instances where the network becomes disconnected.

Finally, it is assumed that communication occurs at a rate much faster than

decision-making, and that decision-making occurs on a much faster time scale than that with which the agents approach the target set. While removing this time scale separation poses interesting questions, such investigations are beyond the scope of this work. Note that in all simulations performed here a communication radius of 600 m is used. This value is selected to be large enough to ensure network connectivity in the majority of simulation cases, but small enough that rebroadcasting must occur and disconnections do happen.

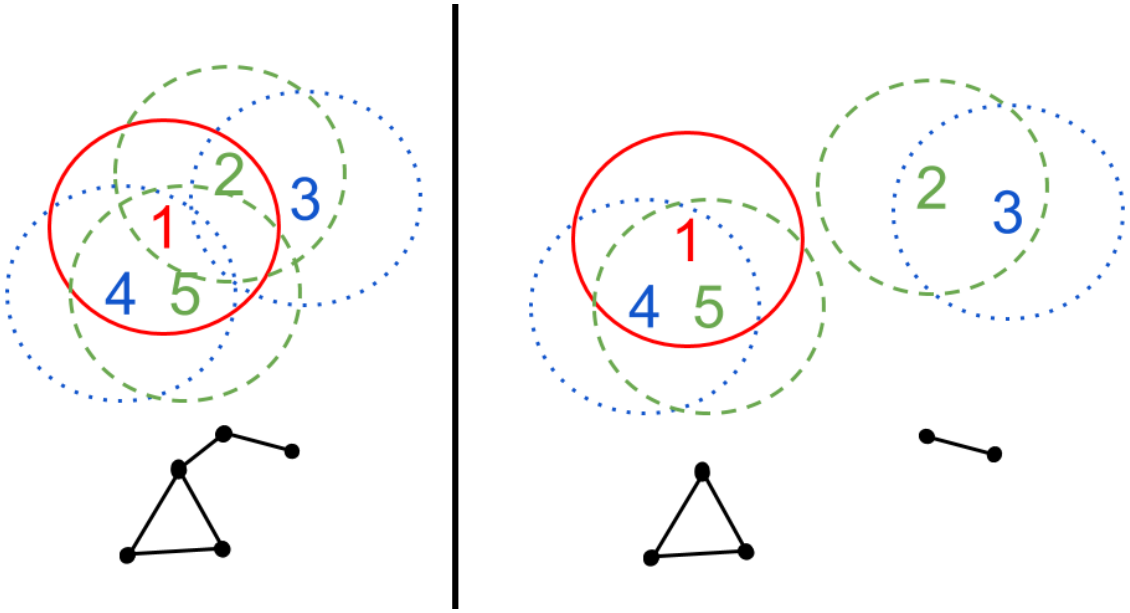

Figure 2.1: Examples of connected (left) and disconnected (right) Networks

CHAPTER 3
PROBABILITY OF KILL SOLUTION ALGORITHMS

3.1 Optimization Algorithms

In this thesis the focus is not on optimizing over a given cost function, but rather taking a desired collective agent behavior or outcome, and designing a cost function that induces that behavior. Agents can then employ an optimization algorithm that attempts to minimize this cost. This section describes the optimization algorithms employed by each agent during targeting decisions, while the next section describes several novel cost functions designed to produce specific desired behaviors on the part of the agent set.

Solution of the optimization problems formulated here may theoretically employ any deterministic or stochastic combinatorial optimization routine (for instance, see those described in [53]). In this thesis, only a special set of distributed optimization algorithms are investigated. These solution algorithms are selected such that they can operate under the agent communication constraints discussed in Section 2.5. One important consideration for practical implementations is that decision-making should be robust to attrition in that no single agent (or group of agents) is responsible for solving the entire global optimization problem and transmitting the solution, since the loss of this agent could destroy the engagement capacity of everyone. Instead, a decision-making scheme in which each agent is responsible for deciding its best action is preferred in this environment, as such a scheme may degrade and adapt gracefully under attrition or network disconnection. Another benefit of the distributed schemes studied here is their ability to handle large optimization problems without excessive computational burden —since each agent has control only over its own actions, the dimensionality of the action space is reduced by a factor of at least N over a centralized scheme which considers all possible actions of all agents. As a result of these benefits, only a specific set of

distributed optimization algorithms are considered and evaluated with respect to the proposed cost functions discussed in the next section.

3.1.1 Greedy Search

The first solution algorithm investigated is a straightforward greedy search across the space of possible assignments. Greedy search has the advantages of being simple to implement, deterministic, and computationally lightweight. For these reasons, greedy search algorithms have been commonly applied to WTA problems [5, 54, 55]. While the cost functions proposed in the next section are in general susceptible to exhibiting local minima, it will be shown that many instances of the problem under investigation actually do exhibit properties which ensure greedy search will converge to the global optimum.

Let Q denote the current weapon-target pairing such that $Q = \{(i,j)|i \in I, j \in J\}$ where $I = \{1, ...N\}$ and $J \subseteq \{1, ..., M\}$. Note that the indices i representing the agent number in Q form a permutation of I, while only a subset of targets in J may be represented in Q. In the implementation of greedy search used here, each agent i selects a target j that minimizes the resulting cost. This search function can summarized as,

$$q_i = \arg\min_{j \in J} C\left(Q_j\right) \tag{3.1}$$

where q_i is the target selected by agent i, Q_j represents the resulting global weapon-target assignment if agent i selects target j, and $C(Q_j)$ represents the cost of the global assignment Q_j. Even when greedy search does not find the globally optimal solution, it reliably converges to a local minimum which may be sufficient for many practical applications. This process is said to be "greedy" because it selects the target which will minimize the resulting total cost C by maximizing its current contribution, irrespective of the subsequent decisions made by other agents.

3.1.2 Simulated Annealing

Simulated Annealing is a stochastic optimization scheme which can be used to solve for the global minimum of a non-convex cost function. Recall that agents make new target selections in continuously recurring decision rounds. When using Simulated Annealing, each time an agent makes a new decision it compares the resulting cost from choosing a new random target to the current cost (with its currently selected target). If the candidate assignment has a lower cost than the current assignment, it will switch to the new target. If the candidate assignment has a higher cost it will switch to the new target with a probability drawn from a tunable probability function. The switching probability is given in equation (3.2) where q_i is the current assignment of agent *i*, q_i' is the candidate assignment, and $C\left(q_i\right)$ is the cost function evaluated for the current assignment.

$$P_s = \frac{2C\left(q_i\right)}{C_{max}} exp\left(\frac{C\left(q_i\right) - C\left(q_i'\right)}{\hat{t}}\right) \tag{3.2}$$

Note that the normalization constant C_{max} in (3.2) imposes the limit $P_s \leq 1$. The temperature variable in (3.2), denoted $\hat{t}$, decreases monotonically with time (or any other independent variable that strictly increases as the engagement progresses). At the beginning of the engagement scenario, $\hat{t}$ is usually large allowing agents to consider higher cost states to potentially escape local minima. As the engagement progresses and the number of future decisions that can be made becomes limited, the agents should become more conservative and thus $\hat{t}$ decreases. The temperature equation used in this work is given by,

$$\hat{t} = 0.9^{t_e} \tag{3.3}$$

where t_e is the time in seconds since the beginning of the engagement. Allowing agents to temporarily increase the cost function with their selection enables a heterogeneous set of weapons to explore potential assignments that the greedy search would not reach. This

algorithm is probabilistically optimal even with local minima [56] but requires time to converge. In contrast, greedy search often converges quickly in the homogeneous case because, when each agent in turn makes its optimal decision, the resulting assignment after all agents have made one decision is globally optimal (this will be shown in Section 3.3). The simulated annealing approach thus proves appealing only when local minima are present. This is often the case with heterogeneous weapon sets (where agents have differing effectiveness values).

3.2 Cost Functions

The methodology pursued in this work is to tailor WTA cost functions to induce specific desired behaviors on the part of the agent set, who are executing a known solution algorithm. In the context of a WTA scenario, it is conceivable that an operator directing an overall engagement may be able to analyze the specifics of the scenario and, based on the relative priorities of various mission objectives, tune a cost function to generate desired agent behaviors.

In formulation of the cost functions below, a desired behavior on the part of the agent set is specified, and a cost function is designed to achieve this behavior. It is important to note that the cost functions proposed here are designed separately from reachability constraints and communications constraints, meaning these limitations are never explicitly accounted for in the cost functions themselves. For instance, the cost functions formulated here do not induce agents to maintain connectivity during the engagement. However, simulation experiments in Section 3.4 do examine performance of these algorithms under reachability and communications constraints, showing that the algorithms are functional and performance is favorable in many cases (even if somewhat degraded compared to ideal cases of unlimited reachability and perfect connectivity). Also note that all subsequent analysis of the proposed cost functions assumes that agents are exercising either the greedy search or simulated annealing algorithms described in Section 3.1. A

major advantage of the cost functions designed here is that they are largely successful in inducing relatively complex agent behaviors using simple solution schemes such as greedy search or simulated annealing.

3.2.1 Sufficiency Threshold (ST) Cost Function

In the classical WTA problem the cost of a particular assignment is equivalent to the expected value of all targets as shown in Eq. (3.4) [5].

$$C_T(Q) = EV = \sum_{j=1}^{M}(1 - Pk_{\Sigma,j})V_j \tag{3.4}$$

Note that in the above equation, referred to hereafter as the traditional WTA cost, the cost function operator C_T denotes the cost associated with a particular assignment and is thus the summation of the costs incurred by each individual agent (note that cost is initially very high, and decreases as optimization proceeds). Further note that in Eq. (3.4), the cost function value continues to decrease as the total achieved $Pk_{\Sigma,j}$ on a target increases.

In some situations the "value" of a given target *j* might depend on both the $Pk_{\mathrm{des},j}$ and the $Pk_{\Sigma,j}$ in that there should be no added incentive to engage a target that is currently meeting or exceeding its specified $Pk_{\mathrm{des},j}$. In other words, agents should have no incentive to switch to targets whose desired Pk has already been satisfied under the current assignment. This can be achieved with the so-called "Sufficiency Threshold" (ST) cost function shown in Eq. (3.5).

$$C_{ST}(Q) = \sum_{j=1}^{M} \begin{cases} 0 & Pk_{\Sigma,j} > Pk_{\mathrm{des},j} \\ \frac{Pk_{\mathrm{des},j} - Pk_{\Sigma,j}}{\left(1 - Pk_{\mathrm{des},j}\right)^{\omega}} & Pk_{\Sigma,j} \leq Pk_{\mathrm{des},j} \end{cases} \tag{3.5}$$

In contrast to Eq. (3.4), the piecewise definition in Eq. (3.5) removes the incentive to engage targets that are already sufficiently targeted, since adding additional agents to a sufficiently engaged target provides zero reduction in cost. The $Pk_{\mathrm{des},j}$ term in the denominator makes the value of a target (in terms of the potential to reduce total cost) a

non-linear function of its desired $Pk_{\text{des},j}$. The value ω can be viewed as a tuning parameter. As the value of ω increases, agents attribute more weight to $Pk_{\text{des},j}$ and thus targets with higher $Pk_{\text{des},j}$ are prioritized more. Using a high value of ω results in the behavior that agents will tend to satisfy higher $Pk_{\text{des},j}$ before engaging lower $Pk_{\text{des},j}$ targets. Note that, unless otherwise specified, all results in the present work use $\omega = 1$.

When $Pk_{\Sigma,j} < Pk_{\text{des},j} \forall j$, the ST cost function can be derived from the traditional WTA cost by replacing $1 - Pk_{\Sigma,j}$ in Eq. (3.4) with $Pk_{\text{des},j} - Pk_{\Sigma,j}$ and setting $V_j = (1 - Pk_{\text{des},j})^{-\omega}$. This means that when using the ST cost, the "value" of each target is reflected by its desired Pk. Figure 3.1, showing the equivalent WTA value from Eq. (3.4) as a function of $Pk_{\text{des},j}$, demonstrates this relationship explicitly.

Furthermore, while the WTA cost can achieve negative costs (and thus adding more agents to the engagement will always reduce the cost further), the ST cost function is lower bounded by zero when the desired Pk's on all targets have been met. This mathematical equivalence means that, if the values V_j in Eq. (3.4) are set appropriately, the traditional WTA and ST cost functions will induce identical agent behavior when $Pk_{\Sigma,j} < Pk_{\text{des},j} \forall j$.

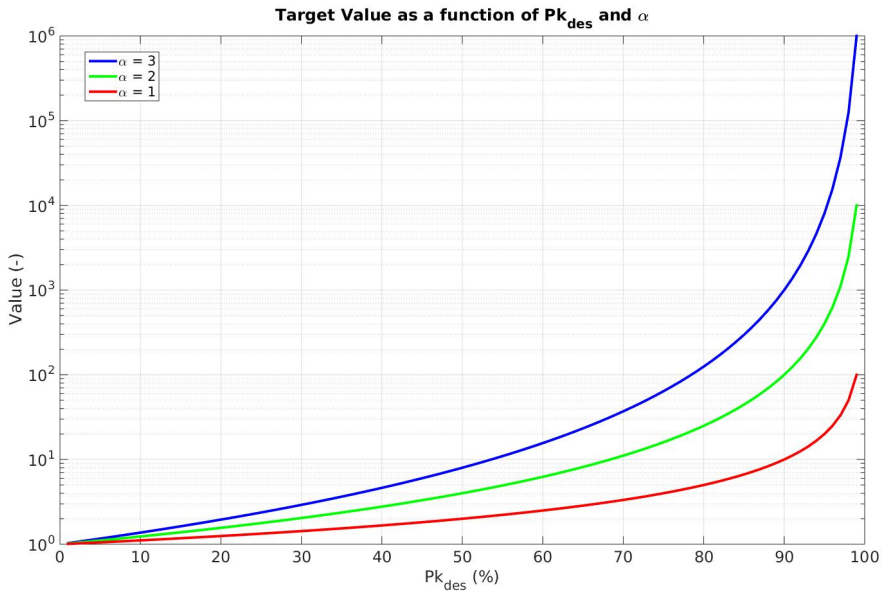

Figure 3.1: Equivalent traditional WTA value vs Pk_{des} when using ST cost function.

Now consider the case where $Pk_{\Sigma,j} > Pk_{\text{des},j}$ for some target j. Using the ST cost function, an assignment state Q with $Pk_{\Sigma,j} > Pk_{\text{des},j}$ has the same cost as an assignment state Q' with $Pk_{\Sigma,j} = Pk_{\text{des},j}$. Thus, the ST cost function will always induce agents to go after unsatisfied targets (if there are any) over satisfied ones, since selecting a satisfied targets yields zero marginal benefit. For the traditional WTA cost function, $C_T(Q) < C_T(Q')$ and thus some cost reduction may be derived from selecting an already-satisfied target. However, just because selection of a satisfied target j results in different cost reductions for the two cost functions does not mean the agents behave differently. In fact, Theorem 3.2.1 shows that agent behavior in this scenario, in which at least one of the targets has reached its desired Pk, is identical between the two cost functions.

Theorem 3.2.1. *Let $J_s = \{j \in J | Pk_{\Sigma,j} \geq Pk_{\text{des},j}\}$ be the set of all satisfied targets, and $J_u = \{j \in J | Pk_{\Sigma,j} < Pk_{\text{des},j}\}$ be the set of all unsatisfied targets. Further assume that $Pk_{i,j}$ is the same $\forall$ i,j. Then agent i implementing the traditional WTA cost function will always select a target from J_u rather than from J_s, and thus its behavior is identical to that induced by the ST cost function.*

Proof. The theorem is proved by contradiction. Let $\tilde{Pk}_{i,j} = 1 - Pk_{i,j}$ represent the complement of the effectiveness of weapon i against target j. Then, neglecting attrition, the change in cost function produced by agent i selecting target j is given by,

$$C_T\left(\tilde{Q}\right) - C_T\left(Q\right) = \frac{1 - \left(1 - \tilde{Pk}_{i,j}^{n_j+1}\right)}{1 - Pk_{\text{des},j}} - \frac{1 - \left(1 - \tilde{Pk}_{i,j}^{n_j}\right)}{1 - Pk_{\text{des},j}} = \frac{\tilde{Pk}_{i,j}^{n_j+1} - \tilde{Pk}_{i,j}^{n_j}}{1 - Pk_{\text{des},j}} \quad (3.6)$$

where Q and $\tilde{Q}$ represent the total assignment before and after agent i's selection. Note in Eq. (3.6) that because the agent set is homogeneous the effective Pk can be computed directly as the agent effectiveness raised to the power of the number of agents targeting j, denoted as n_j.

Now let target $A \in J_s$ and target $B \in J_u$. The total achieved Pk on targets A and B can be written as follows (again neglecting attrition):

$$\begin{aligned} Pk_{\Sigma,A} &= 1 - \tilde{P}k_{i,A}^{n_A} = Pk_{\text{des},A} + \tilde{P}k_{i,A}^{n_A}\epsilon_A \\ Pk_{\Sigma,B} &= 1 - \tilde{P}k_{i,B}^{n_B} = Pk_{\text{des},B} - \tilde{P}k_{i,B}^{n_B}\epsilon_B \end{aligned} \tag{3.7}$$

where $\epsilon_A \geq 0$ and $\epsilon_B > 0$ are arbitrary scalars. Note that in this homogeneous case $Pk_{i,A} = Pk_{i,B}$. If agent i selects target A over target B, the change in cost function $C_T\left(\tilde{Q}\right) - C_T\left(Q\right)$ produced by selecting target A must be more negative than that produced by selecting B. Thus, Eq. (3.7) can be rearranged and substituted into Eq. (3.6) yielding,

$$\frac{\tilde{P}k_{i,A}^{n_A}\left(\tilde{P}k_{i,A} - 1\right)}{\tilde{P}k_{i,A}^{n_A}\left(1 + \epsilon_A\right)} < \frac{\tilde{P}k_{i,B}^{n_B}\left(\tilde{P}k_{i,B} - 1\right)}{\tilde{P}k_{i,B}^{n_B}\left(1 - \epsilon_B\right)} \tag{3.8}$$

Simplifying this expression yields the relationship,

$$\frac{1}{1 + \epsilon_A} > \frac{1}{1 - \epsilon_B} \tag{3.9}$$

This equation cannot hold since $\epsilon_A \geq 0$ and $\epsilon_B > 0$, and thus the theorem is proved by contradiction.

□

The above analysis shows that the traditional WTA and ST cost functions will induce identical agent behaviors when weapon effectiveness is homogeneous across all agents and targets. For heterogeneous cases, however, the traditional WTA cost function may select an already satisfied target over an unsatisfied one. Using the ST cost function this will never happen. Consider a scenario with two targets A and B such that $Pk_{\text{des},A} = 0.35$ and $Pk_{\text{des},B} = 0.09$. Suppose there are three identical agents with effectiveness values of $Pk_{i,A} = 0.20$ and $Pk_{i,B} = 0.10$ (thus this does not satisfy the assumption in Theorem 3.2.1 that $Pk_{i,j}$ is the same $\forall i, j$). Figure 3.2 shows a decision tree for this case, where the primary values in each node show the cost function at that state and the bracketed values

denote $[Pk_{\Sigma,A}, Pk_{\Sigma,B}]$. Edges traveling left from a node indicate selecting target A, edges traveling right indicate selecting target B. As shown in Figure 3.2, the first two agents will select Target A using either the traditional WTA or ST cost functions (shown with green and red arrows respectively). The third agent, however, will select Target A again if using the traditional WTA cost function but will select Target B if using the ST cost function since the Pk on Target A has already exceeded 0.35. Thus use of the ST cost function results in meeting the desired Pk's on all targets, whereas use of the traditional WTA cost function does not.

In general, one would expect that once the desired Pk on a target is met, agents will not continue to engage that target if others are still unsatisfied. The ST cost function achieves this behavior in both homogeneous and heterogeneous cases, while the traditional WTA cost function clearly does not in the heterogeneous case. From a practical standpoint, the ST cost function is designed to operate in what is known as a non-super-additive environment [57] in which adding additional agents is not strictly beneficial, particularly if the real-world cost of constructing and deploying the agents is considered.

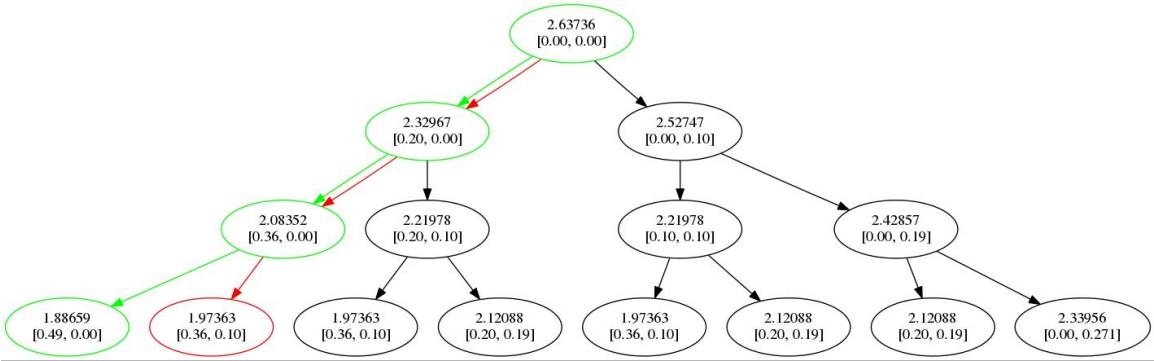

Figure 3.2: Decision tree for an example two-target, three-weapon scenario.

3.2.2 Enforced Tiering (ET) Cost Function

While the desired Pk for each target may be used to denote target priority, it is not the only mechanism to do so. In realistic engagements, targets may be sorted into priority tiers which may or may not be directly tied to the desired Pk. The desired behavior is that targets in lower priority tiers are not targeted if there are higher priority targets that can be reached and have unsatisfied $Pk_{des,j}$. Tiers can additionally be used to specify a different type of priority than desired Pk can. For example, targets could be tiered by geographic region, independent of desired Pk. Alternatively they could be tiered by capabilities, for example the first wave of agents could take out enemy air defense systems, lowering the attrition rate for subsequent waves [58].

This adherence to priority is not reflected in the formulation of the ST cost function, which will potentially incentivize an agent to select a lower priority target over a higher priority one if the weighted marginal return is better. In order to induce agents to abide by the priority tiers the ST cost function may be calculated over the subset of targets representing the highest priority tier that is not fully satisfied by the current assignment. The resulting "Enforced Tiering" (ET) cost function is given in Eq. (3.10).

The ET cost function is defined piecewise as follows. Let tiers be denoted by $T \in 1, ..., \tau$ where lower values of T denote higher priority. If any target in a higher-priority tier has not been satisfied ($Pk_{\Sigma,j} < Pk_{\mathrm{des},j}$ for some target j in a higher priority tier), then a static penalty P_T is incurred for that tier. If all targets in all tiers higher than T have been satisfied, then the ST cost function is used for tier T. In this manner, the cost function introduces a static penalty for all tiers below the highest one which is not satisfied. As tier T is satisfied (meaning all desired Pk's have been met), the static penalty on tier $T+1$ disappears and the ST cost function is applied to that tier.

$$C_{ET}(Q) = \sum_{T=1}^{\tau} \begin{cases} C_{ST}(Q_T) & (Pk_{\Sigma,j} \geq Pk_{\mathrm{des},j} \forall j \in t, \forall t < T) \\ P_T & else \end{cases} \tag{3.10}$$

The penalty P_T associated with a given tier has to be greater than the maximum value of C_{ST} evaluated over all of the targets in tier T (denoted as $C_{ST,max}(j \in T)$). When an agent satisfies the last target in tier $T-1$ the equation for $C_{ET}(Q)$ is

$$C_{ET}(Q) = -P_T + C_{ST,max}(j \in T) + \sum_{t=T}^{\tau} P_t \tag{3.11}$$

If $P_T < C_{ST,max}(j \in T)$, it is possible that fully satisfying tier $T-1$ will cause $C_{ET}(Q)$ to increase. In this case, depending on the solution algorithm used, agents may actually avoid satisfying the last target in $T-1$ since this will lead to a cost function increase. To avoid this undesirable behavior it should be ensured that $P_T > C_{ST,max}(j \in T)$ for all tiers.

A disadvantage of the enforced tiering cost function is that, by looking at only one category of targets at a time, local minima may be more prevalent. To illustrate, consider two targets A and B with $Pk_{\text{des},A} = 0.90$ and $Pk_{\text{des},B} = 0.50$ and three agents with effectiveness values of $Pk_{1,j} = 0.89, Pk_{2,j} = 0.50$ and $Pk_{3,j} = 0.10$. Let target A be in tier 1 (higher), and target B be in tier 2 (lower). Suppose agents select targets in order of decreasing effectiveness. The first agent will obviously select target A, and the second agent will select the same target since the desired Pk is not yet satisfied. The third agent selects target B, leaving a Pk of 0.10 on target B. When using the sufficiency threshold cost function and eliminating target tiers, the second agent will select the lower value target and the third agent will select the first target, meeting the desired Pk's on both targets. In essence, restricting the set of targets that an agent can select from will sometimes preclude the best assignments over the entire target set. This is the natural tradeoff for trying to enforce target tiers.

3.2.3 Completion Cost Function

The third proposed cost function is particularly suitable for heterogeneous weapons, especially in low agent-to-target ratio engagements. Suppose the desired behavior is only to meet the $Pk_{\text{des},j}$ values on as many targets as possible. This may be the case when

weapon resources are extremely limited —i.e., it may be better to satisfy the desired Pk on at least a few targets rather than achieve low Pk on all targets.

To formulate this cost function, consider agent i with effectiveness $Pk_{i,j}$ making a targeting decision. Given the current achieved $Pk_{\Sigma,j}$ on target j, agent i determines how many additional weapons of effectiveness $Pk_{i,j}$ it will take to reach $Pk_{\text{des},j}$ on target j. This is repeated for all M targets. The target for which this number is minimum is then selected as the target to be pursued.

