NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

PATH OPTIMIZATION FOR SINGLE AND MULTIPLE SEARCHERS: MODELS AND ALGORITHMS

by

Hiroyuki Sato

September 2008

Dissertation Supervisor: Johannes O. Royset

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2008	Dissertation

4. TITLE AND SUBTITLE Path Optimization for Single and Multiple Searchers: Models and Algorithms	5. FUNDING NUMBERS
6. AUTHORS Hiroyuki Sato	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	

13. ABSTRACT(*maximum 200 words*)

We develop models and solution methodologies to solve the discrete-time path-optimization problem where a single or multiple searchers look for a moving target in a finite set of cells. The single searcher is constrained by maximum limits on the consumption of several resources such as time, fuel, and risk along any path. We develop a specialized branch-an-bound algorithm for this problem that utilizes several new network reduction procedures as well as new bounding technique based on Lagrangian relaxation and network expansion. The resulting algorithm is quite efficient and promising. For the multiple searchers, an optimal set of paths (search plan) is determined by taking advantage of the cooperative search effect. We present a new exact algorithm and two new heuristics to find an optimal or near-optimal search plan. One of the heuristics is based on the cross-entropy method and is found to perform well for a broad range of problem instances. The exact algorithm deals with the specific case of homogeneous searchers and is based on outer approximations by several new cutting planes. In addition, we prove that under certain assumptions the path-optimization problem becomes equivalent to a large-scale linear mixed-integer program.

14. SUBJECT TERMS Keywords Path optimization, Branch-and-bound, Cross-entropy method, Outer approximation	15. NUMBER OF PAGES 141
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

PATH OPTIMIZATION FOR SINGLE AND MULTIPLE SEARCHERS: MODELS AND ALGORITHMS

Hiroyuki Sato
Technical Official Third Grade, Japan Ministry of Defense
B.S., Kyusyu University, 1991
M.S., Naval Postgraduate School, 2005

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

from the
NAVAL POSTGRADUATE SCHOOL
September 2008

Author:

_____
Hiroyuki Sato

Approved by:

Johannes O. Royset Assistant Professor of Operations Research Dissertation Supervisor	R. Kevin Wood Professor of Operations Research Committee Chairman
James N. Eagle Professor of Operations Research	Moshe Kress Professor of Operations Research
Isaac I. Kaminer Professor of Mechanical and Astronautical Engineering	

Approved by:

_____
James Eagle, Chairman, Department of Operations Research

Approved by:

_____
Doug Moses, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

We develop models and solution methodologies to solve the discrete-time path-optimization problem where a single or multiple searchers look for a moving target in a finite set of cells. The single searcher is constrained by maximum limits on the consumption of several resources such as time, fuel, and risk along any path. We develop a specialized branch-an-bound algorithm for this problem that utilizes several new network reduction procedures as well as new bounding technique based on Lagrangian relaxation and network expansion. The resulting algorithm is quite efficient and promising. For the multiple searchers, an optimal set of paths (search plan) is determined by taking advantage of the cooperative search effect. We present a new exact algorithm and two new heuristics to find an optimal or near-optimal search plan. One of the heuristics is based on the cross-entropy method and is found to perform well for a broad range of problem instances. The exact algorithm deals with the specific case of homogeneous searchers and is based on outer approximations by several new cutting planes. In addition, we prove that under certain assumptions the path-optimization problem becomes equivalent to a large-scale linear mixed-integer program.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1

A. MOTIVATION 1

B. SCOPE OF DISSERTATION 2

C. LITERATURE SURVEY 3

1. Resource-constrained Single Searcher Problem 4

2. Multiple-Searcher Problem 5

D. CONTRIBUTIONS 6

E. ORGANIZATION 7

II. RESOURCE-CONSTRAINED SINGLE SEARCHER PROBLEM 9

A. PROBLEM DESCRIPTION AND FORMULATION 9

B. BRANCH-AND-BOUND ALGORITHM FOR SSP 13

1. Existing Algorithm 14

2. Algorithmic Modifications for SSP 18

C. BRANCH-AND-BOUND ALGORITHM FOR RSSP 35

1. Lagrangian Static Bound 35

2. Network Reduction 38

3. Algorithm 41

D. NUMERICAL EXAMPLE 44

III. MULTIPLE-SEARCHER PROBLEM 53

A. PROBLEM DESCRIPTION AND FORMULATION 53

B. ALGORITHMS FOR MSP 55

1. Branch-and-Bound Algorithm 57

2. Static Bound Heuristic 62

3. Cross-Entropy Method 64

IV. MULTIPLE HOMOGENEOUS SEARCHER PROBLEM . . . 77

A. PROBLEM DESCRIPTION AND FORMULATION 77

B. ALGORITHMS FOR MHSP 80
1. Basic OA Algorithm 80
2. New Cuts for OA Algorithm 83
3. Numerical Study of Three Searchers 106
4. Application with Large Number of Searchers 107
V. CONCLUSIONS AND FUTURE RESEARCH 113
A. CONCLUSIONS 113
B. FUTURE RESEARCH 114
LIST OF REFERENCES 115
INITIAL DISTRIBUTION LIST 121