To implement this cost function the number of weapons of effectiveness $Pk_{i,j}$ required to bring $Pk_{\Sigma,j}$ to $Pk_{j,\text{des}}$ must be computed. To calculate this, agent i considers a homogeneous case where all agents are identical to i. Recall the formula for $Pk_{\Sigma,j}$ from Eq. (2.6). In the homogeneous case, this can be simplified to Eq. (3.12) where N_j is the number of agents currently engaging target j.

$$Pk_{\Sigma,j} = 1 - (1 - Pk_{i,j} + Pk_{i,j}Pa_j)^{N_j} \tag{3.12}$$

By inverting this equation, the number of agents identical to agent i needed to satisfy $Pk_{\text{des},j}$, denoted as $N_{\text{des},j}$, can be calculated as shown in Eq. (3.13). Likewise, the number of identical agents needed to achieve the current $Pk_{\Sigma,j}$, denoted as $N_{\Sigma,j}$, can be calculated from Eq. (3.14).

$$N_{\text{des},j} = \frac{ln\,(1 - Pk_{\text{des},j})}{ln\,(1 - Pk_{i,j} + Pk_{i,j}Pa_{i,j})} \tag{3.13}$$

$$N_{\Sigma,j} = \frac{ln\,(1 - Pk_{\Sigma,j})}{ln\,(1 - Pk_{i,j} + Pk_{i,j}Pa_{i,j})} \tag{3.14}$$

By taking the difference of (3.13) and (3.14) the number of additional agents with $Pk_{i,j}$ needed to reach $Pk_{\text{des},j}$ can be calculated. This value is then set equal to the cost incurred

by agent i in selecting target j, $C_{C,i}(Q)$, for the current assignment Q according to,

$$C_{C,i}(Q_j) = \frac{ln(1 - Pk_{\text{des},j}) - ln(1 - Pk_{\Sigma,j})}{ln(1 - Pk_{i,j} + Pk_{i,j}Pa_{i,j})} \tag{3.15}$$

Eq. (3.15) is the so-called *Completion cost function*. This cost function is unique compared to the ST and ET cost formulations because the cost evaluated for a given assignment Q by agent i cannot be compared to the cost evaluated by agent k for the same Q, if i and k have different effectiveness or attrition values. This differs from the ST and ET cases in which a global cost is defined that is shared and used by all agents. When optimizing under the Completion cost, each agent has its own cost function that is unique compared to agents with other effectiveness values.

Some edge cases must be accounted for in implementation of the Completion cost function. In particular, if $Pk_{\Sigma,j} \geq Pk_{\text{des},j}$ then the cost function should return an arbitrarily high value to ensure that an agent will not select it. The exception is when target j is currently being pursued by agent i (i.e., $q_i = j$) in which case the cost function should return zero so the agent will not switch targets. The logic for handling these cases is given in Algorithm 3.1.

Algorithm 3.1 Completion cost function implementation.

Cost Function $\{q_i\}\{j\}\{Pk_{\text{des},j}\}\{Pk_{\Sigma,j}\}$

$C = \frac{ln\left(1-Pk_{\text{des},j}\right)-ln\left(1-Pk_{\Sigma,j}\right)}{ln(1-Pk_{i,j}+Pk_{i,j}Pa_{i,j})}$

if $C < 0$ **then** {Target j is sufficiently engaged}

  if $q_i == j$ **then** {Agent i already targeting j}

    return 0

  else {Do not switch to satisfied target}

    return ∞

  end if

end if

return C

3.3 Convergence Analysis

The cost functions proposed in Section 3.2 are neither linear nor quadratic, and thus detailed analysis is necessary in order to determine convergence properties. In this section, the convergence properties of each proposed cost function are explored. Furthermore, convergence properties are analyzed in terms of the desired behavior underlying the design of each cost function.

As mentioned in Section 3.1, greedy search will yield an optimal assignment if no local minima exist. A local minimum is an assignment that is not an optimal assignment but is better than all other assignments in its neighborhood. In the optimization setting considered here, the neighborhood for an agent is any state reachable by that agent unilaterally changing its assignment. Escaping a local minimum requires passing through a higher cost state to reach an even lower cost state, which greedy search will never do.

3.3.1 Homogeneous Case

It is well known that when using the traditional WTA cost function in (3.4) an exact solution can be reached by greedy search if the agents are identical [54]. Likewise, when using the ST cost function with homogeneous agents, greedy search is guaranteed to reach the global optimum. This is formalized in the following theorem.

Theorem 3.3.1. *Let a set of N homogeneous agents (denoted as the set I) having equal effectiveness values be engaging a set of M targets (denoted as the set J). Then greedy search will yield the globally optimal assignment under the Sufficiency Threshold cost function.*

Proof. For the set of homogeneous agents I and the set of targets J, there exists an optimal assignment $\hat{n_j} \geq 0, j \in 1, ..., M$ where $\hat{n_j}$ is an integer. Note that this assignment is not unique and that because the agents are identical the optimal assignment is defined only by the number of agents engaging each target.

Under greedy search, agents select the target which will yield the largest reduction in cost due to their selection. Recall Eq. (3.6) which gives the change in state cost from making a selection under the traditional cost function, where Q denotes the assignment before agent i selects target j, and $\tilde{Q}$ denotes the assignment after selection of target j. Modifying Eq. (3.6) for the Sufficiency Threshold cost function is simply a matter of accounting for the fact that there is no benefit for engaging a target beyond $Pk_{\mathrm{des},j}$. This requires adding a nonlinear "maximum" function to the numerator, as seen in Eq. (3.16).

$$C_{ST}\left(\tilde{Q}\right) - C_{ST}\left(Q\right) = \frac{max\left(\tilde{Pk}_{i,j}^{n_j+1}, 1 - Pk_{\mathrm{des},j}\right) - max\left(\tilde{Pk}_{i,j}^{n_j}, 1 - Pk_{\mathrm{des},j}\right)}{1 - Pk_{\mathrm{des},j}} \quad (3.16)$$

Suppose that the set of targets J is partitioned into targets that are engaged sufficiently to meet their $Pk_{\mathrm{des},j}$ (J_s) and those that are not (J_u). If J_s is empty, then the ST cost function is equivalent to the traditional WTA cost function with target values $V_j = (1 - Pk_{\mathrm{des},j})^{-1}$. Reference [54] has shown that greedy search converges to the global optimum using the traditional WTA cost with a homogeneous set of agents. Because the ST and traditional WTA costs are equivalent in the case of $J_s = \varnothing$, greedy search yields the optimal solution for the ST cost in this case.

Now consider the case where J_s is not empty. Defining $\tilde{Pk}_{i,j} = 1 - Pk_{i,j}$, there exists $j \in J_s$ such that $1 - \tilde{PK}_{i,j}^{n_j} \leq Pk_{\mathrm{des},j}$. Thus, if agent i selects this target,

$$C_{ST}\left(\tilde{Q}\right) - C_S T\left(Q\right) = 0 \quad (3.17)$$

Because the marginal cost reduction produced by selecting any target in J_s is zero, agents will only consider selecting targets in J_u which offer nonzero cost reduction. Thus the overall assignment problem is actually reduced to a subproblem consisting of all unsatisfied targets J_u and all agents that have not selected targets in J_s. This subproblem is equivalent to the traditional WTA problem since none of the targets in J_u has been satisfied, and thus

by Ref. [54] greedy search is optimal in this case as well.

□

Note that the above proof neglects agent attrition, which makes any homogeneous set of agents effectively heterogeneous.

While the above theorem establishes that greedy search will yield the globally optimal solution when using the ST cost, some further analysis is required to determine whether agent behavior matches the desired behavior. For the ST cost, the goal is to achieve the desired Pk on all targets, with priority given to targets whose $Pk_{\mathrm{des},j}$ is higher. It is clear by inspection of the cost function that the ST cost is at its minimum possible value if the Pk on all targets meets or exceeds the desired Pk's. Furthermore if an agent is deciding between two targets and it can increase the Pk on either of them by the same amount it will always choose the higher priority target. Finally, if the targets have sufficiently different achieved Pk's, the agents will prefer a small increase on a high priority target over a larger increase on a lower priority target. The difference required depends on the desired Pk's and the value being used for ω. Considering these factors together, the ST cost function will induce the desired behaviors.

Now consider the enforced tiering cost function. The ET cost function in the homogeneous case simply divides the problem up into subproblems that are considered consecutively (as long as P_T is set sufficiently high), and for this reason the proof provided in [54] is also valid in establishing the optimality of greedy search for this cost function. The desired behavior for the ET cost is that agents ignore lower tier targets until higher tier targets are satisfied. Targets may be arranged in tiers completely independently of their desired Pk values. Again, by inspection of the cost function it is clear that, within a tier, the ST cost is applied and thus the desired behavior will arise within a tier. Since selecting targets outside of the highest unsatisfied tier will not affect the value of $C_{ET}(Q)$, those targets will be ignored if an another assignment can lower the value of $C_{ET}(Q)$. Thus the ET cost function will also induce the appropriate behaviors.

For the Completion cost function, the desired behavior is that agents should try to achieve the desired Pk on the highest number of targets possible. In this case, $Pk_{\mathrm{des},j}$ is not considered a measure of target value, but rather just a measure of how many agents are needed on a given target before it can be considered "satisfied". The following theorem establishes that use of greedy search in conjunction with the Completion cost function achieves this desired behavior in the case of homogeneous agents.

Theorem 3.3.2. *Let a set of N homogeneous agents having equal effectiveness values be engaging a set of M targets (denoted as the set J). Then greedy search applied to the Completion cost function will maximize the cardinality of J_s, where J_s is the set of targets such that $Pk_{\Sigma,j} \geq Pk_{\mathrm{des},j}$.*

Proof. Recall the Completion cost function in Eq. (3.15). Neglecting attrition, the function for Pk_{Σ}, j can be substituted in as follows assuming homogeneous agents:

$$C_{C,i}\left(Q_j\right) = \frac{ln\left(\tilde{Pk}_j^{\hat{n}_j}\right) - ln\left(\tilde{Pk}_j^{n_j}\right)}{ln\left(\tilde{Pk}_{i,j}\right)} = \hat{n}_j - n_j \tag{3.18}$$

In Eq. 3.18, $\hat{n}_j$ indicates the number of agents needed to satisfy target *j*. Thus, agent *i* selects target $\hat{j}$ such that

$$\hat{j} = \underset{j \in J}{\arg\min}\left(\hat{n}_j - n_j\right) \tag{3.19}$$

Let the resulting assignment be denoted as Q_j. Consider the next decision step where agent $i+1$ selects a target, and let the resulting assignment after this decision be given by $Q'_{\bar{j}}$ where $\bar{j}$ is target selected by agent $i+1$. If $\bar{j} \neq \hat{j}$, then

$$C_{C,i}\left(Q'_{\bar{j}}\right) = C_{C,i}\left(Q_j\right) \tag{3.20}$$

However, if $\bar{j} = \hat{j}$, then,

$$C_{C,i}\left(Q'_{\bar{j}}\right) < C_{C,i}\left(Q_j\right) \tag{3.21}$$

as long as $C_{C,i}\left(Q_j\right) \geq 0$ (since the logic in Algorithm 3.1 prevents agents from selecting satisfied targets). Thus, agent $i+1$ will select the same target as agent i until the desired Pk is met on that target. Furthermore, inspection of Eq. 3.19 shows that, if there are no unsatisfied targets that have been partially engaged by other agents, agent i will select the target with lowest $Pk_{\mathrm{des},j}$.

Now, suppose all targets in J are sorted according to increasing $\hat{n}_j$. Further suppose that all weapons have been assigned such that targets 1 through j_s in this list have met their desired Pk. Denote this subset of targets as J_s, and denote the remaining unsatisfied targets (with $\hat{n_j} \geq \hat{n_{j_s}}$) as J_u. In order to move one target j from J_u into J_s, at least one target j' must be moved from J_s to J_u because $\hat{n}_{j'} \leq \hat{n}_j$. Thus, because the number of agents is fixed, the cardinality of J_s is maximized when it consists only of the targets with the lowest values of $Pk_{\mathrm{des},j}$.

□

As shown in the above proof, agents executing greedy search in conjunction with the Completion cost function will satisfy targets in order of lowest to highest $Pk_{\mathrm{des},j}$, thereby maximizing the total number of targets satisfied.

3.3.2 Heterogeneous Case

If agents have different effectiveness values, it is possible for the greedy search algorithm to miss the optimal solution. Consider the scenario in which there are four agents of four different types and three targets, each with a different $Pk_{\mathrm{des},j}$. The weapon effectiveness matrix for this example scenario is given in Table 3.1.

Table 3.1: Weapon effectiveness matrix for example heterogeneous case.

	High Tier Pk_{des} = 90%	Med. Tier Pk_{des} = 80%	Low Tier Pk_{des} = 70%
Weapon Type A	40%	69%	1%
Weapon Type B	40%	1%	69%
Weapon Type C	40%	31%	1%
Weapon Type D	40%	1%	44%

Suppose the ST cost function is employed. The assignment that minimizes the total cost is for the agent of type A to select the second target, the agent of type B select the third target and the remaining agents to select the first target. This assignment gives $Pk_{\Sigma,1} = 0.64$, $Pk_{\Sigma,2} = 0.69$, and $Pk_{\Sigma,3} = 0.69$. However, if the agents take turns selecting targets from A to D and a greedy search algorithm is used then agents A and B will select the first target, agent C will select the second target and agent D will select the third target. The results of this assignment are $Pk_{\Sigma,1} = 0.64$, $Pk_{\Sigma,2} = 0.31$, and $Pk_{\Sigma,3} = 0.44$, which yields significantly worse total cost. From this simple example it is clear that if the $Pk_{i,j}$'s of every agent/target pair are not equal then local minima may exist for the ST and ET cost functions, meaning greedy search may not find the globally optimal solution.

As can be seen from this example, the order in which agents select their targets is important when using greedy search. In general, agents with higher effectiveness should select first as they can have the biggest impact on the cost function (this has been referred to as "maximum-gain scheduling" [59]). However, if agents of one type are most effective against one subset of targets and agents of another type are most effective against another subset of targets, there is no clear ordering and a random ordering could be selected. Alternatively, the selection ordering could potentially be changed each decision round, which may improve convergence to the global optimum. Note that when using simulated annealing rather than greedy search selection ordering matters less due to the ability to

escape local minima.

3.4 Results

A comprehensive set of simulation results are presented in order to demonstrate behavior of the proposed cost functions and perform comparisons against the traditional WTA cost. In addition, simulation results of heterogeneous engagements are discussed in order to evaluate the effects of local minima with respect to the two solution algorithms.

For all simulations in this section, when the simulation is initialized targets are randomly distributed over a square area of 750 meters by 750 meters. Agents are also randomly distributed over an area of the same dimensions but located 2,500 meters away. All agents are initialized to an altitude of 500 meters. This geometry is depicted in Figure 3.3. All results shown in this section use values of $v_h = 50.0\,(m/s)$, $\dot{\psi}_{max} = 1.265\,\left(\frac{rad}{s^2}\right)$, and $\gamma_{max} = 6.0$. Decision rounds are assumed to occur once every 100 ms.

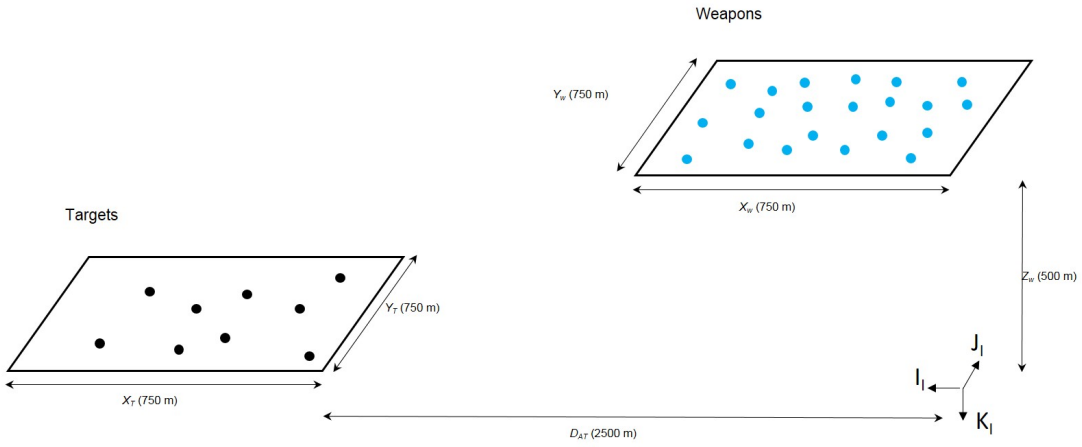

Figure 3.3: Engagement geometry initial conditions.

3.4.1 Example Simulation Results

3.4.1.1 Homogeneous Case

A first example case explores agent behavior with the ST cost function in a homogeneous scenario involving 9 targets and 27 agents. The greedy solution algorithm is employed during target selection. In this example, three targets have $Pk_{\mathrm{des,j}} = 0.90$, three have $Pk_{\mathrm{des,j}} = 0.80$, and three have $Pk_{\mathrm{des,j}} = 0.70$. Weapons are homogeneous with an effectiveness of $Pk_{i,j} = 0.50$ for all weapon-target pairs, but as mentioned above varying $Pa_{i,j}$'s introduce some heterogeneity. Throughout this section, agents are indexed $1, ..., N$ while targets are indexed $1, ..., M$ with higher Pk_{des} having lower indices.

Figure 3.4 shows an example simulation time history for this engagement. To read these plots, note that the system state after the first decision round is shown at the left of each plot (y axis). Moving along the positive x axis shows the time evolution of the achieved Pk and assignments as the engagement progresses. The top plot of Figure 3.4 shows the time evolution of $Pk_{\Sigma,j}$ on each target, where color represents the magnitude of $Pk_{\Sigma,j}$ and time histories end when a target is destroyed. The bottom plot shows the target selected by each agent at each decision round, where again moving along the x-axis indicates later decision rounds and the end of the time history indicates the agent is destroyed. Note that because weapons are homogeneous, agents converge to the optimal assignment within the first decision round and remain in this configuration until the first target is destroyed around 40 sec. This causes several agents to switch their selected targets. Also note that agents abandon target 9 (the lowest value target) near the end of the engagement scenario to pursue higher value targets, and thus target 9 has an achieved Pk of zero starting around 42 sec and survives the engagement.

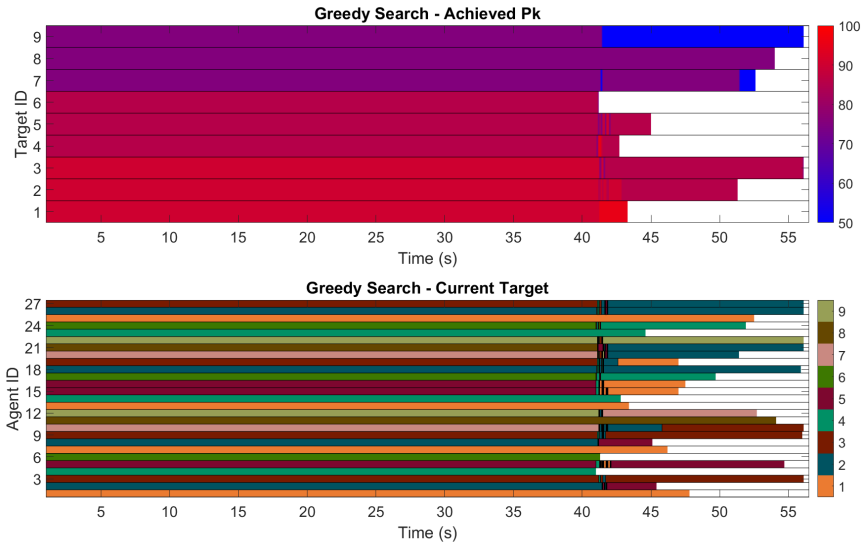

Figure 3.4: Example simulation using ST cost function. (Top) Target $Pk_{\Sigma,j}$ vs time. (Bottom) Agent assignment vs time.

A second example highlights the effect of attrition on switching behavior. This scenario is identical to the previous example except in this case four agents are artificially attrited 20 sec into the engagement. Figure 3.5 shows the decision and achieved Pk time histories. When this attrition occurs, agents engaging medium and low priority targets switch to higher $Pk_{des,j}$ targets because doing so reaches a lower cost state. This example demonstrates the robustness of the solution approach to changes in the environment. Although about 15% of agents are artificially destroyed, the cooperative control scheme is able to reallocate agents effectively to maintain the desired Pk level on high priority targets. Note that it can be seen between $t = 15s$ and $t = 20s$ that $Pk_{\Sigma,4}$ and $Pk_{\Sigma,5}$ increased appreciably. This is due to the cumulative probability of attrition decreasing as the agents approach their targets. Also note the significant switching behavior at the end of the engagement as targets are destroyed and the remaining agents switch away from them.

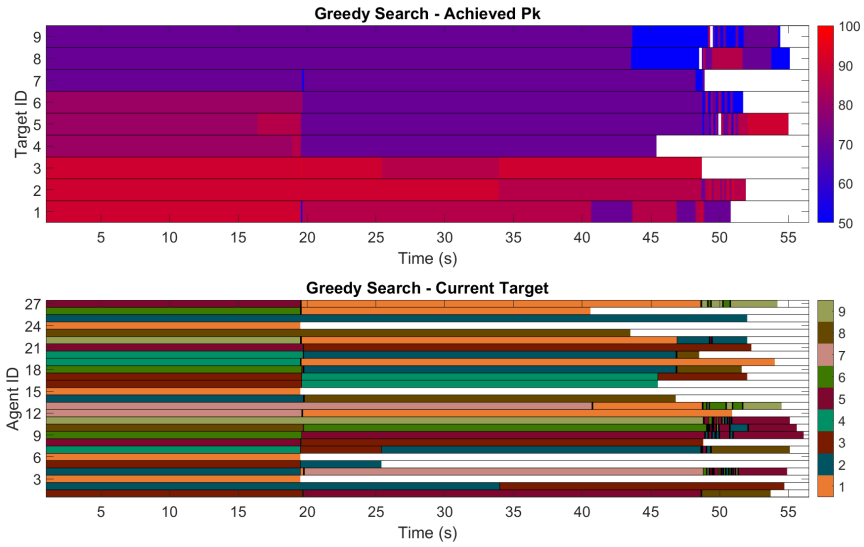

Figure 3.5: Example simulation using ST cost function with four agents attrited. (Top) Target $Pk_{\Sigma,j}$ vs time. (Bottom) Agent assignment vs time.

The ST and ET cost functions essentially use $Pk_{\text{des,j}}$ as a measure of target priority, and thus pursue targets with high $Pk_{\text{des,j}}$ first. In contrast, the Completion cost function seeks to satisfy the desired Pk's on as many targets as possible and does not consider $Pk_{\text{des,j}}$ as a measure of target value. It thus exhibits noticeably different behavior. Figure 3.6 shows an example engagement scenario identical to that shown in Figure 3.4, except using the Completion cost function. In this case, agents select targets with low desired Pk first as these are most easily satisfied, and then continue on to targets with higher $Pk_{\text{des,j}}$. The result is that high $Pk_{\text{des,j}}$ targets may be completely avoided by all agents, as is the case for target 2 in Figure 3.6. These results reflect the desired selection behavior of the Completion cost function, namely that agents would rather satisfy the desired Pk on a small number of targets than achieve low total Pk (which is less than the desired) on a large number of targets. This performance will be further highlighted in the Monte Carlo results. Note that target 2 is not destroyed at $t = 22s$ but rather an agent is attrited and an agent switches from target 2 to target 1 as 2 had the lowest achieved Pk. Since there are insufficient agents to destroy all targets in this scenario, target 2 survives the engagement.

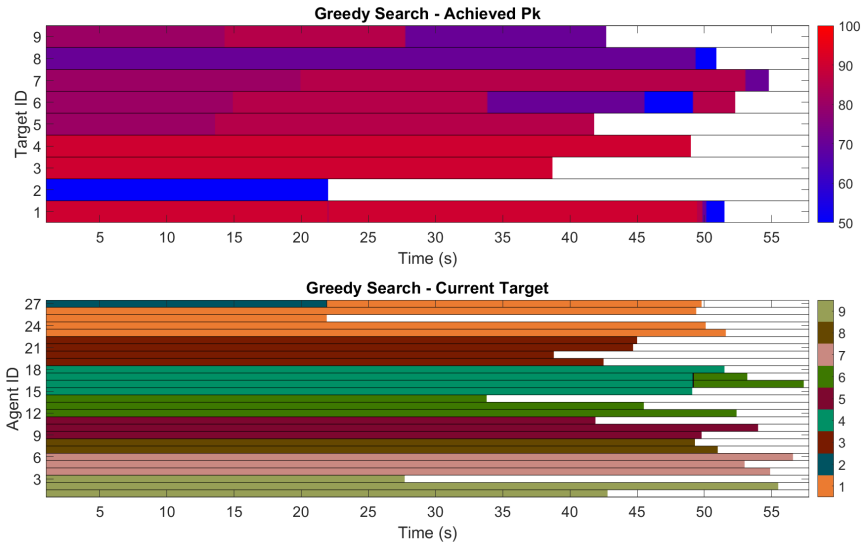

Figure 3.6: Example simulation using completion cost function. (Top) Target $Pk_{\Sigma,j}$ vs time. (Bottom) Agent assignment vs time.

3.4.1.2 Heterogeneous Case

A final example examines the case of heterogeneous agents. Consider 12 agents evenly divided into four different types and 9 targets evenly divided among three different types. Weapon effectiveness values for each agent and desired Pk's for each set of targets are the same as those given in Table 3.1, and the ST cost function is used. Two simulations are conducted in this configuration: one using the greedy search algorithm, the other using simulated annealing. Greedy search results are shown in Figure 3.7, while simulated annealing results are shown in Figure 3.8. The final achieved Pk's for each engagement are shown in Table 3.2, along with the globally optimal assignment. Note that in this particular case the globally optimal assignment can be determined by hand since all weapons are equally effective against high value targets. Specifically, the optimal assignment is given by all agents of Type A targeting medium value targets, all agents of Type B targeting low value targets, and agents of Type C and D targeting high value targets.

Several interesting features are demonstrated in Figures 3.7 and 3.8 as well as Table 3.2. First note that in the greedy search case, the solution converges to a local minimum almost immediately and remains there until the first target is killed. In contrast, the simulated annealing case shows significant switching early in the engagement, finally settling on a solution around 5 sec. The result, as shown in Table 3.2, is that simulated annealing is able to get much closer to the optimal solution and achieve higher Pk values than the greedy search. However, simulated annealing is not able to achieve the global optimum because the target set available to each agent is reduced by dynamic constraints. Overall, this example highlights the ability of a stochastic optimizer to significantly outperform greedy approaches when substantial heterogeneity is present.

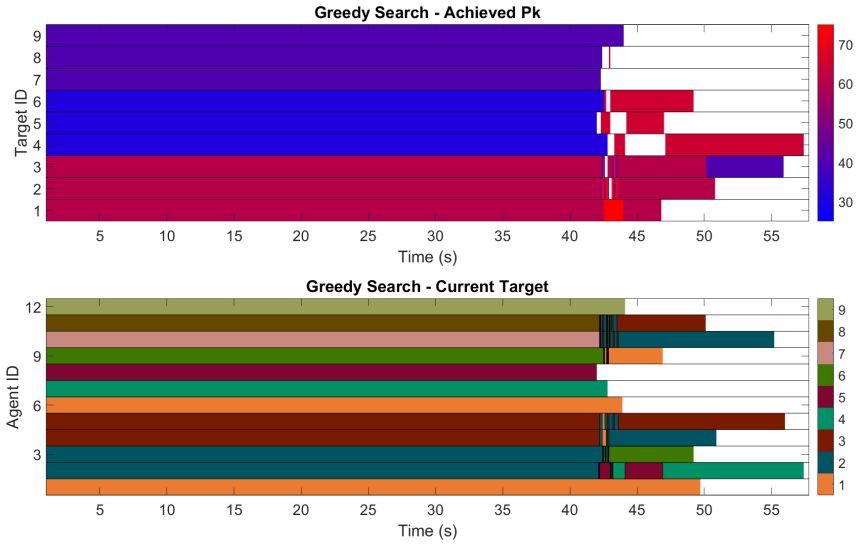

Figure 3.7: Example simulation with heterogeneous agents using ST cost function and greedy search. (Top) Target $Pk_{\Sigma,j}$ vs time. (Bottom) Agent assignment vs time.

Table 3.2: Optimal Pk_{Σ}'s compared to each algorithm

	High Tier $Pk_{des} = 90\%$	Med. Tier $Pk_{des} = 80\%$	Low Tier $Pk_{des} = 70\%$
Optimal	64%	69%	69%
Greedy Search	64%	31%	44%
Sim. Anneal.	68.8%	46%	69%

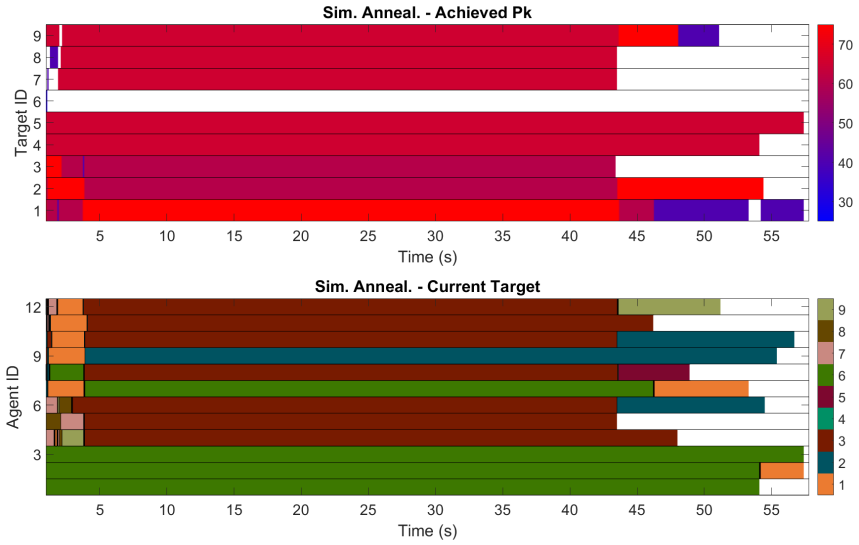

Figure 3.8: Example simulation with heterogeneous agents using ST cost function and simulated annealing. (Top) Target $Pk_{\Sigma,j}$ vs time. (Bottom) Agent assignment vs time.

3.4.2 Monte Carlo Simulation Results

Because each engagement scenario includes probabilistic elements (attrition, random initial location, etc), the behavior induced by the proposed cost functions is best analyzed through the use of Monte Carlo simulation. To this end a variety of Monte Carlo simulations are described using each cost function at varying weapon target ratios.

In the first scenario, nine targets are divided into three groups according to desired Pk: three with $Pk_{\text{des,j}} = 0.90$, three with $Pk_{\text{des,j}} = 0.80$, and three with $Pk_{\text{des,j}} = 0.70$. Weapons are homogeneous with an effectiveness of $Pk_{i,j} = 0.50$ for all weapon-target pairs, but of course varying $Pa_{i,j}$'s introduce some heterogeneity. In each example, the

number of agents is varied to demonstrate behavior of the cost function for several weapon target ratios. For each weapon target ratio, 50 trials are run and the $Pk_{\Sigma,j}$ on each target is recorded after 10 sec. This value is then averaged over each group of targets for the 50 trials. Note that the expected achieved Pk after 10 sec is used as a metric of success for each scenario since, by this point in the scenario, the solution has usually converged but no targets have yet been destroyed. While the number of targets destroyed in each example could potentially have been used as a metric of success, this would have required many more simulation trials to produce a statistically significant result given the relatively low number of targets of each type.

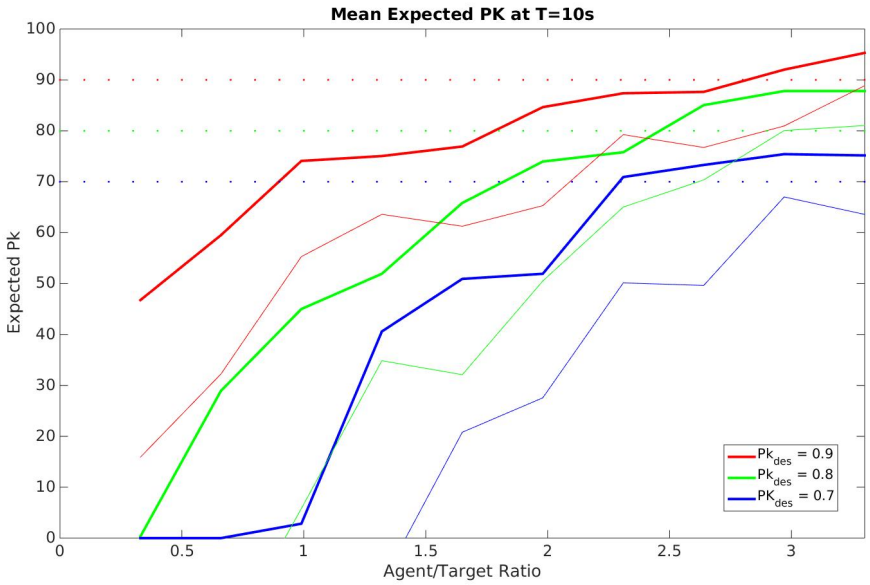

Figure 3.9: Expected Pk vs agent/target ratio using ST cost function ($\omega = 1$) and greedy search. Heavy line denotes mean value, lighter line denotes 2-standard-deviation lower bound.

Figure 3.9 shows Monte Carlo results using the ST cost function with $\omega = 1$. In Figure 3.9, the average result from each ratio is plotted with a heavy line and the 2-standard-deviation lower bound is plotted with the light line. Note that in these cases, at low weapon target ratios agents select high $Pk_{\mathrm{des},j}$ targets first since desired Pk is used as a measure

of priority in the ST cost function. As weapon target ratio grows, agents begin to select targets with lower $Pk_{\text{des},j}$ since this results in the maximum marginal return. As a result, the weapon set does not satisfy all high $Pk_{\text{des},j}$ targets (on average) until a weapon target ratio of about 2.8. In contrast, Figure 3.10 shows the same Monte Carlo results using the ST cost except with $\omega = 2.5$. Note that, unlike in the $\omega = 1$ case, the high $Pk_{\text{des},j}$ targets nearly reach their desired Pk value near a weapon target ratio of 1 (although it is not fully met until a ratio of 2.3). As mentioned in the discussion of the ST cost, increasing ω incentives agents to prioritize high $Pk_{\text{des},j}$ targets over lower $Pk_{\text{des},j}$ targets, a trend that is evident in comparison of Figures 3.9 and 3.10. Thus ω can be viewed as a tuning parameter that allows an operator to specify whether agents should prioritize meeting the desired Pk's on high-value targets first, or meeting them on all targets collectively.

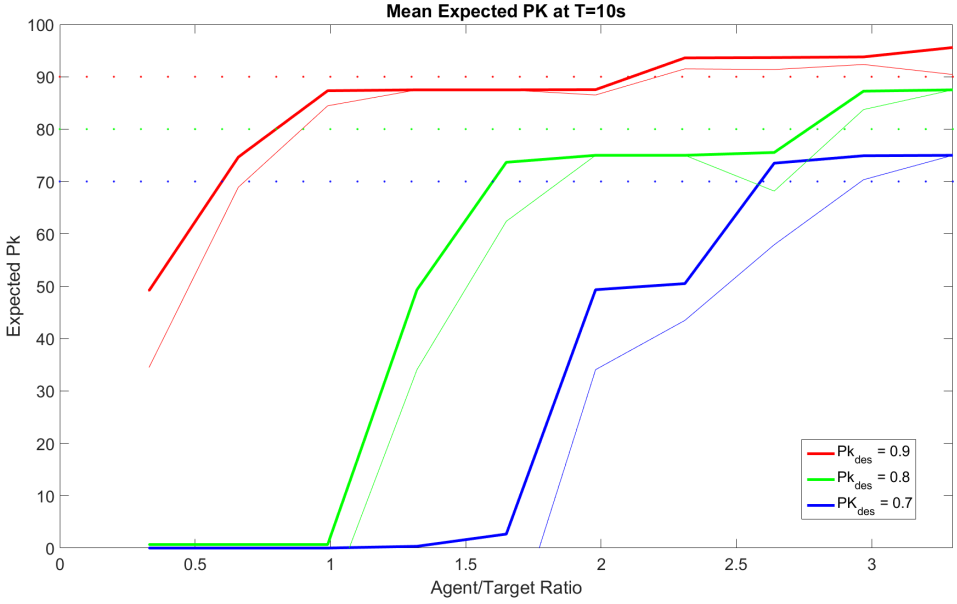

Figure 3.10: Expected Pk vs agent/target ratio using ST cost function ($\omega = 2.5$) and greedy search. Heavy line denotes mean value, lighter line denotes 2-standard-deviation lower bound.

Figure 3.11 shows the same scenario as in Figure 3.9, except using the ET cost function.

In this case targets are arranged in tiers by $Pk_{\mathrm{des},j}$. In contrast to the ST results, the ET cost function prevents any lower $Pk_{\mathrm{des},j}$ targets from being engaged until all higher $Pk_{\mathrm{des},j}$ targets have been satisfied. This results in all higher $Pk_{\mathrm{des},j}$ targets being satisfied (on average) by a weapon target ratio of about 1.2, which is much lower than in the ST case. Thus, in cases where $Pk_{\mathrm{des},j}$ really does reflect target priority, the ET cost function will outperform the traditional WTA and ST cost functions at lower weapon target ratios.

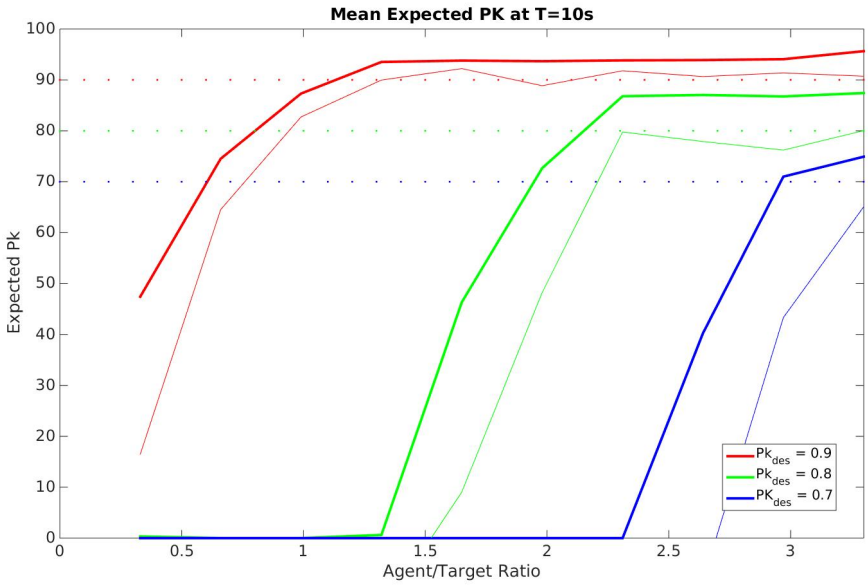

Figure 3.11: Expected Pk vs agent/target ratio using ET cost function and greedy search. Heavy line denotes mean value, lighter line denotes 2-standard-deviation lower bound.

In many practical scenarios, target priority may not be a function of $Pk_{\mathrm{des},j}$ but rather other characteristics such as target location. Consider a case in which three geographical regions are defined as shown in Figure 3.12. One region is defined as a high priority region, one as medium priority, and one as low priority. These regions may for instance reflect areas of different strategic value. One target of each desired Pk level is placed in each area. The ET cost function is then exercised on this example, with tiers given by geographic area rather than $Pk_{\mathrm{des},j}$. The results are shown in Figure 3.13. Here, the "flat line" behavior is still observed similar to Figure 3.11, with targets of lower tiers being neglected until there

is a sufficient weapon target ratio to completely satisfy targets of all higher tiers.

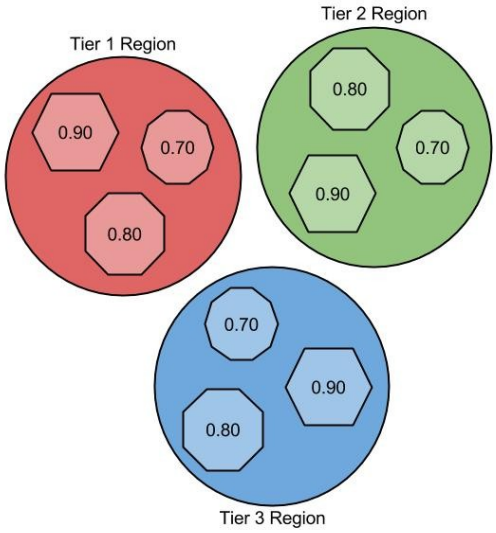

Figure 3.12: Notional target tiering as a function of geographic location. Each tier has one target of each of three desired Pk's.

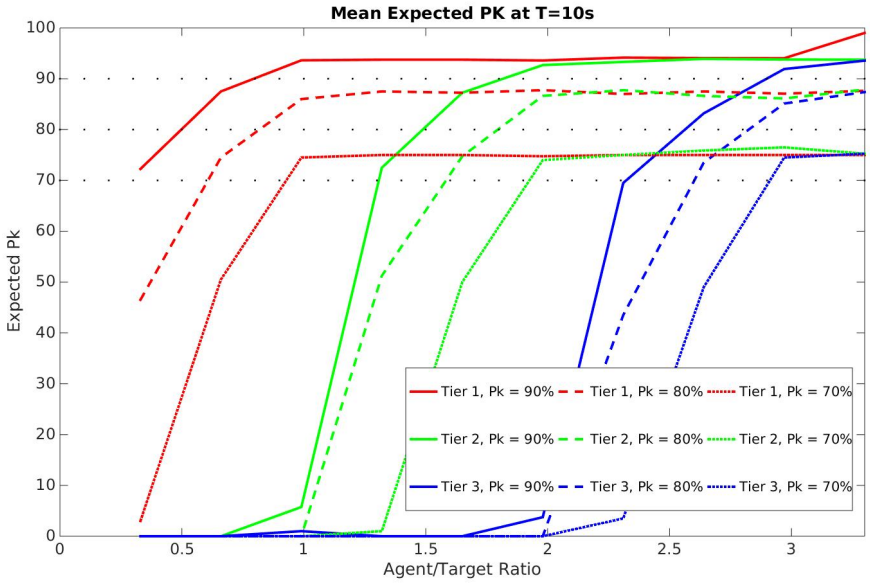

Figure 3.13: Expected Pk vs agent/target ratio using ET cost function and greedy search for Monte Carlo example with geographic target tiering.

Another Monte Carlo example highlights performance of the Completion cost function

at various weapon target ratios. This example uses the same scenario as that used in Figures 3.9 and 3.11. In Figure 3.14, the y-axis is altered to present the number of agents targeting a given class of targets on average, and dashed lines represent the number necessary to achieve the desired Pk. Figure 3.14 clearly shows the expected behavior that targets which are easily satisfied are engaged first (at low weapon target ratios), followed by targets which are more difficult to kill. Note that this behavior is strikingly different from the previous examples but is achieved just by changing the cost function within the same optimization framework.

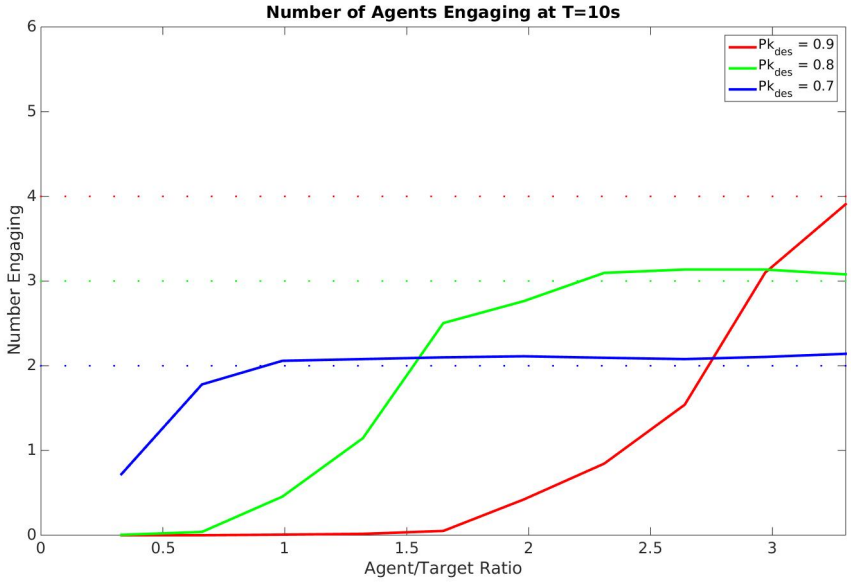

Figure 3.14: Expected Pk vs agent/target ratio using Completion cost function and greedy search.

As a final step in the analysis of the Monte Carlo results, the data from Figures 3.9, 3.11, and 3.14 were processed to determine the number of targets whose $Pk_{\mathrm{des},j}$ had been met. Furthermore, an additional Monte Carlo simulation was performed using the traditional WTA cost given in equation (3.4). This simulation, performed for comparison purposes, uses the same scenario as the above Monte Carlo runs (nine targets of three different desired Pk's).

The results of this comparison are shown in Figures 3.15 - 3.18. In each of these figures, a "completed" target means one whose desired Pk is met. Figure 3.15 shows the number of total completed targets, while Figures 3.16 - 3.18 show completed targets of high, medium, and low desired Pk respectively. Several interesting features are apparent from these figures. First, note that the results for the ST and traditional WTA cost functions are essentially the same, since these cost functions induce identical agent behavior in homogeneous engagement scenarios. Second, in terms of total targets completed Figure 3.15 shows that the Completion cost function satisfies the most targets at any weapon target ratio due to its emphasis on target completion during marginal cost computation. However, comparing Figures 3.16 and 3.18, it is apparent that the ET cost function prioritizes high desired Pk targets, while the Completion cost function leads agents to prioritize low desired Pk targets which are easier to satisfy. The key point evident in these figures is that, in terms of total targets satisfied, both the ET and Completion cost functions outperform the traditional WTA cost at any weapon target ratio below about 3. Furthermore, Figures 3.16 and 3.17 show that the ET cost satisfies more high and medium $Pk_{\mathrm{des},j}$ targets than the traditional WTA cost, at the expense of worse performance against low $Pk_{\mathrm{des},j}$ targets. In cases where $Pk_{\mathrm{des},j}$ is representative of target priority, these results show that the ET cost provides much better target selection behavior especially when the weapon target ratio is relatively low.

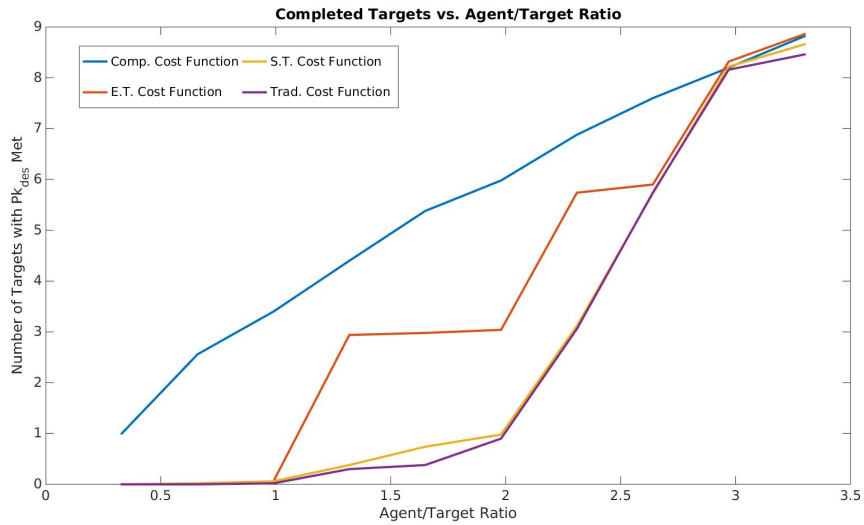

Figure 3.15: Total number of satisfied targets vs agent/target ratio.

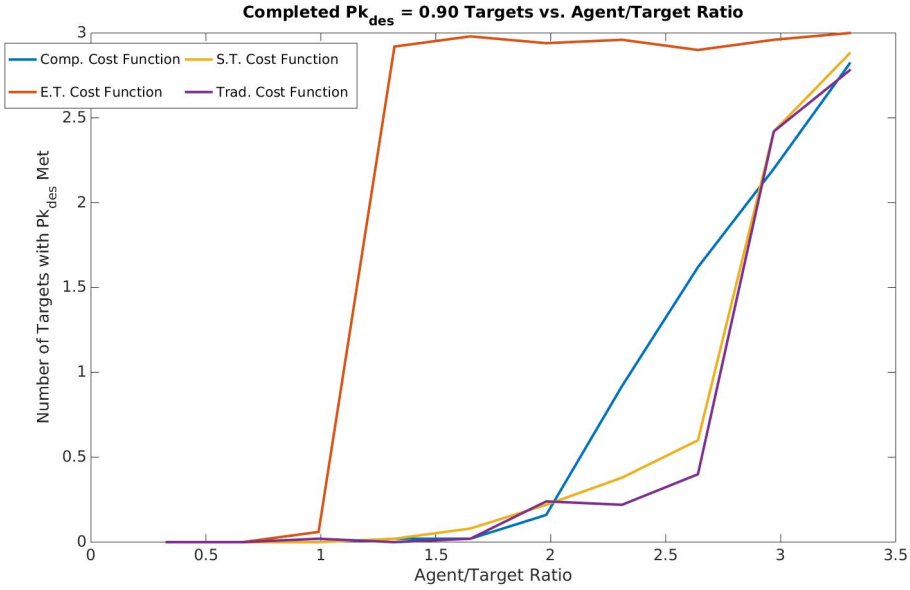

Figure 3.16: Number of satisfied $Pk_{\mathrm{des},j} = 0.90$ targets vs agent/target ratio.

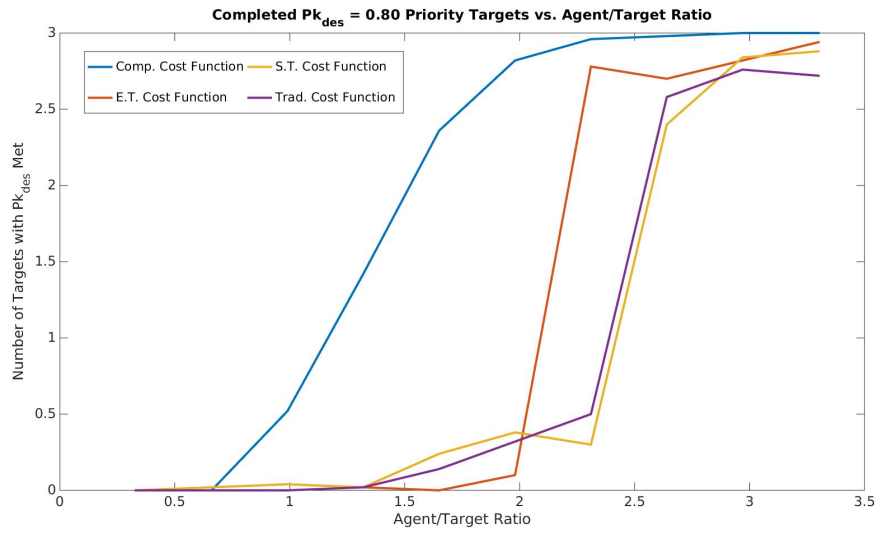

Figure 3.17: Number of satisfied $Pk_{\mathrm{des},j} = 0.80$ targets vs agent/target ratio.