LIST OF FIGURES

1. A discretized area of interest composing of $C = 4$ cells. 10
2. A discretized airspace over the area of interest (Figure 1) with $H = 2$ altitudes. 11
3. A time-expanded graph from the network in Figure 1 and one altitude. The searcher's initial position is $n_0 = \langle v_0, 0\rangle = \langle\langle 1, 1\rangle, 0\rangle$ and final position is $\hat{n} = \langle \hat{v}, T + 1\rangle$. 17
4. A node-and-time expanded network from the time-expanded graph (Figure 3). 25
5. An area of interest composed of 5 by 5 cells. For each time period, the unshaded, lightly-shaded, heavily-shaded, and completely-shaded cells describe the regions where neither searcher nor target stay, only target possibly stays, only searcher possibly stays, and target and searcher possibly stay, respectively. 30
6. An area of interest composed of 10 by 10 cells and two altitudes. Heavily shaded cells ($\mathcal{C}_1$), lightly shaded cells ($\mathcal{C}_2$), and unshaded cells ($\mathcal{C}_3$) describe risky, moderately risky, and non-risky area, respectively. The circle indicates the cell over which searcher starts, and the triangle specifies the initial position of the target. 46
7. An optimal path for survival probability limit 0.90 and fuel limit 400. The solid lines and the dashed lines represent flight segments at low and high altitude, respectively. (See Figure 6 for a description of the underlying figure.) 50
8. An optimal path for a case with edge-dependent glimpse probability, survival probability limit 0.90, and fuel limit 400. The solid lines and the dashed lines represent flight segments at low and high altitude, respectively. (See Figure 6 for a description of the underlying figure.) . 51

9. An optimal search plan for the 5 by 5 cell search problem with 3 searchers and the time horizon 9. 63
10. Linear supporting functions. 81
11. Strong cut and tangent hyper-plane on an exponential objective function in mixed-integer program. 89
12. Piece-wise linearlization of the exponential function $f_\omega(W_\omega)$. 95
13. Illustration of cut generation around the best solution. 100
14. Symmetric environment where the symmetric-cut is available. 104

LIST OF TABLES

1. Run times and number of branching attempts (counted in Step 4) for Algorithm 1 with static and dynamic bounds on 11 by 11 cell search problem with time horizon $T = 17$. Columns labeled "Dynamic Bound [32]" correspond to original and speed-adjusted results from [32]. 22
2. Run time and number of branching attempts (counted in Step 4) for Algorithm 1 on problem instances of Table 1 using directional static bound (D-Static) and for Algorithm 2 using static bound and network reduction (Static & Red.) and directional static bound and network reduction (D-Static & Red.). 29
3. Run times and number of branching attempts for Algorithm 2, with directional static bound and network reduction, compared with Algorithm 1 with dynamic bound on 15 by 15 cell search problem with time horizon $T = 20$. 34
4. Glimpse detection probability $g(v, v', t)$ and survival probability $\sigma(v, v')$ for different cells and altitude. 47
5. Computational results (probability of detection, survival probability, fuel consumption and run times) for Algorithm 3 with survival probability limit = 0.95 and 0.90. 48
6. Computational results (probability of detection, survival probability, fuel consumption and run times) for Algorithm 3 with survival probability limit = 0.85 and 0.80. 49
7. Computational results (probability of detection, survival probability, fuel consumption and run times) for Algorithm 3 with survival probability limit = 0.75 and 0.70. 49

8. Run times for the branch-and-bound algorithm (B/B) and the static bound heuristic (SBH) on 5 by 5 cell search problem with 1-3 searchers and time horizon T = 5-10. 61
9. Probability of detection obtained by the branch-and-bound algorithm (B/B) and the static bound heuristic (SBH) on 5 by 5 cell search problem with 1-3 searchers and time horizons T = 5-10, and relative gap between SBH and B/B solutions. 62
10. Test cases with Algorithm 6, Algorithm 7 and Algorithm 7N with area size, time horizon, number of searchers, and minimum sample size. m is the number of arcs in the network used in the corresponding algorithm. 71
11. Probability of detection for problem instances with a single searcher. . 72
12. Run times on problem instances with a single searcher. 73
13. Probability of detection for problem instances with two searchers. . . . 73
14. Run times on problem instances with two searchers. 74
15. Probability of detection for problem instances with three searchers. . . 75
16. Run times on problem instances with three searchers. 75
17. Numerical results for Algorithm 8 versions 1 and 2 for every ten minutes of calculations. The best known non-detection probability is 0.563472 (=1-0.436528) found in Table 15 (case C3) in Chapter III. 83
18. Numerical results for Algorithm 9 using multi-t2-cut (versions 1 and 2) for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17. 87
19. Numerical results for Algorithm 9 using multi-t3-cut (versions 3 and 4) for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17. 88
20. Numerical results for Algorithms 10 and 11 using strong-cut for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17. 93