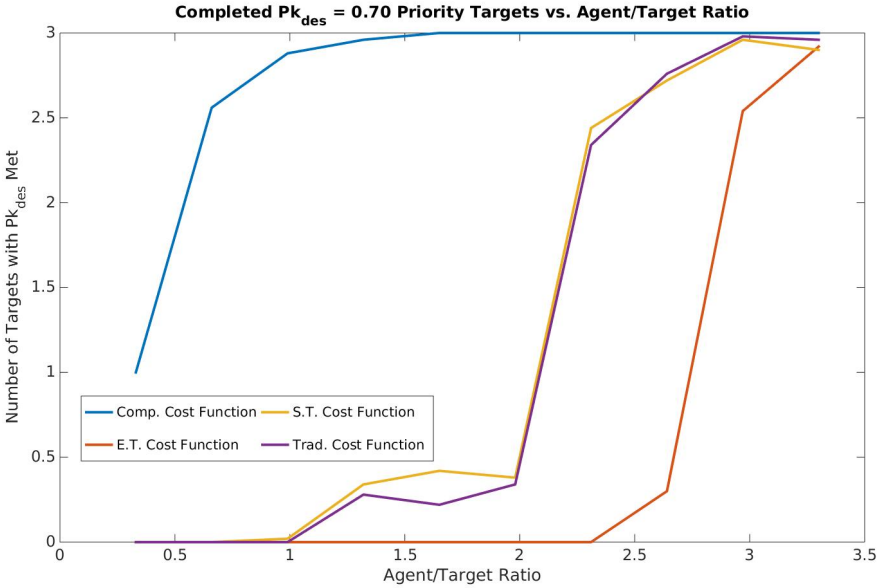

Figure 3.18: Number of satisfied $Pk_{\mathrm{des},j} = 0.70$ targets vs agent/target ratio.

CHAPTER 4

SEQUENCED AND SIMULTANEOUS ARRIVAL SOLUTION ALGORITHMS

This chapter extends the cost functions presented in Ch. 3 to incorporate additional tactical considerations, in particular the arrival times of agents. Recall from Ch. 2 that agents are modeled as point masses with direct control over their descent rate, heading and lateral speed v_h in the $\vec{I}_I - \vec{J}_I$ plane. The agents can vary their lateral speed within the range of $v_{h,\text{nom}} \pm \Delta V$ where $v_{h,\text{nom}}$ is a nominal speed and $2\Delta V$ is the width of the achievable speed range. Because agents are assumed to fly straight-line paths to their selected targets, this restricts the achievable arrival times for an agent to within the following bounds:

$$\begin{aligned} lb_i &= \frac{\sqrt{\left(x_{q_i} - x_i\right)^2 + \left(y_{q_i} - y_i\right)^2}}{v_{h,\text{nom}} + \Delta V} \\ ub_i &= \frac{\sqrt{\left(x_{q_i} - x_i\right)^2 + \left(y_{q_i} - y_i\right)^2}}{v_{h,\text{nom}} - \Delta V} \end{aligned} \tag{4.1}$$

where (x_{qi}, y_{qi}) denote the position coordinates of the target currently pursued by agent i. Note that this straight-line path may be used without loss of generality since a curved path to the target under certain speed constraints has an analogous set of speed constraints for a straight-line path.

4.1 Optimization Algorithms

This section describes a new optimization algorithm employed by each agent during targeting decisions, while the next section describes the novel cost function designed to balance timing and Pk objectives. The previous chapter introduced greedy search and simulated annealing optimization algorithms. However, in this chapter several objectives must be balanced at once, meaning that the straightforward schemes presented in Ch. 3 may lead to reduced performance. When considering timing-based objectives, the initial locations of agents introduces significant heterogeneity and limits the efficacy of greedy

search. For this reason, this chapter introduces a branch and bound algorithm that produces better results for the multi-objective problem, at the expense of additional computational burden and communications requirements. Section 4.3 of this chapter compares the performance of this new algorithm with that of greedy search, quantifying the performance improvement to be gained with the additional computation of branch and bound.

4.1.1 Branch and Bound

Branch and bound is a state-space search algorithm for combinatorial optimization problems which relies on a systematic, targeted enumeration scheme. While all branch and bound schemes use the same overall enumeration strategy, the algorithm implementation is problem-dependent. Several authors have proposed branch and bound implementations for the classical WTA problem [60, 61]. However, due to the addition of sequential timing or simultaneous arrival constraints in the present work, a new branch and bound scheme must be developed.

Before this branch and bound algorithm can be described, two agent behaviors must be defined. Both of these behaviors are implemented as functions that take an assignment problem as an argument and return modified copies. The first behavior, referred to as "Solve", takes a partial assignment as input and applies greedy search on all agents that are unassigned. For one agent at a time, the "Solve" scheme assigns that agent to the target that produces the largest marginal return. The result of this function is a fully-assigned version of the original partial assignment, referred to as a feasible solution, and the cost function evaluated over this copy will be greater than or equal to the optimal cost given the partial assignment that was originally used. This means that applying Solve to a partially assigned problem and then calculating the cost function gives an upper bound on the optimal cost by establishing that a solution at least that good exists.

The second behavior, referred to as "Relax", takes the partial assignment and modifies

it in such a way that applying Solve will result in finding the globally optimal solution to the modified version. The most important property of Relax, as expressed in Eq. 4.2, is that given a partial assignment, Q_P, applying Relax on Q_P yields a problem that can be optimally solved by Solve and for which the optimal cost value is no greater than that exhibited by the optimal solution of the original problem starting from Q_P. The relaxation used here is based on the method presented by Ahuja [60]. It should be noted that the assignment reached by applying Relax and then Solve does not necessarily correspond with the optimal assignment, but rather it provides a lower bound on the optimal cost. By making two copies of a partial assignment, applying Solve to one and Relax/Solve to the other, the upper and lower bounds respectively on the optimal cost given that partial assignment can be found.

$$\begin{bmatrix} Q_R = \text{Relax}\left(Q_P\right) \\ \left\{Q^* | Q_R\right\} = \text{Solve}\left(Q_R\right) \\ C\left(\left\{Q^* | Q_R\right\}\right) \leq C\left(\left\{Q^* | Q_P\right\}\right) \end{bmatrix} \tag{4.2}$$

The scheme used by the Relax algorithm, termed a relaxation, is entirely dependent on the specifics of the problem, but is not necessarily unique for a given problem. The relaxation function used here is designed for the WTA problem with the arrival time-based cost function, which is presented in Section 4.2. In short, the cost is the sum of Pk-based and arrival time-based cost functions, weighted by a user-defined scalar parameter α:

$$C_{\text{COMP}}(Q) = \alpha C_{Pk}(Q) + (1 - \alpha) C_T(Q) \tag{4.3}$$

In this case, the relaxation occurs in two steps: relaxation of effectiveness and relaxation of arrival time. To relax the effectiveness, each unassigned agent's effectiveness against each target is increased to be equal to the maximum effectiveness against that target over

all unassigned agents as shown in Eq. 4.4:

$$Pk_{i,j}^{\text{relax}} = \max_{n \notin Q} Pk_{n,j} \tag{4.4}$$

where $Pk_{i,j}^{\text{relax}}$ denotes the effectiveness assigned to the relaxed agents. This results in the relaxed problem being homogeneous and thus the Solve routine, which performs greedy search, will find the globally optimal solution [60]. Since $Pk_{i,j}^{relax} \geq Pk_{i,j}$ for all unassigned agents, this implies that the Pk-based cost of the relaxed full assignment will be less than or equal to the Pk-based cost of the optimized unrelaxed (actual) full assignment, i.e. $C_{Pk}^{relax} \leq C_{Pk}$.

The above relaxation scheme treats only the Pk portion of the composite cost, but does not address the time-sequencing or simultaneity portion C_T. To relax simultaneity, the allowable deviation of velocity is set to the nominal velocity, $\Delta V = V_{h,nom}$. As can be seen from Eq. 4.5, this extends the arrival time bounds for each agent such that if all agents are relaxed, a set of arrival times can be found that perfectly matches the desired spacing.

$$\begin{aligned} lb_i^{relax} &= \tfrac{lb_i}{2} \\ ub_i^{relax} &= \infty \end{aligned} \tag{4.5}$$

A perfectly timed arrival by all targets gives a time cost of $C_T^{relax} = 0$. The combined effect of these relaxations is that, no matter the value of α, $C^{relax} = \alpha C_{Pk}^{relax} + (1-\alpha)\, C_T^{relax} \leq C$ for a given partial assignment.

With the Relax and Solve functions defined, the branch and bound algorithm description can proceed. The algorithm begins by generating a table of partial assignments for each agent-target pair, totaling $M \times N$ initial assignments. Solve and Relax/Solve provides upper and lower bounds, respectively, on the optimal cost for each partial assignment. These partial assignments form the roots of parallel search trees for the branch and bound algorithm. At this stage, a clean-up step is performed where the algorithm "prunes" any partial assignments for which the lower bound is higher than the

upper bound for another partial assignment. This step reduces the dimension of the search tree and speeds convergence. The full assignment used to create the upper bound for each partial assignment is also saved for reasons that will be explained later.

The bounds calculated in this first step can be used to sort the partial assignments to find the order in which they should be expanded. The ordering selected can affect how long the algorithm takes to converge to the global optimum, and various ordering schemes may be used. In this work, the agents are sorted from lowest to highest upper bound and then, within the partial assignments that have the same upper bound, from lowest to highest lower bound. Figure 4.1 shows the sorted list of initial partial assignments for a simple two agent, two target case. The notation $Q = \{*, 1\}$ denotes a partial assignment in which agent 1 has not selected a target and agent 2 is engaging target 1.

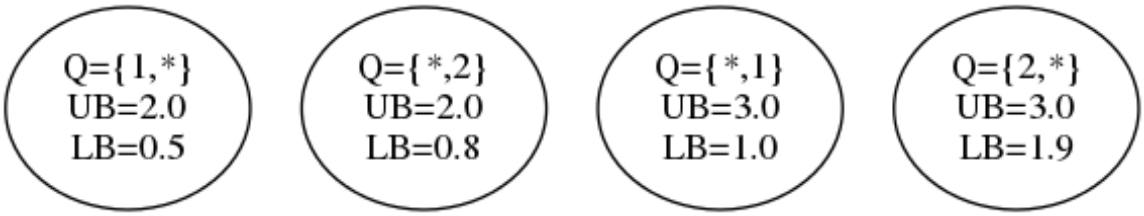

Figure 4.1: Partial assignments, sorted from left to right. Asterisks denote an unassigned agent.

As the algorithm continues to explore and prune the search tree as described below, eventually an agent's choice of targets will be reduced to a single target representing the now-known optimal assignment for that agent. Propagating this information when it is found serves two purposes. First, it can tighten the bounds on unconverged partial assignments. Second, it reduces the dimension of the problem so that the Relax and Solve functions can execute more quickly. For every nonpruned, unconverged partial assignment the solved agents are assigned to their optimal targets and the bounds are updated. If this causes additional agents to prune down to their optimal assignments then this process can be repeated.

Once the initial set of pair-wise partial assignments are enumerated, the main loop of the algorithm begins. In the main loop, agents select a partial assignment, referred to as a node, from one of the search trees rooted in the table to expand. To select a node to investigate, the following tests are applied at each level of the tree in a recursive manner to each of the $M \times N$ trees until a viable node is found. If the test on a node returns that it is not a valid candidate for expansion then the test moves on to the next untested node on that level. If a level is exhausted without finding a valid candidate then the test on the parent node returns no valid candidate and moves on to the next node at the higher level. The nature of the tests are such that if a node is not a valid candidate for expansion, none of its child nodes are either. Additionally, if a node has child nodes that are valid candidates, the algorithm would prefer to expand the children first, only further expanding a node with children if that node is a valid candidate but none of its existing child nodes are valid candidates for expansion. Depth-first search algorithms such as this run the risk of poor run times [62] but, in this case, sorting the nodes for expansion focuses the algorithm on the most promising branches. To be a valid candidate for expansion, a node must not be pruned, must have differing lower and upper bounds on the cost, and the agent performing the expansion must be unassigned with a non-empty domain of possible targets.

Once an agent selects a node to expand, it expands the tree by creating a new child node for each target in the agent's domain. As part of the expansion, the upper and lower bounds of the expanded node are calculated with the generating agent assigned to that target. Agent 1's expansion for the above example is shown in Fig 4.2. It should be noted that in this example there is one optimal child assignment (to the left) given the parent partial assignment and that the other child node is pruned (to the right). Since all assignments where agent 1 is assigned to target 2 are shown to be suboptimal, if there were more nodes to be expanded then target 2 could be removed from agent 1's domain as it is known that the optimal assignment doesn't contain target 2 being assigned to agent 1.

In the next step, the results of the expansion are propagated up the search tree. Once a

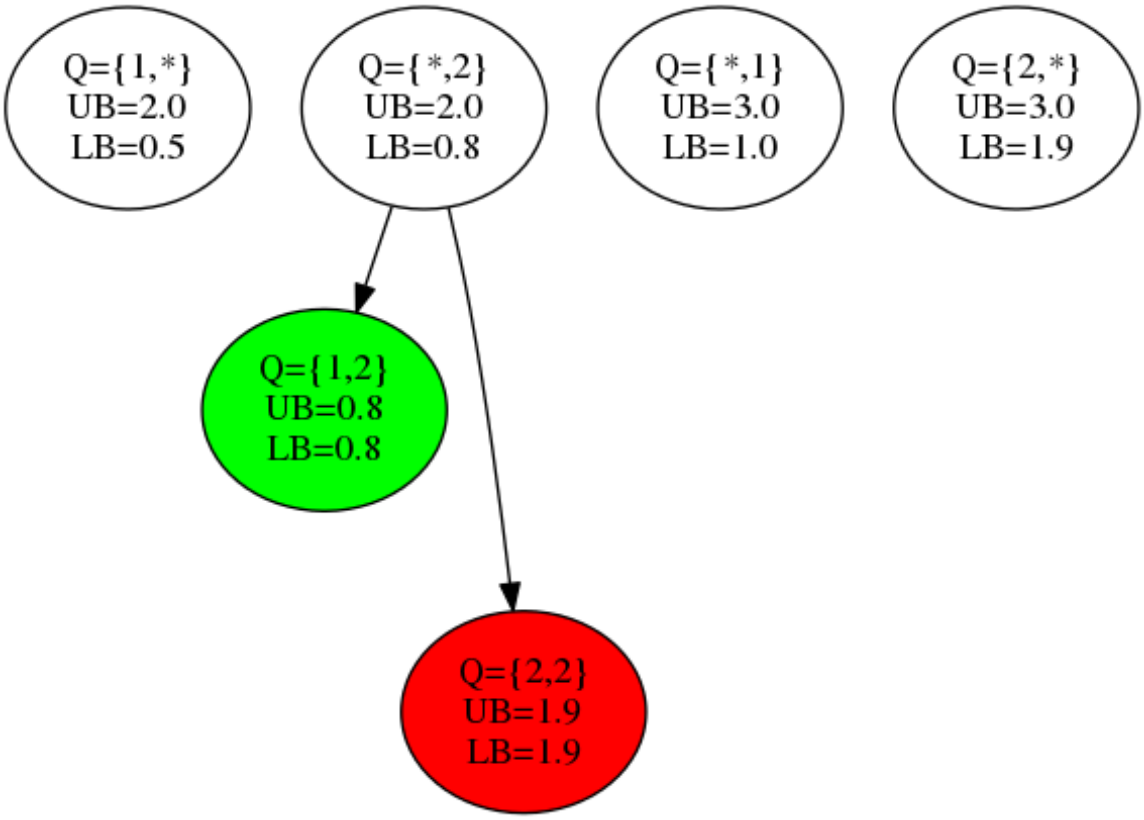

Figure 4.2: Agent 1 expands leftmost node in which it is unassigned.

partial assignment has been expanded, its upper bound is equal to the lowest upper bound amongst its children. If the expansion did not find a better unrelaxed solution than the one that generated the parent's upper bound, then this unrelaxed (incumbent) solution is kept as a known "best" solution for this partial assignment. Due to the nature of the relaxation, the child assignments of a given partial assignment cannot be relaxed to the same degree as the parent. This is why it is important to create a child for each target in the agent's domain, even though its lower bound might be higher than its parent's. The lowest lower bound among the children is propagated up the tree to the parent and is a valid lower bound because the expansion was exhaustive for that agent. The propagation is then performed under these same rules recursively up the tree to each parent. Additionally, the upper bound is passed to the N root nodes corresponding to assignments in the incumbent solution. This progressive tightening of the bounds is what eventually leads to pruning of suboptimal

assignments.

Continuing the above example case, in Figure 4.3 the results of the expansion are propagated. The leftmost two root assignments converge to the optimal cost associated with the optimal assignment of $Q = \{1, 2\}$. The rightmost root assignment now has a reduced upper bound due to propagation up the tree, but the lower bound remains unchanged. As both of the rightmost two assignments have higher lower bounds than another upper bound, they are pruned and the algorithm exits having found the optimal assignment.

The main loop runs until the optimal assignment for each agent is found or until a prespecified end condition is met. With algorithms of this type there is no guaranteed convergence rate, so as a practical matter it is advisable to use an early-termination criterion to restrict run time to some desired limit. In this work, execution is halted after a maximum number of node expansions and the best solution found at that point is applied. Because at any point during execution it is always possible to produce a "best solution found so far", the above branch and bound scheme is an anytime algorithm [63].

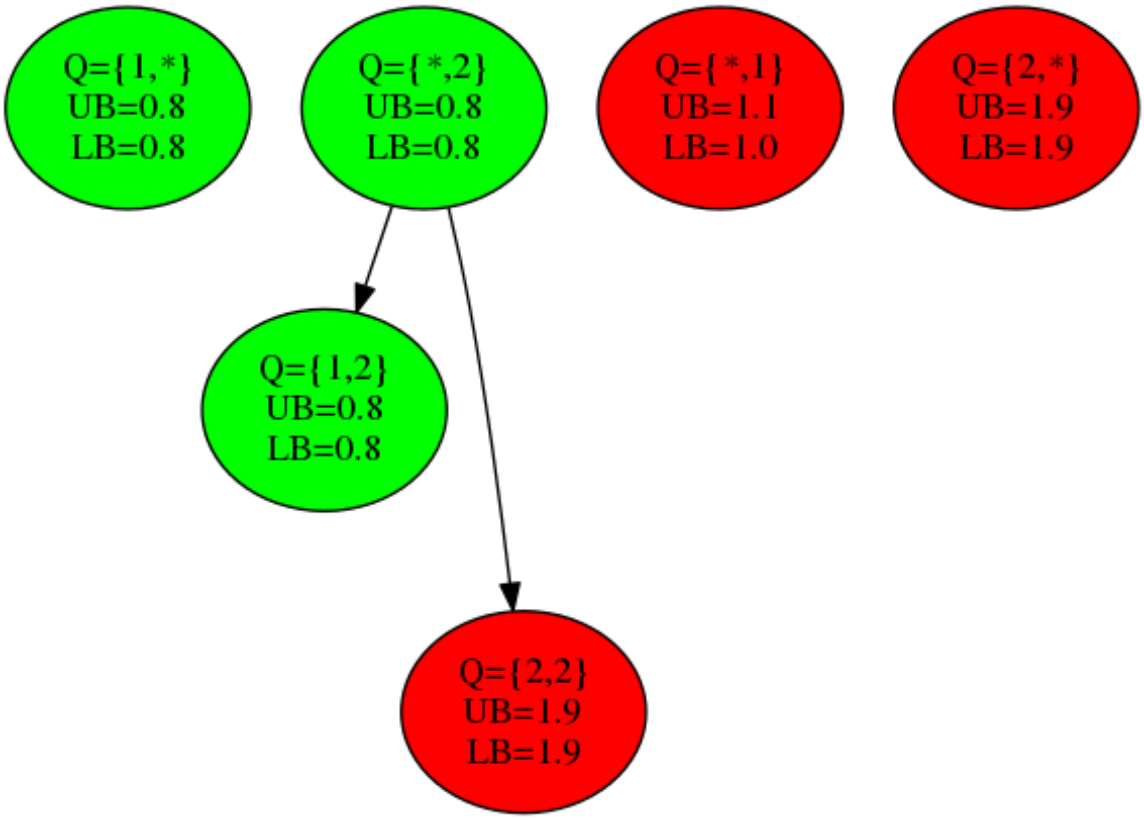

Figure 4.3: Expansion results are propagated upwards to the root assignments.

Assuming that the agents are capable of communicating reliably, the above branch and bound search process can be distributed by having each agent select a partial assignment for expansion from a local copy of the search tree, performing the expansion step and then sharing the results with the other agents. The propagation and pruning steps can then be performed on the local copies independently so as to minimize the amount of information that needs to be shared between agents. A high-level overview of a decision round using this branch and bound scheme is shown for agent i in Figure 4.4. In this context, a decision round is defined as each agent in turn selecting two nodes for expansion: one node which is selected for its potential to find a better solution than the incumbent, and the other for its potential to prune unpromising branches of the search tree. After each agent selects two nodes it broadcasts updates to the search tree and the next agent selects its nodes.

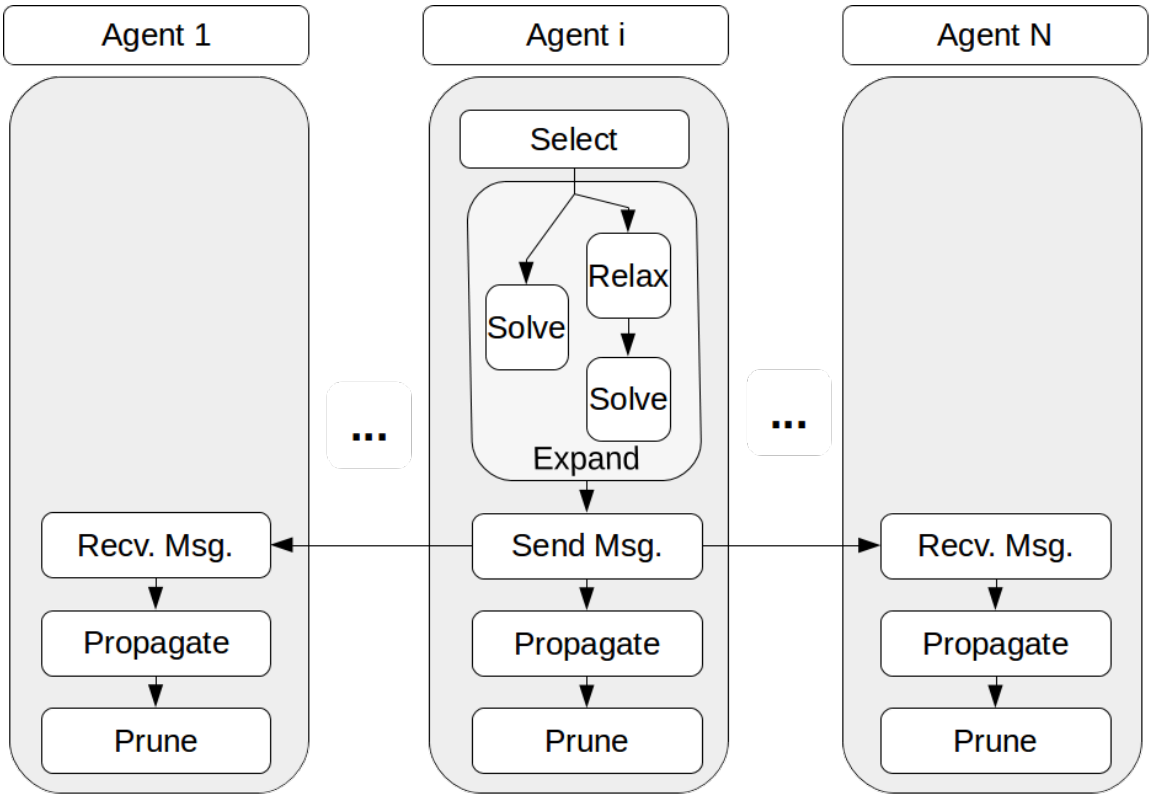

Figure 4.4: Flowchart summarizing an iteration of the algorithm for a single agent.

Branch and bound is fundamentally an enumeration method and its completeness comes from either proving that, for every region in a search space, an optimal solution cannot exist in that region, or by fully exploring the region. As can be seen in Sec. 4.3, an optimal solution will frequently be found long before it is proven to be optimal by the algorithm (through pruning of all other solutions). Since it isn't generally possible to predict when the optimal solution will be found, for an online implementation of this algorithm each agent should assign itself to the target corresponding to its assignment in the incumbent solution, or best solution found so far. This ensures that even if the assignment isn't proven to be optimal before the termination criteria is reached, the agents will still pursue some assignment while attempting to find a better one.

4.1.2 Computational Complexity

The general WTA problem with N agents and M targets is a combinatorics problem, so a full state enumeration requires evaluating $\mathcal{O}\left(M^N\right)$ solutions. This of course guarantees an optimal solution. Greedy search, as shown in [64], can achieve very good results in some cases by evaluating $\mathcal{O}\left(M \times N\right)$ solutions or, when computation is distributed across the agents, $\mathcal{O}\left(M\right)$ solutions onboard each agent. Branch and bound is an enumeration-type algorithm, so its complexity class is $\mathcal{O}\left(M^N\right)$; however, termination criteria and heuristics allow for its practical performance to be much better. For each decision round, each agent will perform two iterations of greedy search for each target in its domain. This gives $\mathcal{O}\left(M^2 \times N^2\right)$ evaluations or $\mathcal{O}\left(M^2 \times N\right)$ per agent if distributed as described above.

Note that the above complexity values are not measured in time but in the number of solutions to be evaluated. Evaluating the Pk-based cost function can be done in $\mathcal{O}\left(N\right)$ or in $\mathcal{O}\left(N + M\right)$ time, depending on the algorithm used. The arrival time cost function is more complicated as it requires optimizing the arrival times of M subsets of agents (i.e., all the agents that choose a given target). If for a particular target j, the subset N_j has been seen before, the arrival times for the agents can simply be looked up from a hash table. In general, however, subsets without known optimal timings will require time-sequencing optimization. A proposed algorithm to accomplish this is discussed below, yielding $\mathcal{O}\left(N_j^2\right)$ runtime complexity in the worst case.

4.2 Cost Functions

The approach used here seeks to tailor the optimization cost function to induce specific desired behaviors from the set of agents, independent of the solution or assignment algorithm implemented. Multiple Pk-based cost functions are defined in Ch. 3, with each eliciting different behaviors from the agents. Of these, the Sufficiency Threshold (C_{ST}) cost function is used for all experiments presented in the results section of this chapter.

The timing cost function (C_T) is integrated, along with existing Pk-based cost functions, into the composite cost function (C_{COMP}) detailed at the end of this section.

4.2.1 Timing Cost Function

It is conceivable that an operator deploying a multi-agent weapon system would want the arrival times of all agents engaging a given target to be separated by certain intervals. Alternatively, the operator may desire simultaneous arrival of all agents at a given target. Note that simultaneity is simply a special case with an arrival interval width of zero seconds, so it is desirable that the same cost function handle both scenarios. Consider a vector $\vec{t}$ of length N_j which consists of the arrival times of all agents engaging target j. Further, there exists a constant linear mapping $h : \mathbb{R}^{N_j} \mapsto \mathbb{R}^{N_j}$ such that the elements of $h(\vec{t})$ are monotonically increasing. The box constraints on arrival times are given by $lb_i \leq t_i \leq ub_i$ where t_i denotes element i of $\vec{t}$. These constraints are due to the minimum and maximum target closing speeds ($v_{h,nom} \pm \Delta V$) for each agent as described earlier. For convenience the notation $\vec{\tau} = h(\vec{t})$ is adopted. This allows a neighborhood of the state space to be defined,

$$\max(\tau_{i-1}, h\left(lb\right)_i) \leq \tau_i \leq \min(\tau_{i+1}, h\left(ub\right)_i) \tag{4.6}$$

For a given set of arrival times and a desired interval, S, the arrival time cost can be defined as follows,

$$C_T = \sum_{i=1}^{N_j - 1} \left(\tau_{i+1} - \tau_i - S\right)^2 \tag{4.7}$$

If the agents can vary their arrival times without bounds, then there are infinitely many optimal solutions. More realistically, however, each agent has minimum and maximum arrival time constraints on each target based on factors such as vehicle dynamics, fuel limitations, path constraints, etc. Under these constraints there could be zero, one or many solutions which yield $C_T = 0$ for a given target and subset of agents. As discussed below,

for a given set of agents pursuing target j, the ordering and arrival times of agents must be optimized using some numerical algorithm. The resulting cost is then used in a composite cost function to evaluate the assignment.

The task of optimizing arrival times for a given set of agents can be broken into two subproblems. The first is to find a possibly unique agent ordering which can yield the optimal set of arrival times. The second step is to find the optimal agent arrival times for that ordering. This is thus the globally optimal time cost for this set of agents.

The natural choice for initial agent ordering is to place them in the order in which they would arrive if each agent traveled at its nominal velocity $v_{h,nom}$. Because the velocities of the agents are assumed to have the same maximum and minimum values for each agent, the minimum and maximum arrival times are also monotonically increasing when the agents are in the nominal order. This leads to the following theorem.

Theorem 4.2.1. *Define* $w_i = ub_i - lb_i$. *If* $w_i \leq w_{i+1} \forall i$ *then the globally optimal timing* $\vec{t}_{\text{opt}}$ *satisfies* $t_{\text{opt,i}} \leq t_{\text{opt,i+1}} \forall i$.

Proof. Assume that for a given set of agents and a given desired arrival time interval the globally optimal solution occurs when agent a arrives at time t_a and agent b at time $t_b < t_a$ even though $t_{a,\text{nom}} < t_{b,\text{nom}}$. Thus, the arrival time ranges for each agent overlap ($t_{a,\max} > t_{b,\min} \implies t_{a,\min} < t_b < t_{a,\max}$, and vice versa). This means that an alternative solution exists at $t_a^* = t_b, t_b^* = t_a$ with the same cost value but with the agents arriving in the nominal ordering. As can be seen in the top half of Figure 4.5, because the bounds on arrival time are computed as the straight line agent-target distance divided by $v_h + \Delta V$ or $v_h - \Delta V$, the nominal ordering will be the same whether agents are sorted by minimum or maximum arrival time, or the midpoint in their arrival time ranges ("mean" time). In contrast, if the bounds are such that one agent's reachable times are a subset of another's as shown in the bottom of Figure 4.5, then the optimal ordering might be based on the minimum arrival time, mean arrival time or (as in this case) maximum arrival time. If this is the case, the analytic solution for computing the optimal arrival times presented later in this section is

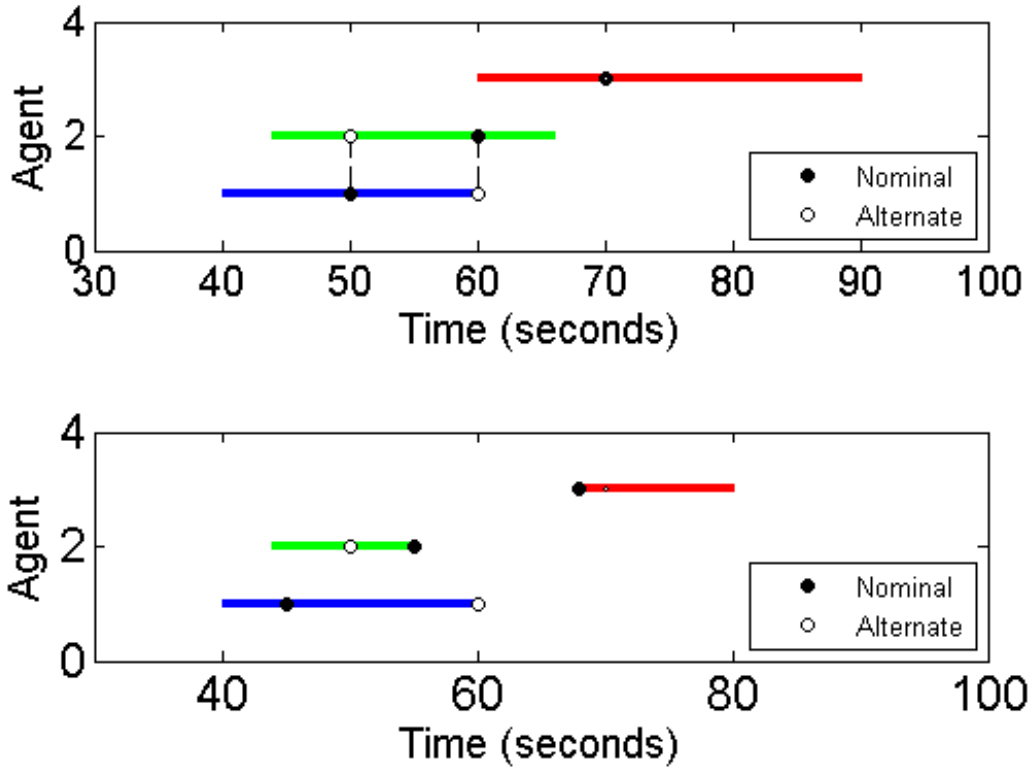

Figure 4.5: Example Arrival Time Intervals and Optimal Orderings.

not applicable. Under the current assumptions this scenario is only possible for relaxed agents under the branch and bound solution scheme. In the case of relaxed agents, $w_i = \infty$ so the arrival times of all unrelaxed agents are subsets of those of all relaxed agents. □

Recall from the previous section that evaluation of the cost function for a partial assignment in branch and bound requires the evaluation of the composite cost function, which includes an optimization of arrival times. The relaxation step complicates the ordering as it requires that all relaxed agents be identical. This means that no speed constraints are imposed on the relaxed agents, meaning they have infinitely large arrival time bounds and no nominal arrival time can be defined. For this reason, the nominal ordering is used for the unrelaxed (or already assigned) agents and then relaxed agents are apportioned into the gaps between pairs of unrelaxed agents in such a way as to yield the

optimal timing given the bounds of the unrelaxed agents. As shown below, greedy search is sufficient for this apportionment task. Each relaxed agent is added to each gap in turn and then the optimal timing is found and the cost evaluated. By assigning each agent to the gap which gives the biggest immediate reduction in C_T, the globally optimal apportionment of relaxed agents is found.

Theorem 4.2.2. *Given a set N_j^U of unrelaxed agents with bounded arrival times engaging target j and a set of N_j^R relaxed agents without arrival time bounds, the apportionment of relaxed agents between unrelaxed agents that yields the optimal arrival time solution can be found using greedy search.*

Proof. The optimization of arrival times for agents engaging the same target is comprised of two parts, finding the optimal arrival ordering and then finding the optimal times given that ordering. As shown above, when all of the agents have bounded arrival times, the optimal ordering is the same as the ordering at the nominal velocity. If there are $N_j^R < N_j$ relaxed agents engaging a target, the optimal ordering is less obvious. The problem can be framed as apportioning the relaxed agents into the gaps between $N_j^U = N_j - N_j^R$ unrelaxed agents.

Define g_i to be the number of relaxed agents between unrelaxed agents i and $i+1$. If the interval between the unrelaxed agents i and $i+1$ is larger than the desired spacing, these relaxed agents enable a reduction in the arrival time intervals closer to the desired spacing. Note that if there are excess agents, they can equivalently be placed after the last unrelaxed agent, or before the first unrelaxed agent, and perfectly spaced out according to the desired interval S. These "excess" relaxed agents can be said to belong to gap $g_{N_j^U}$.

The optimization problem is formally defined by

$$\begin{aligned} \underset{g}{\text{minimize}} \quad & f(g) = \sum_{i=1}^{N_j^U - 1} (g_i + 1) \left(\frac{t_{i+1} - t_i}{g_i + 1} - S \right)^2 \\ \text{subject to} \quad & \sum_{i=1}^{N_j^U} g_i \leq N_j^R \\ & lb_i \leq t_i \leq ub_i, i = 1, \ldots, N_j^U \end{aligned} \tag{4.8}$$

The proof will proceed through induction. To that end, define the apportionment solution with $m \leq N_j^R$ relaxed agents as $g^{(m)}$, and let $g_i^{(m)}$ represent the number of relaxed agents arriving after unrelaxed agent i and before unrelaxed agent $i+1$ in solution $g^{(m)}$. The greedy search algorithm employed here starts with $g_i^0 = 0, i = 0, \ldots, N_j^U$ and iteratively adds one relaxed agent to the gap that results in the lowest cost state until all N_j^R agents have been apportioned. By making the locally optimal choice at each iteration the globally optimal apportionment is reached.

Greedy search will yield the optimal solution to a problem if it can be shown that the problem exhibits what is known as optimal substructure [65]. This means that the optimal solution is constructed from the optimal solutions of all of its subproblems. In this context, this means that given the optimal solution $r^{(m+1)}$ with a set of unrelaxed agents, the optimal subassignment $r^{(m)}$ with the same set of unrelaxed agents and only m relaxed agents is at least as good as any other solution with m relaxed agents.

Define a subassignment $r^{(m)}$ that differs from $r^{(m+1)}$ by the removal of a single relaxed agent in gap A:

$$\begin{aligned} r_i^{(m)} &= r_i^{(m+1)} \qquad \forall i \neq A \\ r_A^{(m)} &= r_A^{(m+1)} - 1 \end{aligned} \tag{4.9}$$

where A is the gap between unrelaxed agents associated with the smallest increase in the cost function if one relaxed agent is removed from the gap. If the problem exhibits optimal substructure, then $r^{(m)}$ is the optimal assignment with m relaxed agents

apportioned. Assuming this isn't the case, let there exist an optimal assignment $g^{(m)}$ that is distinct from $r^{(m)}$.

If the optimal $g^{(m)}$ differs from $r^{(m)}$ only by a single relaxed agent being assigned to a different gap, then note that unassigning that agent cannot increase the cost function less than unassigning it from A (i.e., the step needed to go from $r^{(m+1)}$ to $r^{(m)}$). In this case $r^{(m)}$ can be defined by the following three equalities.

$$\begin{aligned} g_+^{(m)} &= r_+^{(m)} + 1 \\ g_-^{(m)} &= r_-^{(m)} - 1 \\ g_i^{(m)} &= r_i^{(m)} \qquad \text{else} \end{aligned} \tag{4.10}$$

Note that removing a relaxed agent from $r_-^{(m)}$ and assigning it to $r_+^{(m)}$ will transform it into the solution $g^{(m)}$ and, by the assumption of suboptimality, reduce the cost function. Decrementing $r_-^{(m)}$ cannot increase the cost function by more than decrementing A did, and likewise incrementing $r_+^{(m)}$ cannot decrease the cost function by more than $f\left(r^{(m)}\right) - f\left(r^{(m+1)}\right)$. Thus $r^{(m)}$ cannot differ from $g^{(m)}$ by the assignment of only a single relaxed agent if $r^{(m)}$ is suboptimal. At best, reassigning a single relaxed agent from $r^{(m)}$ can only find a solution with an equal cost.

By extension, reassigning multiple agents from one or more gaps to others can at best find an equal cost solution. This is because reassigning multiple agents is the same as sequentially reassinging one, so $r^{(m)}$ must be optimal. Since the cost cannot be further lowered, the optimal subassignment of $r^{(m+1)}$ optimally solves the subproblem of m relaxed agents. This means that the problem exhibits optimal substructure and can be solved with greedy search.

□

With the optimal ordering established, it remains to find the optimal arrival times given that ordering. Since the C_T cost function is simply the sum of squares, the arrival

time optimization once the ordering is established is a linear least squares problem with maximum and minimum constraints on each variable. As such, the problem can be solved using a gradient-based iterative solver such as an interior point scheme. Unfortunately, the convergence properties of these iterative schemes depend on the proximity of the initial guess to the optimal solution and whether the initial guess is valid with respect to the constraints. As an alternative, an analytic solution scheme is developed that has the advantage of requiring a known deterministic number of computations to compute the optimal solution. This is particularly beneficial for practical implementation since a solution can be guaranteed in a known amount of time. This analytical solution scheme is detailed below.

The analytical method starts by ordering the agents as described above, using greedy search if the problem is partially relaxed. The earliest-arriving agent is then assigned an arrival time at its lower bound. Then, the second agent is set to arrive as close as possible to the desired spacing, S, from the first. If the second agent is at its lower bound, then the first agent's arrival time will be increased so as to minimize C_T. This process continues, in which agent i is added to the list and its arrival time is selected to achieve a spacing as close as possible to S from agent $i-1$. Then the arrival times of agents $i-1, i-2, ..., 1$ are adjusted accordingly. This process of adding an agent, then correcting all the prior agent arrival times, is repeated until all agents have been added.

A small example case is presented here to demonstrate the arrival time assignment process. In this case, four agents seek to achieve a desired spacing of $S = 3$ sec. The achievable arrival times are given in Table 4.1. In the example presented, the final agent (4) is being added, and then the arrival times for all prior agents (1, 2, and 3) are adjusted. This represents the final iteration of the algorithm, after which the optimal arrival times have been found.

Table 4.1: Minimum and maximum arrival times for example optimization.

	Minimum Time (seconds)	Maximum Time (seconds)
Agent 1	2.0	4.0
Agent 2	8.5	10.5
Agent 3	11.5	14
Agent 4	15	17.5

At the current iteration of the algorithm, Agent 4 is placed at an arrival time of 15 sec given the prior guess of Agent 3's arrival time. In order to adjust the arrival times of the agents preceding Agent 4, the algorithm must determine which agents will be at their extreme values in the optimal solution, and then equally space the agents between them.

From the latest agent's arrival time, the maximum and minimum interval width is calculated for each preceding agent. Going from the penultimate agent to the earliest, the intersection of sequential interval widths is calculated until an agent is reached that doesn't have an interval within the intersection. To better explain this, as can be seen in the bottom plot of Figure 4.6, Agent 3 can achieve an interval of between 1 and 3.5 sec from the lower bound of Agent 4. This allows for the tentative placement of Agent 3 at 12 seconds (marked with a blue X in the top plot of Figure 4.6), which matches the desired spacing given Agent 4's arrival time of 15 sec. Considering Agent 2, as seen in Figure 4.7, the range of possible intervals shrinks to between 2.25 and 3.25. Since this still includes the desired spacing, a zero-error timing of agents 2, 3, and 4 can still be found.

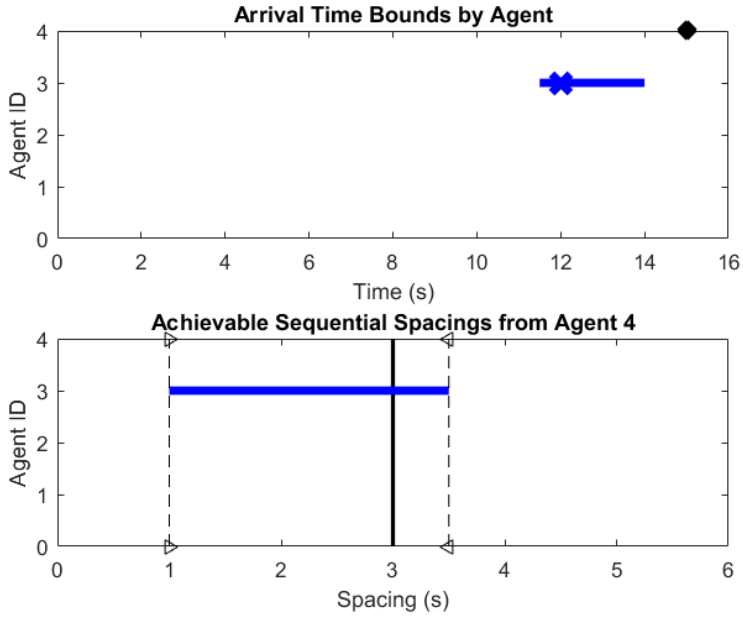

Figure 4.6: (Top) Span of Achievable Arrival Times for Agent 3. (Bottom) Achievable Intervals for Agent 3.

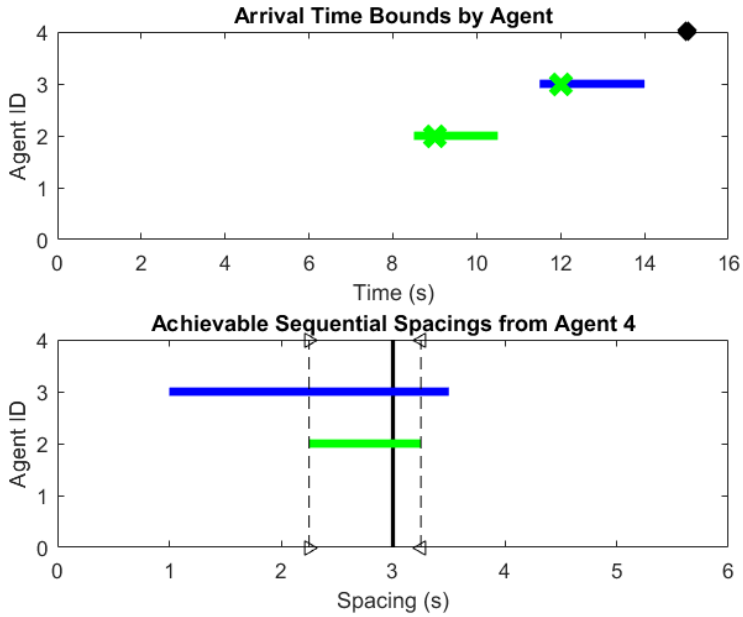

Figure 4.7: (Top) Agent 2 Added (Bottom) The Intersection Has Shrunk, But Still Includes the Desired Spacing.

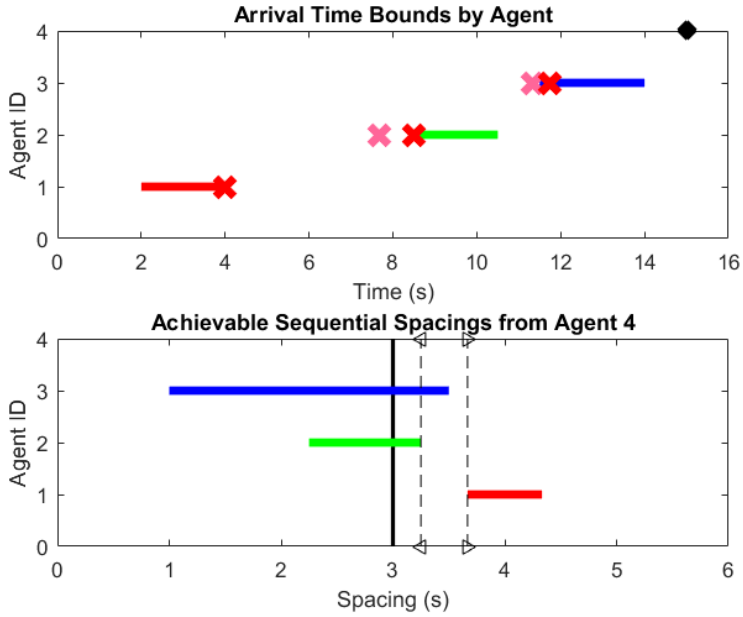

Figure 4.8: (Top) Equal Spacing (pink) is Not Achievable. (Bottom) Minimum Bound (▷) is Above the Maximum Bound (◁)

.

When Agent 1 is considered, its minimum spacing from Agent 2 is greater than the maximum spacing of the intersection so far as shown in the bottom of Figure 4.8. This bifurcates the problem and ensures that Agent 2 (which generated the maximum spacing in the intersection) is at its lower extreme. Agent 3 is then evenly placed between Agents 2 and 4. The justification for placing Agent 2 at its extreme value is shown in Figure 4.8, where the desired placement of Agent 2 (denoted with a pink x) is to the left of its achievable bounds.

With the arrival times of Agents 2 and 4 set, then the arrival time of Agent 3 is evenly spaced between them. Because the achievable sequential spacings between Agents 2 to 3 and Agents 3 to 4 overlap, the midway point is in the range of Agent 3. Thus, Agent 3 is placed evenly spaced between Agents 2 and 4. Finally, Agent 1's arrival time can be set so as to yield a gap with Agent 2 that is as close to the desired spacing as possible. Once Agent 1's arrival time is found, this completes the optimal timing solution for Agents 1-4.

This procedure is formalized in the pseudocode listed in Algorithm 4.1. The following

theorem states that this algorithm leads to the globally optimal arrival timing vector.

Algorithm 4.1 Arrival time optimization algorithm given lb and ub

```
Initialize empty vector t
Append lb[0] to t
for i = 1:length(lb) - 1 do
    Append min(max(t[i − 1] + S, lb[i]), ub[i]) to t
    for j = i:-1:0 do
        Get interval width limits to next fixed time and compare
        to intersection
        if contradiction found then
            Evenly space times in t from fixed to exceeded limit
            Update fixed time
            Reset intersection of interval widths
        end if
    end for
end for
return t
```

Theorem 4.2.3. *Given an arrival ordering of agents with bounded arrival times, Algorithm 4.1 will result in the optimal arrival time assignments.*

Proof. Without loss of generality, the first agent can be tentatively set to arrive at its earliest possible arrival time. This is trivially the optimal timing for a single agent. Assume $\vec{t}^{(m)}$ is the optimal timing found for the first m agents. To add agent $m+1$, the first step is to append

$$t_{m+1}^{(m+1)} = \min(\max(t_m^{(m)} + S, lb_{m+1}), ub_{m+1}) \tag{4.11}$$

where $t_{m+1}^{(m+1)}$ represents the $(m+1)^{st}$ element of $\vec{t}^{(m+1)}$. This minimizes the cost increase

$C_T\left(\vec{t}^{(m+1)}\right) - C_T\left(\vec{t}^{(m)}\right)$. Consider the optimal arrival time of an arbitrary agent i, which is neither the first nor last agent. Because C_T is a sum-of-squares cost, then agent i's arrival time will only be globally optimal if it is either at its maximum or minimum arrival time bound, or if it is equally spaced from agents $i-1$ and $i+1$. Likewise, the globally optimal arrival times for the first and last agents will either meet the desired spacing requirements exactly, or will be at the upper bound for the first agent and/or lower bound for the last agent. This has two implications: first that $t_{m+1}^{(m+1)}$ as given in (4.11) is at its optimal value and second that if

$$t_{m+1}^{(m+1)} - t_m^{(m)} \neq t_m^{(m)} - t_{m-1}^{(m)} \tag{4.12}$$

then by adjusting some of the values $t_1 \dots t_m$ the cost can be further reduced (but no lower than $C_T\left(\vec{t}^{(m)}\right)$).

Define a vector $\vec{\Gamma}$ and rewrite C_T as follows:

$$\begin{aligned} \Gamma_i &= t_{i+1} - t_i \\ C_T &= \textstyle\sum_{i=1}^{m} \left(\Gamma_i - S\right)^2 \end{aligned} \tag{4.13}$$

Let $t_{m+1} \equiv t_{m+1}^{(m+1)}$ for simplicity of notation. Since t_{m+1} is fixed, then $t_{m+1} - ub_m \leq \Gamma_m \leq t_{m+1} - lb_m$. Define an equally-spaced interval length between agent i and agent $m+1$ recursively as follows:

$$\begin{aligned} \Gamma_{i,\min} &= \max(\Gamma_{i+1,\min}, \tfrac{t_{m+1}-ub_i}{m+1-i}) \\ \Gamma_{i,\max} &= \min(\Gamma_{i+1,\max}, \tfrac{t_{m+1}-lb_i}{m+1-i}) \\ \Gamma_{i,\min} &\leq \Gamma_i \leq \Gamma_{i,\max} \end{aligned} \tag{4.14}$$

If $\Gamma_{1,\min} \leq \Gamma_{1,\max}$ then there exists at least one constant interval width which is achievable between each pair of agents given t_{m+1}. The constant interval width closest to S can be used to calculate the optimal timings $t_1^{(m+1)} \dots t_m^{(m+1)}$.