21. Numerical results for Algorithms 10.2 and 11.2 using an improved initial solution for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17. 99
22. Numerical results for Algorithms 12 and 13 using relative-cut for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17. 102
23. Numerical results for Algorithms 14 using symmetric-cut for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17. 105
24. Relative optimality gaps for Algorithm 13 applied three searcher problem instances during two hours of calculations. 107
25. Numerical results for Algorithms 13 and 15 for the case with 5 searchers for every 10 minutes of calculations. 110
26. Numerical results for Algorithms 13 and 15 for the case with 10 searchers for every 10 minutes of calculations. 110
27. Numerical results for Algorithms 13 and 15 for the case with 15 searchers for every 10 minutes of calculations. 110
28. Numerical results for Algorithms 13 and 15 for the case with 30 searchers for every 10 minutes of calculations. 111

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would sincerely like to thank my advisor, Dr. Johannes Royset, for his expert guidance during this study. His insight and forward thinking were instrumental for me to stay on the right truck.

I would also like to thank Dr. Kevin Wood, Dr. James Eagle, Dr. Moshe Kress, and Dr. Isaac Kaminer for their time and willingness to serve on my Ph.D. committee.

My fellow Ph.D. students, CDR. David Ruth, LT. Ahmed Al Rowaei, and Mr. Eng Yau Pee, relaxed, encouraged, and supported me as I struggled to prepare for my qualifying exams.

Finally, I would like to thank the Maritime Staff Office of the Japan Ministry of Defense for giving me a wonderful opportunity to study in NPS.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

This dissertation develops models and solution methodologies to solve the discrete time-and-space path-optimization problems where a single searcher or multiple searchers look for a non-evading moving target. We extended the single searcher problem (SSP), generally called the optimal searcher path problem, to realistic situation where: (1) the searcher needs to change its flight altitude during a mission to balance sensor quality and risk from ground threats; (2) the searcher is subject to constraints on multiple resources such as fuel consumption and risk exposure along any path; and (3) the search effect depends on the current and previous locations of the searcher. The latter situation may occur if moving from some search sector to a new sector results in a lower search effect due to ineffective search during transit. We refer to this extended SSP as the resource-constrained single searcher problem (RSSP).

We present new bounding techniques (static bound and directional static bound) and network reduction procedure in a branch-and-bound algorithm to optimally solve the single searcher problem (SSP). The static bound simplifies the bound calculation substantially at the expense of a weaker bound. The directional static bound improves the weaker static bound. The network reduction procedure takes advantage of the initial positions of searcher and target, and eliminates parts of the branch-and-bound tree while retaining a optimal solution. The resulting algorithm solves a problem instance with 15 by 15 cells and a time horizon of 20 optimally in less than 4 seconds on average, which is a significant reduction from the 9 minutes required by a state-of-the-art algorithm.

We extend our branch-and-bound algorithm for the SSP to treat the RSSP and construct a new bounding technique based on Lagrangian relaxation. In addition, we develop novel network reduction techniques using dominance tests. The resulting algorithm solves a realistic problem instance (10 by 10 cells, two altitudes, time

horizon 40, and two resource constraints), on average, in less than 10 minutes.

We extend the specialized branch-and-bound algorithm to the case of multiple searchers and also develop two new heuristics (static bound heuristic and cross-entropy heuristic). In the context of search problems, this dissertation appears to be the first one which utilizes the cross-entropy method. Among these three algorithms, the cross-entropy heuristic performs best for a broad range of problem instances. While the branch-and-bound algorithm and the static bound heuristic are limited to small problem instances, the cross-entropy heuristic obtains good approximate solutions of moderately sized instances (15 by 15 cells, time horizon 18, and three searchers) in about 10 minutes.

For the case with multiple identical searchers, we construct an exact algorithm based on outer approximations by cutting planes. We introduce several new cutting planes (multiple-cut, strong-cut, relative-cut, and symmetric-cut), combine them, and develop a specialized outer approximation algorithm. For a moderate-size problem instance (15 by 15 cells, time horizon 16, and three searchers), the resulting algorithm essentially provides an approximate solution that is guaranteed to be within 5% of an optimal solution in less than about 20 minutes. Instances with more searchers (such as 10, 15 and 30 searchers) are solved even faster. In addition, we prove that under certain assumptions the nonlinear convex multiple homogeneous searcher problem is equivalent to a large-scale linear mixed-integer program.

I. INTRODUCTION

A. MOTIVATION

The need to search for moving objects arises in many civilian and military applications. Rescue teams search for lost persons and shipwrecks. Autonomous robots could find victims and survivors after natural disasters. In military search operations, patrol aircraft and unmanned aerial vehicles (UAVs) look for high-valued targets such as missile launchers and terrorist vehicles. In all these applications, the choice of paths for the searchers strongly influences the probability of finding the targets within a specific time limit. Unfortunately, the problem of selecting the "best" path is fundamentally hard due to the nonlinearity induced by the probability of detection. For example, looking twice by a searcher generally does not double the detection probability.

In military search missions with UAVs, the problem of finding good paths for searchers (UAVs) is complicated even further. Rather than manned aircraft, UAVs are preferred to operate in dangerous environments such as hostile regions and battlefields. In these applications, the searchers (UAVs) often need to balance search effectiveness with threats to themselves posed by surface-to-air missiles and small-arms fire. Moreover, the search problem becomes significantly more difficult as the number of searchers grows. With the technological advances in cooperative control of multiple UAVs, such multi-searcher problems are appearing with increasing frequency in applications. The need for optimal path planning in these complicated situations motivates this dissertation.

In this dissertation, we consider several search problems including situations with searchers subject to threats as well as multiple searchers. We particularly focus on search by aerial assets that are subject to threats from the ground. We formulate search problems for single and multiple searchers and develop solution methodologies for efficiently finding an optimal or near-optimal path.

B. SCOPE OF DISSERTATION

We consider discrete time-and-space path-optimization problems where a non-evading target moves in a two-dimensional area of interest (AOI). The AOI is discretized into a finite set of cells and the target moves between the cells in discrete time. Stone [47] and Brown [7] introduced the basic forms of this search problem. A single searcher or multiple searchers move, in discrete time, through a discretized two-dimensional space (on the ground in the AOI or at a constant altitude over the AOI) or a discretized three-dimensional airspace over the AOI. The searchers' space is represented by a directed graph, where vertices represent waypoints (or search sectors) from which a searcher can search a particular cell in the AOI and where directed edges represent transition between two waypoints. Thus a path of the searcher is represented as a sequence of waypoints (vertices).

We note that our aim is *not* to find an optimal flyable trajectory that fully accounts for turn radius, climb/dive angle, and max/min speed for the searcher. We aim for "high-level" control of the searchers where waypoints may represent points in space that are many minutes apart in flying time and may represent areas to be searched for some period of time. We consider imperfect searchers that may search a cell that contains a target without finding the target. However, in this dissertation, we assume that the searchers will not report a target in a cell where there is no target.

The goal for the searchers is to maximize the probability of detecting the target within a finite mission time or, equivalently, to minimize the probability that the target is not detected during the duration of the mission. The searchers are subject to constraints on path continuity, mission time, and possibly other "resources" such as risk exposure and fuel consumption.

This dissertation refers to the path optimization problem for a single searcher with path- and time-constraints as the single searcher problem (SSP). Other authors refer to this problem as the optimal searcher path problem [41, 49, 32]. In this dissertation, we focus on the following variants of SSP.

1. SSP with additional constraints on, e.g., risk exposure to threats and fuel consumption during the missions. We refer to this path-optimization problem as the resource-constrained single searcher problem (RSSP).
2. SSP for multiple searchers where the probability of detecting the target by at least one searcher is maximized. We term this problem the multiple-searcher problem (MSP).
3. MSP for homogeneous searchers where the fact that searchers are identical is utilized. We refer to this multiple-homogeneous-searcher problem as MHSP.

This dissertation always assumes that the searchers are subject to constraints on path continuity and mission time. We note that the multiple-searcher problem with resource-constraints (risk exposure and fuel consumption, etc.) is outside of the scope of this dissertation.

C. LITERATURE SURVEY

The topic of searching for a moving target in discrete time and space where a searcher's path is constrained has received much attention. Benkoski et al. [41] provide a comprehensive survey of this problem (until 1990). The path- and time-constrained search problem with a stationary target is NP-complete [49]. Thus the path- and time-constrained, moving-target, search problem is at least as difficult.

One scheme to find an optimal path for a searcher is the branch-and-bound procedure. A branch-and-bound procedure was first studied by Stewart [46]. Since Stewart's "bounds" are not valid, an optimal branch of the enumeration tree may be fathomed. More recent studies [18, 34, 52, 32] focus on the development of specialized branch-and-bound algorithms for finding an optimal path for the single searcher. In these algorithms, a path is a sequence of vertices that the searcher will visit and branching corresponds to extending a subpath by one more vertex. Bounds on the optimal value of the problem are obtained by replacing the probability of detection with, effectively, the expected number of detections [34, 52, 32]. Bounds are also obtained by assuming that the searcher can divide its effort among multiple cells each time period [18]. The efficiency of various branch-and-bound methods is investigated

in [52], with emphasis on the tradeoff between the accuracy of the bound employed and the time required to compute it.

An alternative approach to obtain an optimal path is based on dynamic programming and partially observable Markov decision processes [17]. The approach can solve small problems quickly, but requires a large amount of computer storage as the problem size increases. Later the same author reports that a branch-and-bound algorithm [18] performs more efficiently and can solve larger problems than the existing dynamic programming procedures. Thus, we mainly focus on branch-and-bound procedures in this dissertation.