However, there are two ways that this procedure can generate a non-sensical interval width in which $\Gamma_{i,\min} > \Gamma_{i,\max}$. This occurs when $\frac{t_{m+1}-ub_i}{m+1-i} > \Gamma_{i+1,\max}$, or $\Gamma_{i+1,\min} >$

$\frac{t_{m+1}-lb_i}{m+1-i}$, and means that at least one agent should be placed at its lower bound (in the former case) or upper bound (in the latter case). Considering the former scenario, identify agent $j > i$ such that

$$t_{m+1} - lb_j = \Gamma_{i+1,\max}\,(m+1-j) \tag{4.15}$$

which means that agent j cannot achieve the equal spacing $\Gamma_{i,\max}$ desired between i and $m+1$ since the minimum achievable spacing $\Gamma_{i,\min} > \Gamma_{j,\max}$. Define the minimum equal spacing between any values of t_i and t_j as

$$\Gamma_{i\to j,\min} = \frac{lb_j - ub_i}{j-i} \tag{4.16}$$

and the maximum spacing between any value of t_j and the fixed t_{m+1} as

$$\Gamma_{j\to m+1,\max} = \Gamma_{j,\max} = \frac{t_{m+1} - lb_j}{m+1-j} \tag{4.17}$$

Because C_T has the structure of a sum of squares, it is minimized when $\Gamma_{i\to j}$ is as close as possible to $\Gamma_{j\to m+1}$. Since $t_j > lb_j \implies \Gamma_{i\to j} > \Gamma_{i\to j,\min}$ and $t_j > lb_j \implies \Gamma_{j\to m+1} < \Gamma_{j\to m+1,\max}$, then if the condition $\Gamma_{i\to j,\min} > \Gamma_{j\to m+1,\max}$, $t_j = lb_j$ is the optimal timing for agent j. This is because for $t_j > lb_j$, the difference in $\Gamma_{i\to j}$ and $\Gamma_{j\to m+1}$ is larger than it would be if $t_j = lb_j$.

It remains to be shown that the condition $\Gamma_{i\to j,\min} > \Gamma_{j\to m+1,\max}$ is equivalent to $\frac{t_{m+1}-ub_i}{m+1-i} > \Gamma_{i+1,\max}$. Substituting into the first inequality the values from Eqs. 4.16 and 4.17 yields,

$$\frac{lb_j - ub_i}{j-i} > \frac{t_{m+1} - lb_j}{m+1-j} \tag{4.18}$$

Cross multiplying these fractions and substituting in $j - i = (m+1-i) - (m+1-j)$ yields

$$(t_{m+1} - ub_i)\,(m+1-j) > (t_{m+1} - lb_j)\,(m+1-i) \tag{4.19}$$

Rearranging terms from here and using the definition in (4.14) yields the second inequality. Based on the criteria after Eqn. (4.17), this means that $t_j^{(m+1)} = lb_j$. An analogous process can be used to show that $t_j^{(m+1)}$ should be placed at ub_j if $\Gamma_{i+1,\min} > \frac{t_{m+1}-lb_i}{m+1-i}$.

With $t_j^{(m+1)}$ and $t_{m+1}^{(m+1)}$ fixed, the times of the agents arriving between them can be equally spaced. From Eq. (4.14), it can be seen that every agent in this range can achieve an equal spacing of $\Gamma_{i+1,\max}$. With these agents optimally timed and t_j fixed, the above process is repeated from $j, j-1, ..., 1$ or until the next agent that must be placed at its limit is detected. If the first agent is reached and $\Gamma_{1,\min} \leq \Gamma_{1,\max}$, then the agents from agent 1 to the earliest agent fixed at one of its bounds should be equally spaced by $\Gamma = \min(\max(S, \Gamma_{1,\min}), \Gamma_{1,\max})$. This yields the optimal $\vec{t}^{(m+1)}$.

Since it is shown that given an optimal $\vec{t}^{(m)}$ an iteration of the main body of the optimization algorithm gives the optimal $\vec{t}^{(m+1)}$ and the initial step starts with the optimal $\vec{t}^{(1)}$, the algorithm finds the optimal timing vector.

□

4.2.2 Composite Cost Function

The problem of simultaneously optimizing agent assignments under the effectiveness-based cost as well as arrival time cost is one of multi-objective optimization. Since these costs may yield conflicting solutions, their relative priority can be established via a scalar weighting factor in a composite cost function. This allows for the use of a composite cost function given by

$$C_{\text{COMP}}(Q) = \alpha C_{ST}(Q) + (1-\alpha) C_T(Q) \tag{4.20}$$

where $0 < \alpha < 1$. The scalar α is set by an operator prior to the scenario, and weighs the importance of achieving the desired Pk objectives versus the timing objectives. Note that $\alpha = 1$ is a valid setting and reduces the composite cost to the sufficiency threshold Pk cost from Ch. 3. Likewise, while theoretically $\alpha = 0$ is also valid, from a practical standpoint this setting is undesirable as the Pk is not considered at all by the agents.

4.3 Results

4.3.1 Example Cases

A variety of results are presented here to illustrate both the ability of the combined cost function to induce the desired behavior and the performance of the branch and bound algorithm in solving the multi-objective problem, particularly in comparison to greedy search. For the results in this section, the agents and targets are uniformly randomly distributed over their respective starting areas. The agents are initially distributed over an area 800 meters by 80 meters. The target starting location is a square that is 800 meters on each side and is offset by 2500 meters North and 2500 meters East from the agent area. For the engagements detailed in Figures 4.9-4.12, 27 homogeneous agents with $Pk_{i,j} = 0.5$ engage 9 targets, evenly split into three groups of $Pk_{\text{des},1-3} = 0.9$, $Pk_{\text{des},4-6} = 0.8$, and $Pk_{\text{des},7-9} = 0.7$. All results shown in this section use values of $v_{h,nom} = 10.0$ m/sec and $\dot{\psi}_{max} = 1.265 \left(\frac{rad}{s^2}\right)$. Note that in all simulations performed here a communication radius of 600 m is used. This value is selected to be large enough to ensure network connectivity in the majority of simulation cases, but small enough that rebroadcasting must occur and disconnections do happen.

Figures 4.9-4.12 each summarize a single engagement scenario. Agents travel from their initial starting location to their targets, continually computing the solution to the assignment problem. The horizontal axis of these figures represent the elapsed time during an engagement but also, indirectly, shows the progression of the branch and bound algorithm. Inside each decision round, each agent will expand two nodes of the search tree and decision rounds are assumed to occur every 0.10 seconds.

The top half of these plots uses color to represent the achieved $Pk_{\Sigma,j}$ against each target as a function of time. The end of the horizontal bars represent the targets being destroyed. The bottom half represents the target selected by each agent at each timestep. Note that for ease of reading the agents are reordered in a data processing step so that they are ordered

by target and arrival time. In the bottom plots of Figures 4.9 and 4.11, which show cases where ΔV is quite restrictive, note that the solution state (i.e., best assignment found so far) changes multiple times at the beginning of the scenario. This is due to branch and bound finding better assignments as it progresses. For all cases in this section, equal weight is given to the C_{Pk} and C_T cost functions ($\alpha = 0.5$).

Figure 4.9 shows a case in which $\Delta V = 0.02 v_{h,\mathrm{nom}}$ and $S = 0$, indicating a desire for simultaneous arrival. Due to the relatively low value of ΔV, agents have limited control over their arrival times. Furthermore, with $\alpha = 0.5$, C_T has a large effect on the quality of an assignment. Thus it takes several rounds for branch and bound to find the optimal solution as many assignments must be evaluated. In contrast, Figure 4.10 shows a case in which $\Delta V = 0.10 v_{h,\mathrm{nom}}$. This converges immediately as the larger ΔV increases the number of possible assignments that can achieve the desired spacing. Comparing the arrival times of agents (the ends of the bars in the bottom plots) in Figs. 4.9 and 4.10, there is some minor variation in arrival times for the more restrictive ΔV, but almost perfect simultaneity is achieved with $\Delta V = 0.10 v_{h,\mathrm{nom}}$. The metrics in Table 4.2 confirm this trend, and also show a comparison against a $\Delta V = 0$ case (not shown here).

Figures 4.11 and 4.12 show similar examples, except this time using $S = 4$ sec. The same overall trends are observed and confirmed with the metrics in Table 4.2, with the $\Delta V = 0.10 v_{h,\mathrm{nom}}$ case achieving the desired spacing more precisely than the more restrictive cases.

For all of the results presented above, the agents were able to reach a state with $C_{ST} = 0$, meaning that the desired Pk's on all targets were met for the steady-state solution. For the $\Delta V = 0$ cases, three agents elected not to engage any targets as they would increase C_T by more than they decreased C_{ST}. It is for this reason that in Table 4.2 the RMS timing error associated with $\Delta V = 2\%$ is higher than that of $\Delta V = 0$. Note for all values of ΔV that the error associated with sequenced arrival is higher than that for simultaneous arrival. This is due to the engagement scenario geometry causing the initial agent-target distances

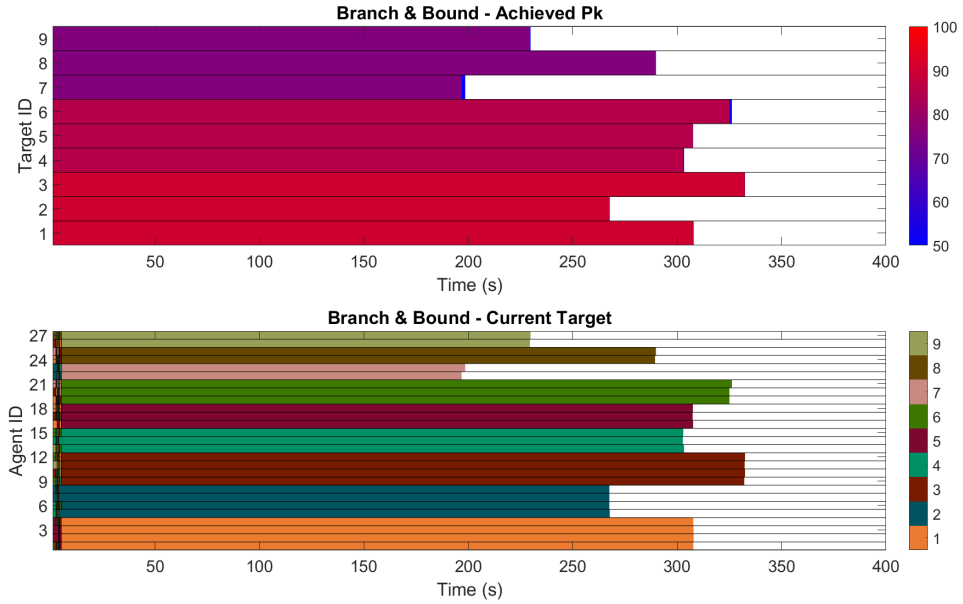

Figure 4.9: Example simulation with 27 agents, 9 targets, S = 0 seconds, $\Delta V = 0.02 v_{h,\mathrm{nom}}$.

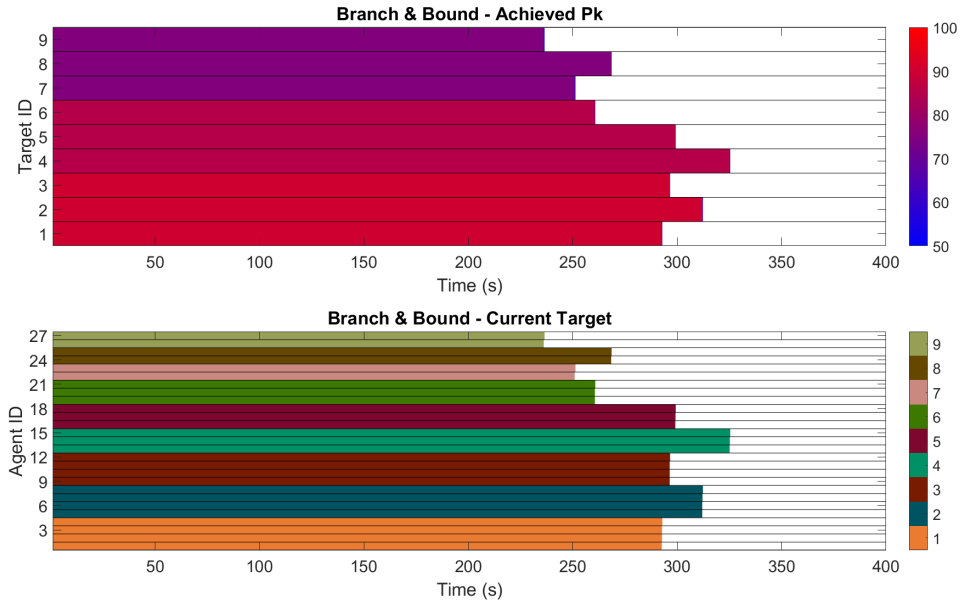

Figure 4.10: Example simulation with 27 agents, 9 targets, S = 0 seconds $\Delta V = 0.10 v_{h,\mathrm{nom}}$.

to be similar for all agents. With a constrained ΔV, the achievable arrival time differences between agents are not substantial. Thus two agents are unlikely to have substantially

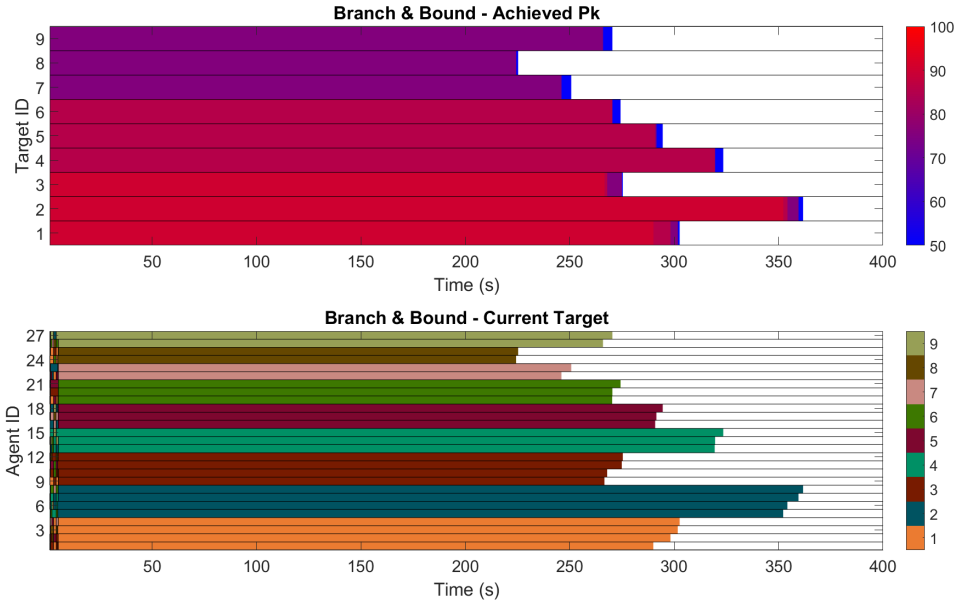

Figure 4.11: Example simulation with 27 agents, 9 targets, S = 4 seconds, $\Delta V = 0.02 v_{h,\text{nom}}$

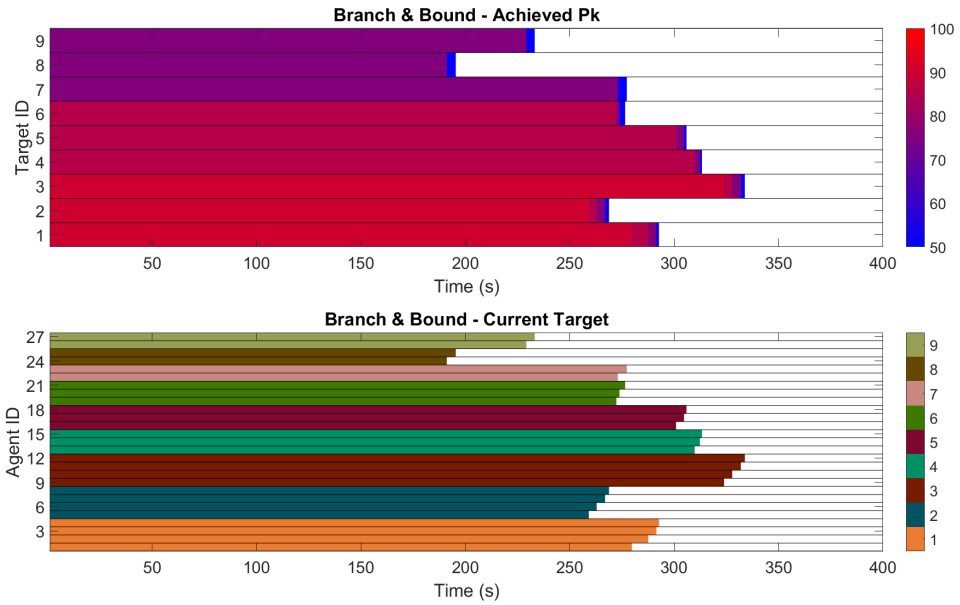

Figure 4.12: Example simulation with 27 agents, 9 targets, S = 4 seconds, $\Delta V = 0.10 v_{h,\text{nom}}$

Table 4.2: Achieved versus desired Pk and root mean square error in arrival timing.

	S = 0 sec		S = 4 sec	
	$Pk_\Sigma\,[Pk_{\rm des}]$	RMS Error (sec)	$Pk_\Sigma\,[Pk_{\rm des}]$	RMS Error (sec)
$\Delta V = 0\%$	0.9375 [0.9] 0.7500 [0.8] 0.7500 [0.7]	1.271	0.9375 [0.9] 0.7500 [0.8] 0.7500 [0.7]	1.883
$\Delta V = 2\%$	0.9375 [0.9] 0.8750 [0.8] 0.7500 [0.7]	3.204	0.9375 [0.9 0.8750 [0.8] 0.7500 [0.7]	2.121
$\Delta V = 10\%$	0.9375 [0.9] 0.8750 [0.8] 0.7500 [0.7]	0.153	0.9375 [0.9] 0.8750 [0.8] 0.7500[0.7]	1.509

different arrival times, which makes large intervals with many agents difficult to achieve.

4.3.2 Monte Carlo Simulations

To illustrate the trade-off between timing and Pk objectives inherent in the composite cost function, a set of Monte Carlo simulations was performed for a heterogeneous engagement scenario. Each simulation consisted of 10 agents and 4 targets using the same scenario geometry as in the previous section. The specific weapon effectiveness values are given in Table 4.3. For each of several α values between 0 and 1, a total of 200 simulations was performed and the results averaged at each α value. For low α values (i.e., 0.1) the cost function prioritizes timing, while for high values (i.e., 1.0) the cost function prioritizes achieved Pk. For comparison purposes, these Monte Carlo simulations were performed twice —once using greedy search as the solution algorithm, and once using Branch and Bound as the solution algorithm. Furthermore, each set of experiments was performed for three different values of ΔV (0, $0.02v_{h,\rm nom}$, and $0.10v_{h,\rm nom}$).

In Figs. 4.13-4.15 the average achieved Pk for each of two target categories, as well as the average arrival time cost, is plotted for both greedy search and branch and bound at $\Delta V = 0$, $\Delta V = 0.02v_{h,\rm nom}$, and $\Delta V = 0.10v_{h,\rm nom}$, respectively. In all three cases the desired arrival time interval $S = 0$. As can be seen in Figure 4.13, as the value of α

Table 4.3: Weapon effectivenesses and Pk_{des} for Monte Carlo simulations.

	$Pk_{des,1-2}$ = 0.90	$Pk_{des,3-4}$ = 0.75
Agents 1-4	0.15	0.50
Agents 5-10	0.50	0.15

increases, the average achieved $Pk_{\Sigma,j}$ tends to converge to the desired value. Furthermore, the arrival time cost grows with increasing α. The same trends can be seen for both algorithms; however, the performance of branch and bound exceeds that of greedy search for all values of α, particularly for high-value targets.

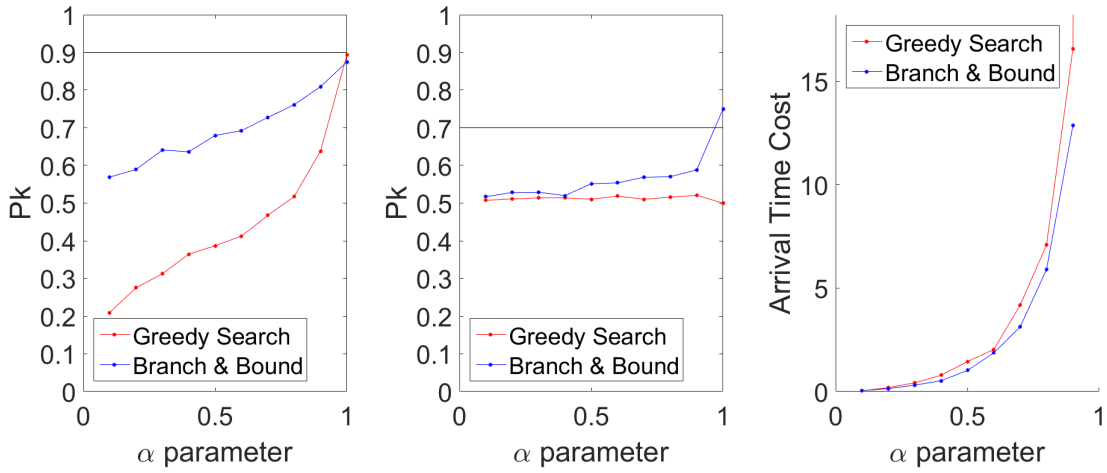

Figure 4.13: Monte Carlo results ($\Delta V = 0$): (Left) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.9$, (Center) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.75$, and (Right) Average arrival time cost.

If the agents are able to vary their velocities, the trade-off between Pk and simultaneity becomes less pronounced. This is because the same assignment will have a lower time-based cost than if $\Delta V = 0$, and thus the time-based cost has less of an impact on the assignment decision. Figure 4.14, showing an analogous case to Figure 4.13 except with $\Delta V = 0.02 v_{h,\text{nom}}$, demonstrates that agents can achieve higher Pk and lower arrival time cost simultaneously. Overall, the results of Figure 4.14 show that branch and bound outperforms greedy search when considering the combined results of $Pk_{\Sigma,j}$ and simultaneity.

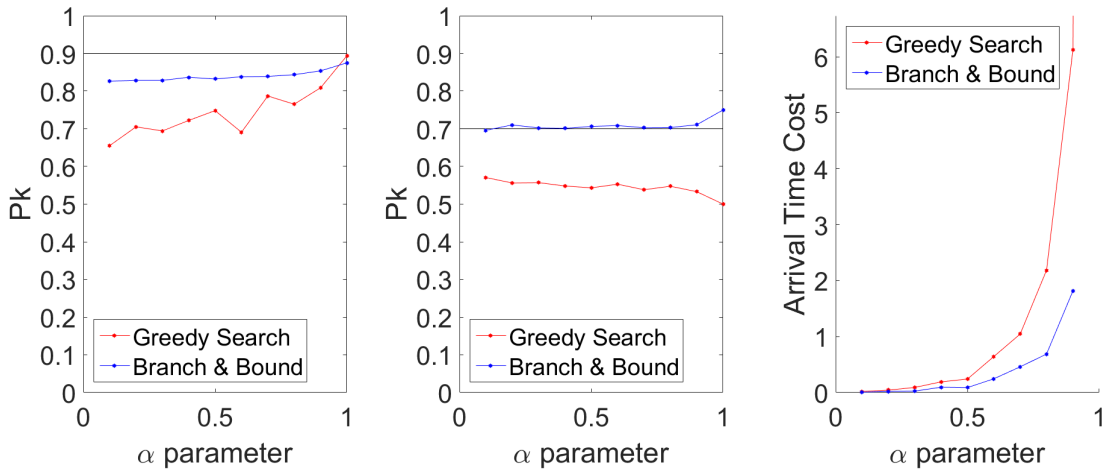

Figure 4.14: Monte Carlo results ($\Delta V = 0.02 v_{h,\text{nom}}$): (Left) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.9$, (Center) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.75$, and (Right) Average arrival time cost.

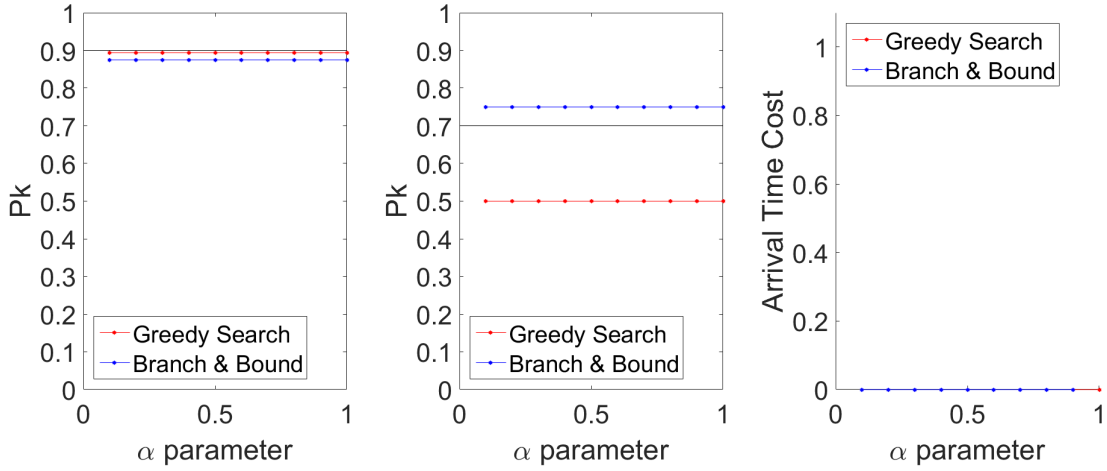

Figure 4.15: Monte Carlo results ($\Delta V = 0.10 v_{h,\text{nom}}$): (Left) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.9$, (Center) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.75$, and (Right) Average arrival time cost.