1. Resource-constrained Single Searcher Problem

In the studies listed above, it is assumed that the searcher moves on the ground in the AOI or searches the AOI from a constant altitude. However, in military search, surveillance, and reconnaissance operations over land, factors such as risk and fuel consumption become independent elements of concern for planners [37], and the searcher is required to change its flight altitude during a mission. Risk to the searcher arises from exposure to enemy threats on the ground such as small-arms fire, anti-aircraft artillery, and surface-to-air missiles. Risk of pilot error (e.g., for a low-flying helicopter [29]) and mechanical failure (e.g., for small and unreliable UAVs [31]) may also be significant. Fuel consumption tends to be proportional to the search duration. However, for small UAVs, it is not always proportional to the mission time, due to variation in the searcher's speed and/or altitude, as well as varying weather conditions. During search over land, terrain features require altitude changes. The altitude is also varied to balance the searcher's risk with image quality. We note that image quality is of particular concern for small UAVs operating with low- to moderate-quality sensors (For information about UAV surveillance operations and sensors, see [37, 42]). In civilian search and surveillance operations over land, we may face many of the same factors with the exception of ground fire.

There are only a few studies dealing with constraints on risk and fuel consumption for the searchers. Thomas and Eagle [48] consider a search problem with a random time horizon; i.e., with a perishable target with a random lifetime that follows a particular distribution (a geometric distribution in that study). This situation is similar to having the searcher being subject to threats and having the search terminate after searcher "failure." We will consider a similar situation, but will limit the total risk a searcher can be exposed to during the mission. Secrest [43] considers optimal path planning with multiple-objectives (simultaneously accounting for flight time, risk and fuel consumption, etc.) in surveillance missions while visiting all assigned locations. The resulting model does not consider the probability of detecting a target.

Models for military aircraft routing frequently consider minimum risk along a path subject to fuel consumption and mission duration constraints, see, e.g., [35, 55, 10]. These models have similar constraints to the ones in RSSP, but deal with linear and time-invariant objective functions. In contrast, RSSP has a nonlinear, time-variant objective function representing the probability of detection (see Chapter II). Only recently have researchers extended the models for aircraft routing to situations with nonlinear objective functions, and then only for special cases of path-dependent risks [29].

2. Multiple-Searcher Problem

The branch-and-bound procedure is also applicable in solving the multiple-searcher problems (MSP/MHSP). Dell et al. [13] present an exact procedure (branch-and-bound algorithm) and six heuristics (local search, expected detection heuristics, genetic algorithms, and moving time-horizon heuristic) to solve the MSP/MHSP. However, the branch-and-bound algorithms need prohibitive runtime to guarantee an optimal path for multiple-searchers, because the number of possible paths grows exponentially with the number of searchers. This motivates the development of heuristic algorithms. We find other heuristic algorithms based on myopic and receding-horizon

approaches in [23] and sequential optimization of each searcher in [26]. For a specific scenario, Riehl et al. [28] present a receding-horizon approach that jointly optimizes paths and sensor orientations for multiple searchers. For a stationary-target case, Song et al. [45] use a forward induction approach to find an optimal search strategy.

Control and robotic communities have analyzed the search problems for multiple UAVs and sensors as well. Ryan et al. [2] review the current issues and developments in the field of cooperative control (real-time navigation, collision avoidance, and flight formation, etc.). Decentralized search techniques [20, 54] and Bayesian heuristic approaches [53, 25] are also studied in the context of real-time search operations.

D. CONTRIBUTIONS

This study considers the path optimization problems, including "searcher subject to threat" and cases with multiple searchers. We extend the classical single searcher problem (SSP) to realistic situations such as: (1) the searcher needs to change its flight altitude during a mission to balance sensor quality and risk from the ground treats; (2) the searcher is subject to constraints on multiple resources such as fuel consumption and risk exposure along its path; and (3) the search effect depends on the current and previous locations of the searcher. The latter situation may occur if moving from some search sector to a new sector results in a lower search effect due to ineffective search during transit.

We present new bounding techniques and network reduction procedures in a branch-and-bound algorithm to efficiently solve the SSP. Based on this algorithm, we develop a specialized branch-and-bound procedure to solve the resource-constrained single searcher problem (RSSP) by utilizing several novel network reduction procedures as well as new bounding technique taking account of the multiple resource constraints. We also develop several new algorithms to solve the multiple searchers problems (MSP/MHSP).

E. ORGANIZATION

The reminder of this dissertation is outlined as follows. In the next chapter, we formulate RSSP and develop a specialized branch-and-bound algorithm for RSSP. The resulting algorithm utilizes a new bounding technique, applies several network reduction procedures, and exhibits promising behavior in computational tests on instances of SSP and RSSP. In Chapter III, we formulate MSP by extending SSP and present three new algorithms (an exact procedure and two heuristics) to solve MSP and examine their computational efficiency. In Chapter IV, we focus on MHSP. We use the convexity of the nonlinear objective function (the non-detection probability) and construct an exact algorithm using cutting planes (outer approximations). We prove that under certain assumptions the problem is equivalent to a large-scale linear mixed-integer program. We also introduce several new cuts for the MHSP and demonstrate their effect in computational tests. Chapter V provides conclusions of our study as well as suggestions for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RESOURCE-CONSTRAINED SINGLE SEARCHER PROBLEM

This chapter formulates the resource-constrained single searcher problem (RSSP) by generalizing existing models of the single searcher problem (SSP) to the case with multiple resource constraints. The RSSP also considers an altitude-dependent searcher with its search effect depending on not only the searcher's current location but also its previous location. We combine the branch-and-bound methodology for solving SSP with the line of research on the resource-constrained, shortest-path problem [10]. Specifically, we merge the algorithms in [32] and [10], and develop a specialized branch-and-bound algorithm for RSSP. The resulting algorithm utilizes new bounding techniques, applies several new network reduction procedures, and exhibits promising behavior in computational tests on instances of SSP and RSSP.

A. PROBLEM DESCRIPTION AND FORMULATION

The area of interest (AOI) is discretized into a finite set of cells $\mathcal{C} = \{1, \ldots, C\}$ (see Figure 1) and the time horizon is discretized into a finite set of time periods $\mathcal{T} = \{1, 2, ..., T\}$. A target occupies one cell each time period and moves among cells according to a Markov process with known transition matrix $\Gamma = (\Gamma_{c,c'})$, where $\Gamma_{c,c'}$ is the probability that the target moves from cell c to cell c' during one time period. This Markovian target motion is first modeled in Stone [47]. Let $p(\cdot, t) = [p(1,t), p(2,t), \ldots, p(C,t)]$, where $p(c,t)$ is the probability that the target is in cell $c \in \mathcal{C}$ at the beginning of time period $t \in \mathcal{T}$ and the target has not been detected before t. We refer to $p(\cdot, t)$ as the *undetected target distribution*. We note that $p(\cdot, t)$ may not be a probability mass function as the sum of the elements could be less than 1. The initial target distribution $p(\cdot, 1)$ is known.

A single searcher moves through a designated airspace over the area of interest with the goal of finding the moving target on the ground. The airspace over each

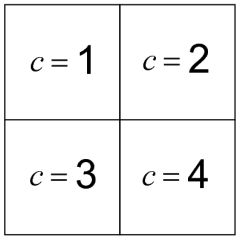

Figure 1. A discretized area of interest composing of $C = 4$ cells.

cell $c \in \mathcal{C}$ is vertically discretized into a set of altitudes (or boxes) $\mathcal{H} = \{1, 2, \ldots, H\}$ (see Figure 2). For any $c \in \mathcal{C}$ and $h \in \mathcal{H}$, we refer to the cell-altitude pair $\langle c, h \rangle$ as a *waypoint* where the searcher can loiter and carry out search of cell c. We assume that, from waypoint $\langle c, h \rangle$, the searcher can sweep cell c only. We model the designated airspace by a directed network $(\mathcal{V}, \mathcal{E})$, with set of vertices $\mathcal{V}$ and set of directed edges $\mathcal{E}$, in which vertices $v = \langle c, h \rangle \in \mathcal{V}$ represent waypoints and directed edges $e = (v, v') \in \mathcal{E}$ represent transition between waypoints $v, v' \in \mathcal{V}$. The waypoints $v, v' \in \mathcal{V}$ are *adjacent* to each other if v and v' touch at any portions. The searcher can only transit between two waypoints that are adjacent to each other. Let $\mathcal{F}(v) \subset \mathcal{V}$ be the set of vertices that are adjacent to $v \in \mathcal{V}$. We refer to $\mathcal{F}(v)$ as the forward star of vertex v. We adopt the convention that $v \in \mathcal{F}(v)$ for all $v \in \mathcal{V}$. Then, the set of edges $\mathcal{E} = \{(v, v') \in \mathcal{V} \times \mathcal{V} |\ v' \in \mathcal{F}(v)\}$.

During each time period $t \in \mathcal{T}$, the searcher is at a particular vertex (waypoint). We assume there is *no* transit time between waypoints. Hence, $(v, v') \in \mathcal{E}$ simply represents search at waypoint v followed by search at waypoint v' in the next time period. The situation with nonzero transit time between waypoints can be modeled, at least approximately, by introducing artificial vertices. (We refer to [32] for a comprehensive study of nonzero transit times.) We note that the edge $(v, v) \in \mathcal{E}$ represents searching at waypoint v for two consecutive time periods.

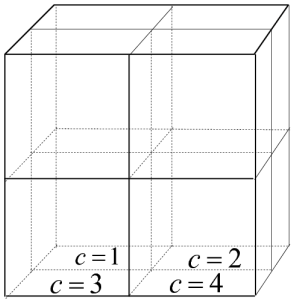

Figure 2. A discretized airspace over the area of interest (Figure 1) with $H = 2$ altitudes.