When the scalarization parameter is reframed as the relative priority ($\frac{1-\alpha}{\alpha}$) then the arrival time cost varies much more linearly. This can be seen in Figure 4.16 where the arrival time cost varies much more linearly when plotted against relative priority of cost functions than when plotted strictly against α in Figure 4.13.

If the range of closing velocities for each agent is large enough, the timing and Pk

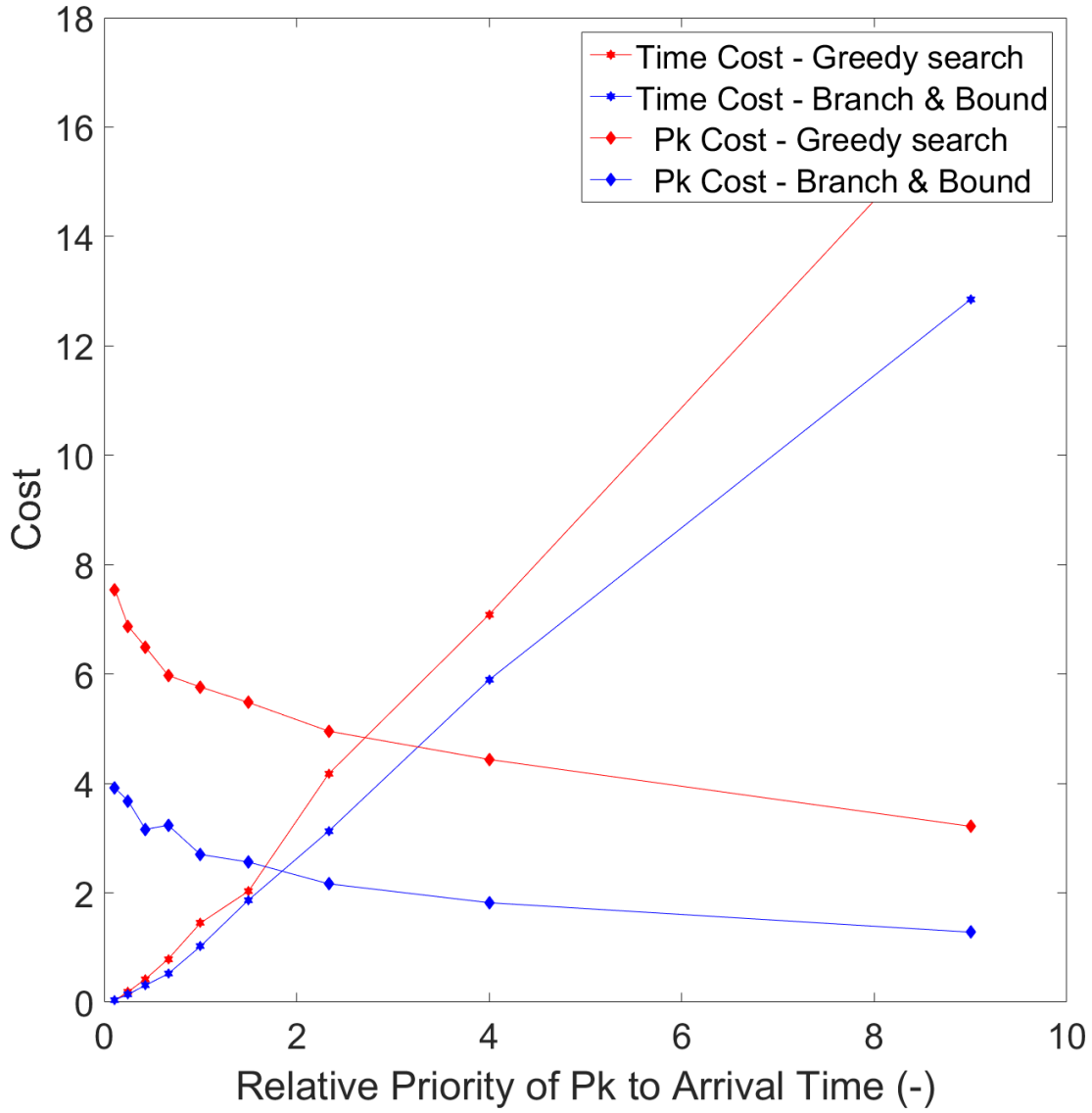

Figure 4.16: Rescaled tradeoff between Pk and arrival time costs taken from Figure 4.13.

objectives become completely decoupled, and any assignment solution that optimizes the Pk-based cost function can yield timing assignments that optimize the arrival time cost function. As a result, the α value has no effect on the solution found. This can be seen in Figure 4.15, where ΔV is increased to 10% of $v_{h,\text{nom}}$. In this case $Pk_{\Sigma,j}$ is constant for all α and the arrival time cost is identically zero for both branch and bound and greedy search.

Note that depending on the set of agents and their effectivenesses, ΔV need not be so

large that every optimal solution to the Pk-based cost function is also an optimal solution to the arrival time-based cost function, but rather bifurcation occurs when a solution exists that is optimal for both.

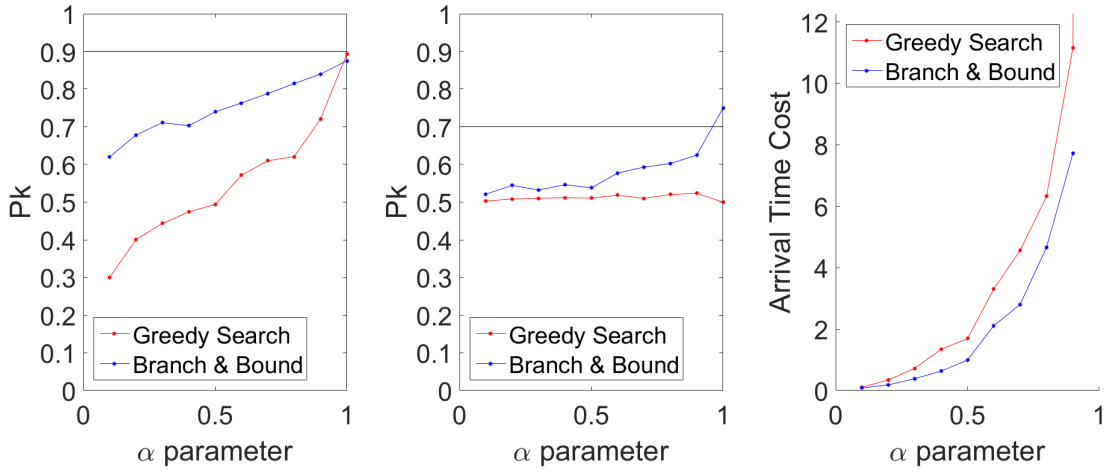

Figure 4.17: Greedy search versus branch and bound with $\Delta V = 0$. (Left) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.9$, (Center) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.75$, and (Right) Average arrival time cost.

All of the Monte Carlo results presented so far have been aimed at inducing simultaneous arrival from the agents. As mentioned above, simultaneous arrival is simply a special case of sequenced arrival with an interval of zero seconds. To illustrate this, the same values of ΔV and α were run with a desired interval of four seconds. Comparing Figure 4.17 to Figure 4.13 shows very similar values for both algorithms and nearly identical trends but with different values.

For each Monte Carlo simulation using branch and bound, the size of the state space was recorded at each time step to measure the extent of the pruning achieved by the algorithm. Figure 4.20 shows the average state space size at every time step for several values of α. For lower values of ΔV the algorithm takes much longer to prune possible assignments due to the nature of the relaxation. When expanding a node, the unassigned agents are temporarily given infinite variability of their arrival times so that the lower bound is guaranteed to be at least as low as the optimal given the parent assignment. As a result, at the beginning there

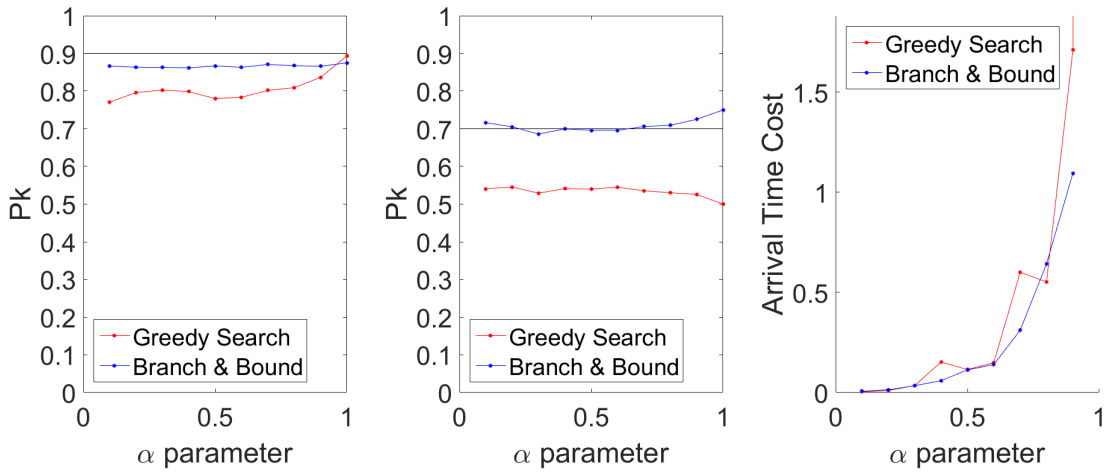

Figure 4.18: Greedy search versus branch and bound with $\Delta V = 0.02 v_{h,\text{nom}}$. (Left) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.9$, (Center) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.75$, and (Right) Average arrival time cost

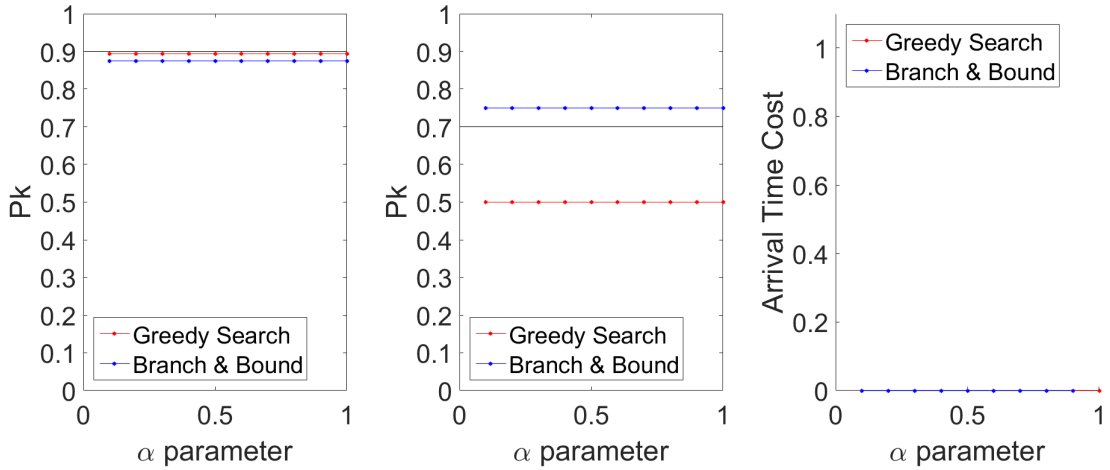

Figure 4.19: Greedy search versus branch and bound with $\Delta V = 0.10 v_{h,\text{nom}}$. (Left) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.9$, (Center) Average $Pk_{\Sigma,j}$ for $Pk_{des,j} = 0.75$, and (Right) Average arrival time cost.

are many unassigned agents and so the lower bound is much lower than both the upper bound and true optimal value, and thus very few potential assignments are eliminated. With higher values of ΔV the bounds on the optimal cost are much tighter and pruning occurs much more frequently. Note that when $\Delta V = 0.10 v_h$ almost every suboptimal solution

can be pruned immediately. However, it is also important to note that there is a difference between pruning the solution space and finding the optimal (or a near-optimal) solution that can be used immediately. In most cases the best solution found early in the scenario is quite good, although the algorithm might not be able to provably find the global optimum through pruning until much later.

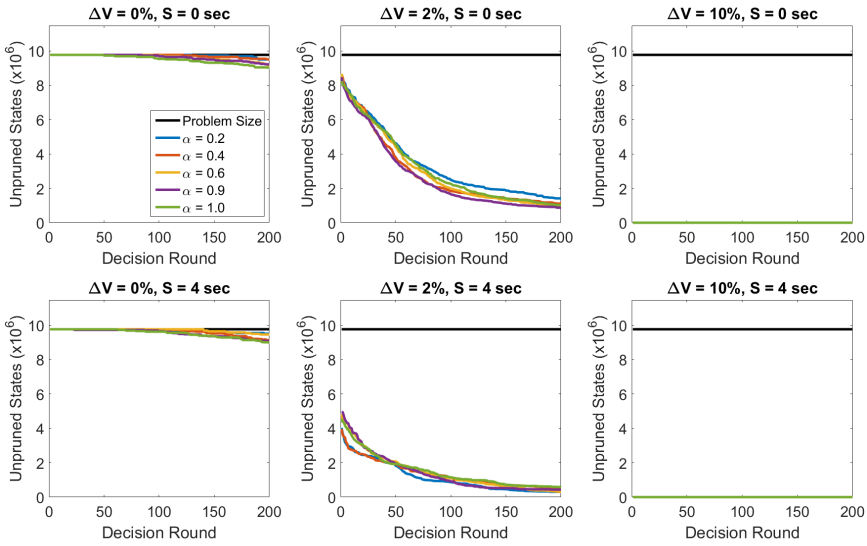

Figure 4.20: State space size vs. decision round for Monte Carlo cases.

The results shown so far are limited to one specific set of agents and targets with fixed weapon effectiveness values. To verify that the superiority of branch and bound compared to greedy search is not limited to this specific set of heterogeneous agents, a set of eight weapons with effectiveness values uniformly randomly distributed in the range $0.25 \leq Pk_{i,j} \leq 0.5$ were used against a set of four targets with desired Pks uniformly randomly distributed in the range $0.95 \leq Pk_{des,j} \leq 0.99$. For each α value, 100 simulations were performed and for each simulation the converged assignment was calculated using greedy search and branch and bound, and compared to the actual optimum computed through full-state enumeration of all possible assignments. The number of solutions that failed to achieve the optimal was recorded, as was the percent difference of the achieved cost from the optimal.

Figure 4.21 shows the results of this study, demonstrating that branch and bound largely avoids the local minima that lead greedy search to yield a suboptimal solution. When branch and bound does yield suboptimal performance, it is much closer to the actual optimal cost. Because the values for $Pk_{\text{des},j}$ and $Pk_{i,j}$ are generated randomly, the high frequency of suboptimal solutions with greedy search shows that local minima exist generally and are not a unique feature of specific scenarios. At the same time, note that for all values of α, greedy search always yields a solution that is within about 20% or better of the optimal cost. Given the fact that greedy search is much simpler to implement than branch and bound, this result is significant in that, for some applications, it may be preferable to implement greedy search and accept a suboptimal (albeit usable) solution as the price to pay for algorithm simplicity. The acceptable tradeoff between performance and ease of implementation is system-specific, but the results in Figure 4.21 provide an example of the type of metrics that may be used to inform such a decision.

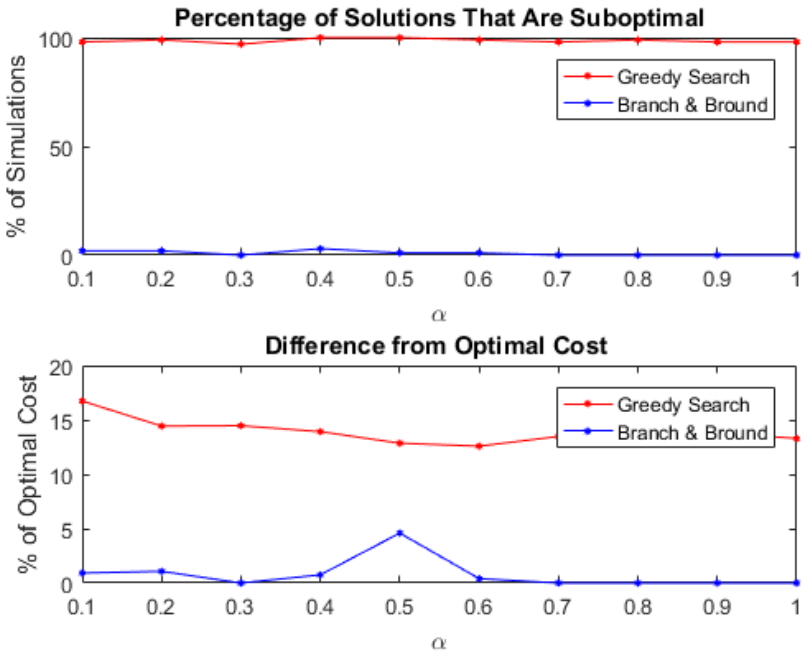

Figure 4.21: Comparison of solutions achieved by greedy search and branch and bound.

CHAPTER 5
CONTINGENCY ASSIGNMENTS IN PROBABILITY OF KILL CALCULATION

5.1 Introduction

In the simulation experiments shown above it was observed that the estimate of Pk prior to any of the agents reaching their targets is always lower than the measured posterior Pk calculated by tallying the number of targets killed in simulation (assuming sufficiently fast battle damage assessment). This is because once an agent destroys its target, any remaining agents are free to pursue other targets that are within reach, leading to higher achieved Pk's on these secondary targets. In this chapter, a novel approach to computing PK is investigated to mitigate this phenomenon. Agents are allowed to contribute to the total Pk on multiple targets by weighting their effectiveness according to the probability that they will abandon their first target. It is believed that this will provide a more accurate measure of the achieved Pk which, in turn, will allow for a more efficient use of limited resources.

This approach is not without downsides, however. One issue that must be dealt with is the sheer scope of the state space of potential solutions —each agent can have a primary backup target, a secondary backup target, and so on, up to $M - 1$ backup targets. In the traditional WTA problem the number of potential assignments grows exponentially (number of targets to the power of the number of agents, M^N), but in this work the ordering of targets for a single agent alone has as a search space that is factorial in the number of targets. Thus, to exhaustively search the entire space would require looking at the number of targets factorial to the power of the number of agents. This is clearly unfeasible, even for moderately-sized scenarios, so heuristic based methods such as greedy search or branch and bound are needed to solve this problem. The solutions found with this new approach to computing Pk will be compared to those found by the previous approach in order to assess

its utility and benefit.

Recall from Ch. 3 that for a homogeneous set of agents greedy search is optimal and it is near optimal in many heterogeneous cases. What was not addressed, however, is that if the arrival of the agents are spread out through time and weapons are allowed to update their assignments throughout the engagement, then the achieved Pk on each target is greater than or equal to the calculated Pk at any point during engagement. If the assignments are made based on an inaccurate measure of the combined effectiveness, then it is possible that they are making suboptimal decisions.

Additionally, in a situation in which the effective Pk is being determined offline for the purposes of determining what weapons are necessary to achieve the desired Pk's, the ability to more accurately estimate Pk may allow for equivalent performance with fewer weapons. Alternatively, if the reachability of some targets is a function of agent paths, then a path planner that is aware of contingency (or backup) targets can find paths that maintain reachability on those targets for as long as possible.

In this chapter, the selection of multiple targets is investigated. Specifically, each weapon will select a reserve target and a flight path to reach the primary and secondary target. Once the probability of each weapon becoming redundant on its primary target is known, a more accurate Pk can be calculated recursively. Assigning multiple targets to a single weapon system is not novel and has been explored previously, for instance by Rosenberger [50]. However, in that work the systems being used are capable of engaging multiple targets in series rather than being able to only engage the first undestroyed target it reaches. Despite this difference, Rosenberger presents a good explanation of applying branch and bound methods for this type of problem. Typically, branch and bound algorithms are implemented in a centralized manner; however, there has been some work (for example by Bader [51]) on parallel methods. While Bader's work is in large part focused on modeling of various computer architectures, it also contains sound insight into solving assignment problems on distributed memory systems using branch and bound

techniques.

5.2 Methodology

With the exception of the Pk_{Σ} calculation and the assignment algorithm, the simulation environment used here is substantially similar to the one presented in Ch. 2. The differences are presented below.

5.2.1 Dynamic Model

The dynamic model used in this chapter differs from that presented in Ch. 2 in a few notable elements. Agents are assumed to fly level paths to their targets where they make a final BDA and either engage or proceed to their next target. Conceptually this is similar to standoff munitions such as the AGM-114 "Hellfire" or GBU-44 which operate in a similar way. Additionally, the agents are assumed to fly in straight lines at constant velocity ($\Delta V = 0$).

5.2.2 Pk Estimation

The cornerstone of this chapter is the inclusion of a contingency plan in the estimate of achieved Pk. The cost functions presented in Ch. 3 and 4 rely on accurate estimates of the achieved Pk on each target, but it was observed that the calculated probabilities consistently underestimate the posterior Pk as measured by the final BDA. The instantaneous estimate of Pk assumes each agent will continue to engage its current target and treats the individual contributions to Pk as independent probabilities. In reality, the achieved Pk is time-varying and leveraging knowledge of how it varies allows for a more efficient use of agents.

When an agent reaches its target the Pk on that target will go to one if it is successfully destroyed, or it will decrease since there are now fewer agents engaging the target. The $Pk_{\Sigma,\mathrm{j}}$ on target j after an unsuccessful engagement by agent i is denoted $Pk_{-\mathrm{i},\mathrm{j}}$ and given

by Eq. 5.1.

$$Pk_{-\mathrm{i,j}} = 1 - \frac{Pk_{\Sigma,\mathrm{j}}}{Pk_{\mathrm{i,j}}} \tag{5.1}$$

If a target is successfully engaged, then any other agents assigned to it will select their contingent assignment. If the arrival times of each agent are known, then the probability that an agent will arrive at a successfully engaged target can be calculated. From this, the weapon effectiveness against its secondary target can be scaled by this value, which is equivalently the likelihood of engaging the secondary target. The Pk estimation algorithm implemented here is generalized for $1 \leq l \leq M$ levels of contingency, however only a single additional layer is considered in the results section.

To account for the effect of successful engagements on Pk, a list of $l \times N$ arrival events is created. These events are sorted sequentially and then a $Pk\left(t\right)$ can be calculated at the time of arrival for each agent's primary and contingency targets. If an agent is to arrive at its primary target j at time t_1 then its effectiveness against its contingency target k is given by Eq. 5.2. The full algorithm for calculating Pk_{Σ} is given in Algorithm 5.1.

$$\hat{Pk}_{\mathrm{i,k}} = Pk_{\Sigma,\mathrm{j}}\left(t_1\right) \times Pk_{\mathrm{i,k}} \tag{5.2}$$

Algorithm 5.1 Algorithm for calculating Pk incorporating contingency assignments.

Initialize zero vector Pk of size M

Initialize vector E of size $l \times N$

Sort E by arrival times

for i = 1..$l \times N$ **do**

$j_i = E[i]$.target

$t_i = E[i]$.time

$p_i = E[i]$.effectiveness

$k = E[i]$.contingency

$p_k = Pk[i] * p_k$

$Pk[j_i] = 1 - (1 - Pk[j_i])(1 - p_i)$

end for

return Pk

5.3 Assignment

The assignment algorithms presented earlier must be modified to accommodate the additional state variables that the secondary targets represent. There are two obvious approaches to this problem. The first is to assign targets at a given level to all agents before proceeding to the next level. The second, and the one implemented here, is to treat a partial permutation (alternatively known as a *k-permutation*) of length $l(k)$ as the state variable and assign all levels at once for each agent. The number of potential assignments at all levels, given in Eq 5.3 is a well known formula [66]. The dramatic increase in number of potential assignments can be seen in Figure 5.1.

$$P(M, l) = \frac{M!}{(M - l)} \tag{5.3}$$

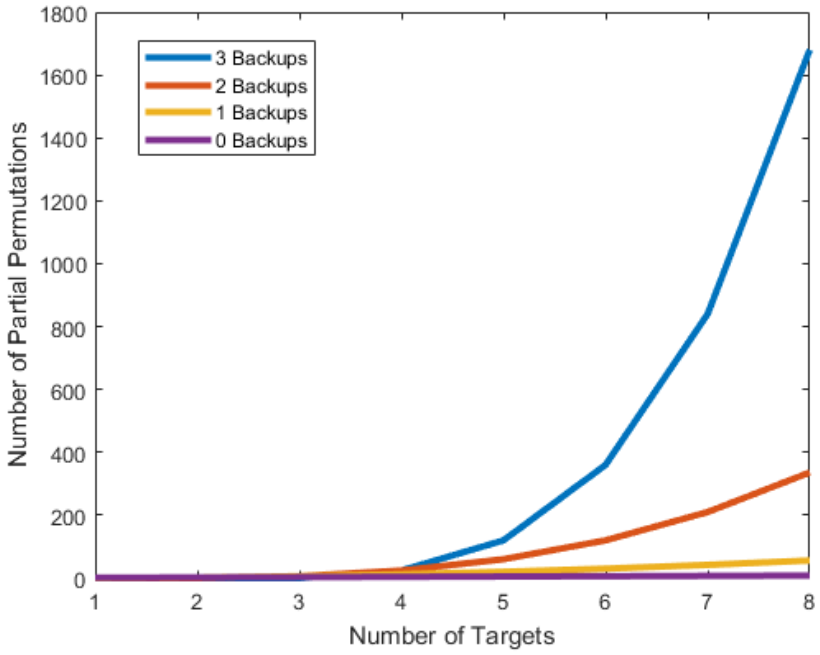

Figure 5.1: Number of partial permutations as a function of the number of targets and contingencies.

For this investigation, only greedy search is considered due to the the size of the search space making branch and bound impractical. Generalizing the size of the search space from the $l = 1$ case to the $1 \leq l \leq M$ case gives a search space of size of $N^{\frac{M!}{(M-l)!}}$.

The greedy search algorithm implemented by each agent generates and evaluates all $P(M, l)$ partial permutations given a starting partial assignment, and selects the one that resulted in the largest decrease in the cost function. For the scenarios presented in Sec. 5.4, a decision round evaluates $2,430$ out of a possible 6.65×10^{128} assignments. It is shown in Sec 5.4 that greedy search achieves reasonable performance for the scenarios considered here, despite the large search space. The weak dependence of Pk on arrival times does introduce some heterogeneity and approaches such as branch and bound are of interest, but beyond the scope of this work.

5.4 Results

The modification to the Pk calculation proposed here seeks to achieve two goals. The first is to achieve a lower-cost state by including the contribution against contingency targets, and the second is to more accurately calculate Pk such that it better matches the posterior Pk computed at the end of the scenario. In order to evaluate performance with respect to these goals, 1,000 simulations were run with 27 agents and 10 targets, with $Pk_{\text{des,j}} = 0.9$ for odd-numbered targets and $Pk_{\text{des,j}} = 0.7$ for even-numbered targets. The same agents and targets were used in each case, but the initial positions of each was randomized. Note that when the contingency is neglected, there is no dependence on the target and agent location so the same solution is found for each case. With the contingency incorporated, the effectiveness against the secondary target is affected by the initial positions because the agents arrive at their targets in different orders. The geometry of the engagement scenario is the same as in Ch. 4, however the agents have a higher lateral velocity of $v_h = 100$ m/s.

The second goal is to show that incorporating contingency targets allows for a more efficient use of resources by lowering the posterior cost as measured by calculating the percentage of targets actually destroyed at the end of the simulation and applying the cost function to that distribution. As mentioned above, when using either Eq. 2.6 or Algorithm 5.1 to calculate Pk, the prior probability of kill estimates will in both cases be lower than the actual posteriori Pk measured by the simulation results. However, the achieved Pk estimates should be better for the simulations run with the new probability model as agents will not be wasted engaging targets that do not actually need additional agents.

Tables 5.1 and 5.2 show the estimated and empirical Pk_Σ with and without contingency targets. The "One Level" case uses the Pk_Σ calculation from Eq. 2.6 to evaluate potential assignments, while the "Two Level" case denotes the new Pk model with a single contingency target. In order to reliably assess the empirical Pk_Σ, 1,000

simulations were run for each case and the results were averaged. The tables are laid out with the first two data columns representing the estimated Pk considering only the primary targets, the next two representing the estimate with one contingency, and the final two showing the posterior Pk as calculated after the simulation.

In Table 5.1, the agents were entirely homogeneous with a $Pk_{i,j} = 0.35$. Note that even though this scenario is referred to as homogeneous, some mild heterogeneity is observed in the single contingency case as Pk_{Σ} is mildly coupled with arrival times. For completeness, the estimated Pk_{Σ} was calculated as though there were no contingency targets for these cases (lower left elements of the table) and an interesting effect is observed. The estimated Pk_{Σ} is actually lower for both categories of targets and while this is seemingly counterintuitive, it is a consequence of the nonlinear combination of Pk's and the optimization algorithm. As described above, agents select a partial permutation representing the sequence of targets for that agent. Consider the first agent to select. It will select a high-priority target as its primary but is also likely to select a high-priority target as its secondary. As a result, some high-priority targets will predominately be selected as primary targets and some predominantly as secondary targets. When the secondary targets are ignored, this results in a large disparity in Pk_{Σ} and thus a much lower average.

In these cases, the agents had unrestricted ability to select a new target at any point in the engagement and even still the agents with secondary targets were able to achieve a higher total effectiveness against lower priority targets. In a situation where agents are forced to commit to their plans, then having the contingency target would have an even more dramatic effect on the efficacy of the agents. The already high achieved Pk against high-priority targets left little room for improvement. Expressed as percentages, the estimated Pk's when no contingency targets were assigned were 85.4% and 60.3% of the achieved Pk's, but when one contingency target is considered these percentages increase to 94.3% and 93.2%. This increase in accuracy illustrates the efficacy of considering contingency targets to improve Pk estimation, indicating favorable performance with respect to the

second objective described above.

Table 5.1: Simulation results for a homogeneous scenario.

	Estimated (Original)		Estimated (Revised)		Empirical	
	Pk = 0.9	Pk = 0.7	Pk = 0.9	Pk = 0.7	Pk = 0.9	Pk = 0.7
One Level	0.8215	0.4410	-	-	0.9620	0.7318
Two Levels	0.7753	0.4152	0.9113	0.7173	0.9666	0.7700

Figure 5.2 gives the estimated Pk achieved by each simulation in this Monte Carlo run. Notice that for the original Pk estimation function, the values are constant because the scenario is homogeneous and so the same solution is reached every time. The simulations run with the revised Pk estimation function show some heterogeneity induced by the random initial agent and target locations affecting the ordering. More importantly, note that when considering the contingency target, the same set of agents is able to achieve the desired Pk's of 0.9 and 0.7, compared to the no-contingency case which predicts that the achieved Pk's fall short. Thus, while one might conclude that this weapon set is incapable of achieving the desired Pk levels when using the original calculation for Pk, once the Pk computation is improved to include contingency targeting it is seen that the weapon set is actually capable of achieving the desired Pk. This indicates favorable performance with respect to the first objective mentioned above.

Table 5.2 shows similar Monte Carlo results, except using heterogeneous agents with effectiveness values uniformly randomly distributed in the range $0.25 \leq Pk_{i,j} \leq 0.45$. As before, the new Pk estimation method yields a better estimate than the original estimation method. The heterogeneity in weapon effectiveness makes a one-to-one comparison of the posterior Pk difficult, but recall that the simulation used represents a best case scenario for retargeting with no contingency. The one level case yields Pk estimates that are 87.7% and 73.7% of the achieved Pk, compared to 93.1% and 88.5% for the two level cases.

Figure 5.3 shows the estimated Pk for every simulation. Note that the estimates vary over a much greater range for simulations using the original estimation function. This is due

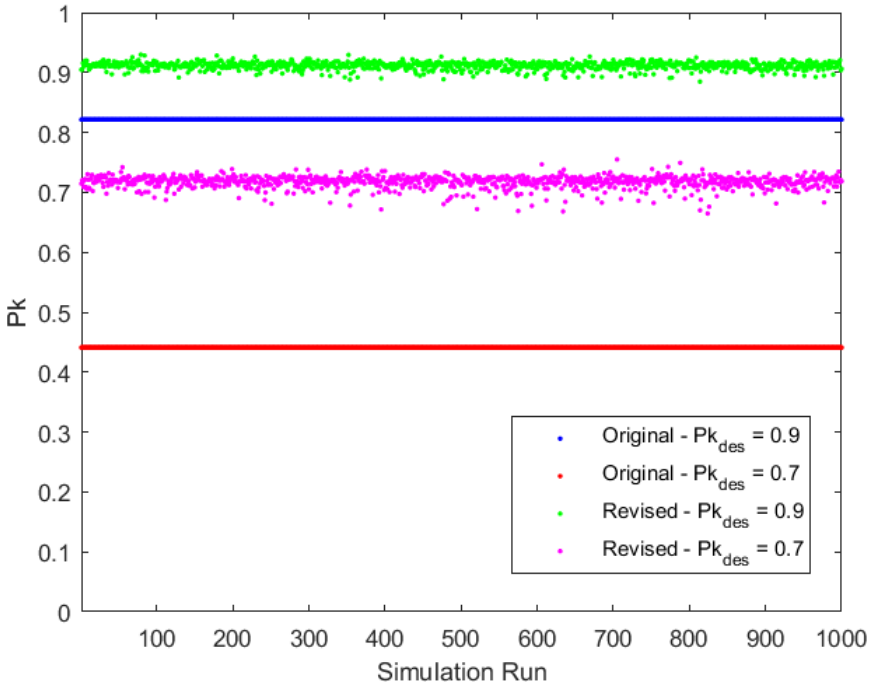

Figure 5.2: Estimated *Pk* values for homogeneous scenario.

Table 5.2: Simulation results for a heterogeneous scenario

	Estimated (Original)		Estimated (Revised)		Empirical	
	$Pk = 0.9$	$Pk = 0.7$	$Pk = 0.9$	$Pk = 0.7$	$Pk = 0.9$	$Pk = 0.7$
One Level	0.8501	0.6307	-	-	0.9690	0.8556
Two Levels	0.8157	0.3724	0.9134	0.7402	0.9814	0.8366

to the fact that the contribution of a single agent to the total Pk diminishes with increasing Pk and so there is less variance with higher estimated Pks. Regardless, the same trend is observed as in the homogeneous case, namely that the higher (and more accurate) Pk estimates show that the heterogeneous weapon set is capable of achieved the desired Pk levels, whereas with the original Pk calculation method it is predicted to be insufficient in almost all cases.

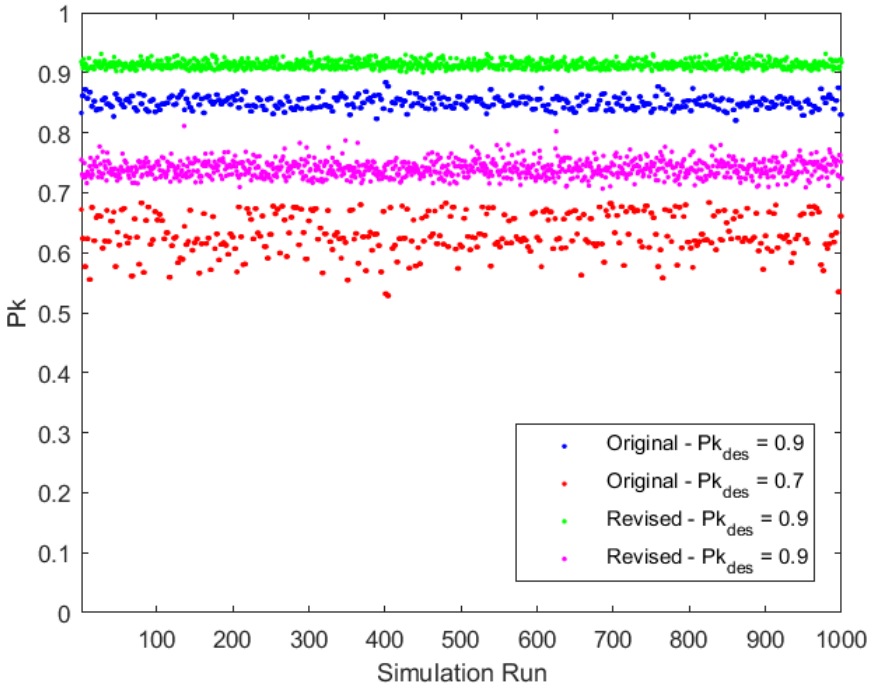

Figure 5.3: Estimated *Pk* values for heterogeneous scenario.

CHAPTER 6
CONCLUSIONS

This thesis presented several contributions to the Weapon Target Assignment literature by investigating new optimal control methods motivated by practical, real-world engagement characteristics. First, three novel cost functions based on the achieved and desired probabilities of kill (Pk) have been proposed for a modified version of the weapon target assignment problem. These cost functions are tailored to induce specific agent targeting behavior that reflects different notions of target priority. In particular, the Sufficiency Threshold (ST) and Enforced Tiering (ET) cost functions attempt to enforce a rigid adherence to target priority which is quantified in terms of desired Pk. A third cost function, the so-called Completion cost, removes the connection between desired Pk and target priority and seeks to simply satisfy the largest number of targets possible. Monte Carlo results show that the ST, ET, and Completion cost functions provide benefit over the traditional weapon target assignment cost in a variety of notional engagement scenarios, particularly at lower weapon target ratios. Overall, these results show that agent behavior in the Weapon Target Assignment problem can be tailored for specific mission objectives through formulation of optimization cost functions.

In the second portion of this thesis, a multi-objective form of the WTA problem is explored in which agents consider both their arrival time relative to other agents, and achieved probability of kill when selecting a target. Simulation studies show that an operator-defined scalar tuning parameter can be used to adjust the relative priority of timing versus probability of kill objectives. It is also shown that the degree to which agents can vary their target closing speeds affects the extent to which the problems are coupled, where a sufficiently large speed range leads to a complete decoupling of the timing and probability of kill optimization problems.

Due to the extreme heterogeneity introduced by the multi-objective cost function, an

implementation of branch and bound that provably solves for the optimal assignment is introduced. It is shown via simulation results that branch and bound consistently outperforms greedy search in heterogeneous cases, though greedy search provides solutions that may be practically useful while being simpler to implement and requiring much less computation.

In the final chapter, it was shown that a team of autonomous agents solving the WTA problem can achieve improved performance by selecting a contingency target. The contingency target is engaged by an agent if its primary target is destroyed by another agent. Achieving the most efficient use of resources requires a model for estimating the achieved probability of kill that accounts for the weapon effectiveness against the contingency target. The benefit of this approach is shown through simulation results in both homogeneous and heterogeneous cases.

There still remain avenues for future work on this modified form of the Weapon Target Assignment problem. For example, the assumption that weapon effectiveness combines as independent probabilities could be removed, and a new model could be considered in which two agents engaging together are more effective than the sum of their effectiveness values. Such a model could be applied to area, rather than point, targets for which a single agent has a negligible Pk and multiple agents are required to engage in order to achieve non-zero Pk_{Σ}.

Another avenue of potential future work is modeling additional sources of uncertainty that would be present in a real world implementation. For example, it is assumed that BDA reports correctly whether or not a given target is destroyed, but perhaps a BDA system can only estimate the probability that the target is destroyed. In that case, rather than assuming spent agents have zero effectiveness, their effectivenesses could be adjusted such that the $Pk_{\Sigma,j}$ incorporates the probability given by the BDA system. Another source of potential uncertainty can be found in more complex communications and attrition models. When agents falsely assume that non-communicative agents have been attrited, they are forced to

underestimate $Pk_{\Sigma,J}$. Better estimates can be achieved by weighing the relative likelihood of various modes of communication failure versus the likelihood of attrition and scaling weapon effectiveness accordingly.

Other potential research questions in this area include developing specific heuristics to reduce the intractable search space created by contingency targeting as discussed in Ch. 5, investigating more detailed models of agent attrition, or considering active targets rather than stationary ones, and investigating the coupling between path planning algorithms, spatially-varying attrition probabilities, and agent assignment algorithms. This last area promises to be a rich problem domain with significant practical application for military planners.

REFERENCES

[1] L. G. Weiss, "Autonomous robots in the fog of war", *IEEE Spectrum*, vol. 48, no. 8, pp. 30 –57, 2011.

[2] E. Ackerman, *Darpa wants swarms of cheap "gremlin" drones*, IEEE Spectrum; Posted online 1 September 2014, Aug. 2015.

[3] J. Hsu, *U.s. navy tests robot boat swarm to overwhelm enemies*, IEEE Spectrum; Posted online 5 October 2014, Oct. 2014.

[4] E. Ackerman, *U.s. army considers replacing thousands of soldiers with robots*, IEEE Spectrum; Posted online 22 January 2014, Jan. 2014.

[5] A. S. Manne, "A target-assignment problem", *Operations Research*, vol. 6, no. 3, pp. 346 –35, 1958.

[6] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem", *Computers & Operations Research*, vol. 24, no. 1, pp. 17–23, 1997.

[7] S. Matlin, "A review of the literature on the missile-allocation problem", *Operations Research*, vol. 18, no. 2, pp. 334–373, 1970.

[8] W. P. Malcolm, "On the character and complexity of certain defensive resource allocation problems", Tech. Rep., 2004.

[9] R. E. Bellman, S. E. Dreyfus, O. A. Gross, and S. M. Johnson, "Missile-allocation problems", *On the Computational Solution of Dynamics Programming Processes*, 1959.

[10] J. C. Bradford, "Determination of optimal assignment of a weapon system to several targets", *AER-EITM-9*, 1961.

[11] D. P. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming", *Operations research*, vol. 51, no. 6, pp. 850–865, 2003.

[12] J. M. Danskin, "The theory of max-min, with applications", *SIAM Journal on Applied Mathematics*, vol. 14, no. 4, pp. 641–664, 1966.

[13] P. A. Hosein, "A class of dynamic nonlinear resource allocation problems", PhD thesis, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION and DECISION SYSTEMS, 1989.

[14] G. Arslan, J. Marden, and J. Shamma, "Autonomous vehicle-target assignment: A game-theoretical formulation", *Journal of Dynamic Systems, Measurement and Control*, vol. 129, no. 5, pp. 584 –596, 2007.

[15] A. C. Chapman, R. A. Micillo, R. Kota, and N. R. Jennings, "Decentralised dynamic task allocation: A practical game: Theoretic approach", in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 915–922.

[16] D. G. Galati, "Game theoretic target assignment strategies in competitive multi-team systems", PhD thesis, University of Pittsburgh, 2004.

[17] E. Gelenbe, S. Timotheou, and D. Nicholson, "Fast distributed near-optimum assignment of assets to tasks", *The Computer Journal*, vol. 53, no. 9, pp. 1360–1369, 2010.

[18] E. Wacholder, "A neural network-based optimization algorithm for the static weapon-target assignment problem", *ORSA Journal on computing*, vol. 1, no. 4, pp. 232–246, 1989.

[19] Z.-J. Lee and W.-L. Lee, "A hybrid search algorithm of ant colony optimization and genetic algorithm applied to weapon-target assignment problems", in *Intelligent data engineering and automated learning*, Springer, 2003, pp. 278–285.

[20] R. Y. Albert, B. B. Thompson, and R. J. Marks, "Competitive evolution of tactical multiswarm dynamics", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 3, pp. 563–569, 2013.

[21] F. Johansson and G. Falkman, "An empirical investigation of the static weapon-target allocation problem", in *Proceedings of the 3rd Skövde Workshop on Information Fusion Topics*, 2009, pp. 63–67.

[22] W. Yanxia, Q. Longjun, G. Zhi, and M. Lifeng, "Weapon target assignment problem satisfying expected damage probabilities based on ant colony algorithm", *Journal of Systems Engineering and Electronics*, vol. 19, no. 5, pp. 939–944, 2008.

[23] E. Witte, R. Chamberlain, and M. Franklin, "Task assignment by parallel simulated annealing", in *Computer Design: VLSI in Computers and Processors, 1990. ICCD'90. Proceedings, 1990 IEEE International Conference on*, IEEE, 1990, pp. 74–77.

[24] P. Chandler and M. Pachter, "Hierarchical control for autonomous teams", in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2001, p. 4149.

[25] Z. Bogdanowicz, N. Coleman, *et al.*, "Sensor-target and weapon-target pairings based on auction algorithm", in *Proceedings of the 11th WSEAS International Conference on APPLIED MATHEMATICS*, World Scientific, Engineering Academy, and Society (WSEAS), 2007, pp. 92–96.

[26] M. A. Şahin and K. Leblebicioğlu, "Approximating the optimal mapping for weapon target assignment by fuzzy reasoning", *Information Sciences*, vol. 255, pp. 30–44, 2014.

[27] A. Tokgöz and S. Bulkan, "Weapon target assignment with combinatorial optimization techniques", *IJARAI) International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 7, pp. 39–50, 2013.

[28] E. Çetin and S. T. Esen, "A weapon–target assignment approach to media allocation", *Applied Mathematics and Computation*, vol. 175, no. 2, pp. 1266–1275, 2006.

[29] G. T. Ross and R. M. Soland, "A branch and bound algorithm for the generalized assignment problem", *Mathematical programming*, vol. 8, no. 1, pp. 91–103, 1975.

[30] S. Göttlich, K. Hameister, and M. Herty, "A novel branch-and-bound algorithm for quadratic mixed-integer problems with quadratic constraints", 2017.

[31] A. Chechetka and K. Sycara, "No-commitment branch and bound search for distributed constraint optimization", in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, ACM, 2006, pp. 1427–1429.

[32] P. A. Hosein, J. T. Walton, and M. Athans, "Dynamic weapon-target assignment problems with vulnerable c2 nodes", DTIC Document, Tech. Rep., 1988.

[33] N. Léchevin, C. Rabbath, and M. Lauzon, *Optimization and Cooperative Control Strategies*. Berlin: Springer, 2014, pp. 47 - 67.

[34] R. Murphey and P. M. Pardalos, *Cooperative control and optimization*. Springer Science & Business Media, 2002, vol. 66.

[35] M. Alighanbari and J. P. How, "Decentralized task assignment for unmanned aerial vehicles", in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, IEEE, 2005, pp. 5668–5673.

[36] J. Sousa, T. Simsek, and P. Varaiya, "Task planning and execution for uav teams", in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, IEEE, vol. 4, 2004, pp. 3804–3810.

[37] M. Ehrgott, "A discussion of scalarization techniques for multiple objective integer programming", *Annals of Operations Research*, vol. 147, no. 1, pp. 343–360, 2006.

[38] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: New insights", *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[39] T. W. McLain, P. R. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of uav rendezvous", in *American Control Conference, 2001. Proceedings of the 2001*, IEEE, vol. 3, 2001, pp. 2309–2314.

[40] T. McLain and R. Beard, "Trajectory planning for coordinated rendezvous of unmanned air vehicles", in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000, p. 4369.

[41] P. Chandler, S. Rasmussen, and M. Pachter, "Uav cooperative path planning", in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2000, pp. 1255–1265.

[42] P. Ogren, M. Egerstedt, and X. Hu, "A control lyapunov function approach to multi-agent coordination", in *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, IEEE, vol. 2, 2001, pp. 1150–1155.

[43] P. Chandler, M. Pachter, S. Rasmussen, and C. Schumacher, "Distributed control for multiple uavs with strongly coupled tasks", in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2003, pp. 11–14.

[44] J. W.C. R. Murphey, "Simultaneous area search and task assignment for a team of cooperative agents", 2003.

[45] Y. Eun and H. Bang, "Cooperative task assignment/path planning of multiple unmanned aerial vehicles using genetic algorithms", *Journal of aircraft*, vol. 46, no. 1, p. 338, 2009.

[46] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating uavs", in *Cooperative Control: Models, Applications and Algorithms*, Springer, 2003, pp. 23–41.

[47] F. A. Glenn and J. M. Bennett, "Decision aiding concepts for air strike planning.", ANALYTICS INC WILLOW GROVE PA, Tech. Rep., 1980.

[48] P. R. Weaver, "Development and evaluation of an automated decision aid for rapid re-tasking of air strike assets in response to time sensitive targets", PhD thesis, Monterey, California. Naval Postgraduate School, 2004.

[49] B. Zacherl, "Weapon-target pairing; revising an air tasking order in real-time", NAVAL POSTGRADUATE SCHOOL MONTEREY CA, Tech. Rep., 2006.

[50] J. M. Rosenberger, H. S. Hwang, R. P. Pallerla, A. Yucel, R. L. Wilson, and E. G. Brungardt, "The generalized weapon target assignment problem", TEXAS UNIV AT ARLINGTON, Tech. Rep., 2005.

[51] D. A. Bader, W. E. Hart, and C. A. Phillips, "Parallel algorithm design for branch and bound", in *Tutorials on Emerging Methodologies and Applications in Operations Research*, Springer, 2005, pp. 5–1.

[52] M. Tavana, M. D. Bailey, and T. E. Busch, "A multi-criteria vehicle-target allocation assessment model for network-centric joint air operations", *International Journal of Operational Research*, vol. 3, no. 3, pp. 235–254, 2008.

[53] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Mineola, NY: Dover Publications, 1998, Chs. 12, 17, 18, ISBN: 40258409.

[54] G. G. denBroeder Jr, R. Ellision, and L. Emerling, "On optimum target assignments", *Operations Research*, vol. 7, pp. 322 –326, 1959.

[55] S. E. Kolitz, "Analysis of a maximum marginal return assignment algorithm", in *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, IEEE, 1988, pp. 2431–2436.

[56] S. Kirkpatrick, J. C. D. Gellat, and M. P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, no. 4598, pp. 671 –680, 1983.

[57] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation", *Artificial Intelligence*, vol. 101, no. 1, pp. 165–200, 1998.

[58] T. Sikanen, "Solving weapon target assignment problem with dynamic programming", Technical report, Mat-2.4108 Independent research projects in applied mathematics, Tech. Rep., 2008.

[59] A. C. Chapman, A. Rogers, and N. R. Jennings, "Benchmarking hybrid algorithms for distributed constraint optimisation games", *Autonomous Agents and Multi-Agent Systems*, vol. 22, no. 3, pp. 385–414, 2011.

[60] R. Ahuja, A. Kumar, K. Jha, and J. Orlin, "Exact and heuristic algorithms for the weapon-target assignment problem", *Operations Research*, vol. 55, no. 6, pp. 1136 –1146, 2007.

[61] O. Kwon, K. Lee, D. Kang, and S. Park, "A branch-and-price algorithm for a targeting problem", *Naval Research Logistics (NRL)*, vol. 54, no. 7, pp. 732–741, 2007.

[62] R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search", *Artificial intelligence*, vol. 27, no. 1, pp. 97–109, 1985.

[63] S. Russell, P. Norvig, and A. Intelligence, "A modern approach", *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, p. 1048, 2010.

[64] K. Volle, J. Rogers, and K. Brink, "Decentralized cooperative control methods for the modified weapon–target assignment problem", *Journal of Guidance, Control, and Dynamics*, pp. 1934–1948, 2016.

[65] T. H. Cormen, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2009, p. 416, ISBN: 9780262032933.

[66] K. H. Rosen, *Handbook of discrete and combinatorial mathematics*. CRC press, 1999.