Let $\phi : \mathcal{V} \to \mathcal{C}$ be the function that specifies the cell over which a vertex is located, i.e., cell $\phi(v)$ is searched from vertex v. We denote the searcher's vertex (waypoint) prior to time period 1 by $v_0 \in \mathcal{V}$. This starting vertex could be a designated entry waypoint into the area of interest. We also define $\hat{\mathcal{V}} \subset \mathcal{V}$ to be a set of possible destination vertices for the searcher (e.g., $\hat{\mathcal{V}} = \{v_0\}$ if the searcher is required to return to the starting vertex or $\hat{\mathcal{V}} = \mathcal{V}$ if the search can end anywhere).

For any $k \in \mathcal{T}$ and $v_l \in \mathcal{V}$, $l = 0, 1, 2, ..., k$, such that $(v_{l-1}, v_l) \in \mathcal{E}$ for all $l = 1, 2, ..., k$, let the sequence $\{v_l\}_{l=0}^{k}$ denote a directed v_0-v_k subpath. If $v_k \in \hat{\mathcal{V}}$, then the directed v_0-v_k subpath is a directed v_0-v_k path. When the meaning is clear from the context, we refer to a directed v_0-v_k (sub)path as a (sub)path. In this notation, the searcher flies from v_0 to some $v_k \in \hat{\mathcal{V}}$ along a directed v_0-v_k path. The searcher occupies only one vertex $v \in \mathcal{V}$ each time period, and stays at the same vertex or moves to another vertex in $\mathcal{F}(v)$ for the next time period.

We adopt the following target-detection model. If the target is in cell $c \in \mathcal{C}$ during time period $t \in \mathcal{T}$ and the searcher is at the same time at waypoint $v' \in \mathcal{V}$ above cell c, i.e., $\phi(v') = c$, then detection occurs with a probability. We term this probability as *glimpse detection probability* and refer to $g(v, v', t)$, where $v \in \mathcal{V}$ is the searcher's waypoint during time period $t-1$. Hence, the glimpse detection probability

during time period t depends on the previous and current waypoints for the searcher. This is a generalization of earlier models where the glimpse detection probability depends only on v' and t [52, 32] and is one of our contributions. Our model accounts for the fact that moving from some waypoint to a new waypoint may result in a lower glimpse detection probability than if the searcher already was loitering at the latter waypoint, i.e., $g(v', v', t) > g(v, v', t)$ for $v' \neq v$. This effect occurs if refocusing the sensor and becoming familiar with a new cell have a significant detrimental effect on the glimpse detection probability. In general, change of waypoint, especially change of altitude and frequent, irregular change of direction, may distract from the search. This generalization also allows us to account indirectly for small transit times (much less than the length of a time period) between waypoints without adopting a fine time discretization with resulting high computational cost. In this notation, the probability of detection at waypoint v' during time period t, given search at waypoint v during time period $t-1$, and no prior detections becomes $p(\phi(v'), t)g(v, v', t)$.

The glimpse detection probability may also depend on the searcher's speed. To account for this situation, edges can be duplicated as in [10] to represent search at different speeds in cases where speed influences the searcher's effectiveness significantly. For the sake of simplicity, however, we do not introduce notation to handle that situation.

Since $p(\cdot, t)$ is the undetected target distribution at the beginning of time period t, it depends on searches up to time period $t-1$. Specifically, if $p(\cdot, t) = [p(1, t), \ldots, p(c', t), \ldots, p(C, t)]$ and cell c' is searched from waypoint v' (i.e., $\phi(v') = c'$) during time period t, the undetected target distribution at the beginning of the next time period $t+1$ is

$$p(\cdot, t+1) = [p(1, t), .., p(c'-1, t), p(c', t)(1 - g(v, v', t)), p(c'+1, t), .., p(C, t)]\Gamma, \quad \text{(II.1)}$$

where v is the searcher's vertex during time period $t-1$. Thus, the probability of

detection along a directed v_0-v_k path $\mathcal{P} = \{v_l\}_{l=0}^{k}$, denoted $q(\mathcal{P})$, is defined as

$$q(\mathcal{P}) = \sum_{t=1}^{k} p(\phi(v_t), t) g(v_{t-1}, v_t, t). \tag{II.2}$$

Let $\mathcal{I} = \{1, 2, ..., I\}$ be a set of resources and $f_i(v, v', t)$ be the amount of resource $i \in \mathcal{I}$ consumed by the searcher at vertex v' during time period t, given search at vertex v during time period $t-1$. Resources may represent physical commodities such as fuel and ordnance that are depleted during the search as well as abstract factors such as notions of risk exposure during the search. In contrast to search by manned aircraft, where significant risks are usually avoided, planners accept higher risks for UAV search missions and would like to balance risk with other factors during the planning process. We discuss the calculation of risk in Section D. The total "consumption" of resource $i \in \mathcal{I}$ along the directed path $\mathcal{P} = \{v_l\}_{l=0}^{k}$ is

$$r_i(\mathcal{P}) = \sum_{t=1}^{k} f_i(v_{t-1}, v_t, t). \tag{II.3}$$

The searcher cannot consume more than $\hat{r}_i$ of resource $i \in \mathcal{I}$ along a path. Hence, the resource-constrained single searcher problem (RSSP) is the problem to find a directed v_0-v_k path $\mathcal{P} = \{v_l\}_{l=0}^{k}$, with $v_k \in \hat{\mathcal{V}}$, $k \in \mathcal{T}$, that maximizes $q(\mathcal{P})$ subject to the constraints

$$r_i(\mathcal{P}) \leq \hat{r}_i, i \in \mathcal{I}. \tag{II.4}$$

We contribute that the single searcher problem (SSP) is generalized to the case with multiple resource constraints. We refer to constraints (II.4) as *side constraints*. In Section D, we examine a case study with two time-invariant resources: risk and fuel. The next section deals with the "unconstrained" problem with no side constraints (referred as SSP), while Sections C and D address the full RSSP.

B. BRANCH-AND-BOUND ALGORITHM FOR SSP

In this section, we consider the single search problem (SSP) that maximizes (II.2) without the side constraints (II.4). Several branch-and-bound algorithms for

the solution of SSP have been developed [46, 50, 18, 34, 13, 52, 32]. These algorithms implicitly enumerate all searcher paths that cannot be proven, by means of a bound, to be nonimproving. The next subsection presents the algorithm in [32], which appears to be the fastest in the literature for SSP under a broad range of conditions; see [32] for an empirical study. The subsequent subsection presents four modifications of that algorithm which generalize and speed up that algorithm.

This section ignores side constraints and, without loss of generality, assumes that an optimal path consists of $T+1$ vertices. To simplify the notation, we also assume in this section that there is no end-point restriction, i.e., $\hat{\mathcal{V}} = \mathcal{V}$. We find identical assumptions in [52, 32]. Initially, we assume that the glimpse detection probability $g(v, v', t)$ is independent of v and write $g(v', t)$, but relax that assumption later in this section.

1. Existing Algorithm

For completeness and ease of reference later, we outline the algorithm in Lau et al. [32]. Given a subpath $\{v_l\}_{l=0}^{t-1}$, $t \in \mathcal{T}$, let $\mathcal{K}(t)$ be the set of triplets of the form $(v_t, t, \bar{q}(v_t, t))$ representing extensions of $\{v_l\}_{l=0}^{t-1}$ yet to be explored. The first element v_t refers to the next vertex to visit, the second element t is the time period[1] to visit the vertex v_t, and the third element $\bar{q}(v_t, t)$ is an upper bound on the probability of detection along any path that starts with the subpath $\{v_l\}_{l=0}^{t}$. The upper bound $\bar{q}(v_t, t)$ consists of three parts. Let $d_t(v_t, t)$ be an upper bound on the probability of detection during time periods $t+1, t+2, ..., T$, given that the searcher starts at v_t at time t, and no detection occurs along the subpath $\{v_l\}_{l=0}^{t}$. The two other parts are the probability of detection on the subpath $\{v_l\}_{l=0}^{t-1}$ and the probability of detection during t. Hence,

$$\bar{q}(v_t, t) = q(\{v_l\}_{l=0}^{t-1}) + p(\phi(v_t), t)g(v_t, t) + d_t(v_t, t). \tag{II.5}$$

[1]This information is currently redundant but the notation is convenient in later generalizations.

We also let $\hat{q}$ denote the largest detection probability found so far among all the examined paths. In this notation, the algorithm in [32] takes the following form.

Algorithm 1 (Solves SSP).

Input: A time-expanded network $(\mathcal{N}, \mathcal{A})$ (defined later as Figure 3) with nodes $n \in \mathcal{N}$ and arcs $(n, n') \in \mathcal{A}$ where $n = \langle v, t-1 \rangle$, $n' = \langle v', t \rangle$. Arc lengths $g(v', t)$ in $(\mathcal{N}, \mathcal{A})$, the initial target distribution $p(\cdot, 1)$, Markov transition matrix Γ, and upper bound $\hat{q}$.

Output: An optimal search path $\mathcal{P}^* = \{v_l\}_{l=0}^{T}$ and its value q^*.

Step 0. Set $t = 0, \mathcal{K}(t) = \{(v_0, 0, \infty)\}$, and $\hat{q} = 0$.

Step 1. If $\mathcal{K}(t)$ is empty, replace t by $t-1$. Else, go to Step 3.

Step 2. If $t = 0$, stop: the last saved path is optimal and $\hat{q}$ is its probability of detection. Else, go to Step 1.

Step 3. Remove from $\mathcal{K}(t)$ the triplet $(v_t, t, \bar{q}(v_t, t))$ with the largest bound $\bar{q}(v_t, t)$.

Step 4. If $\bar{q}(v_t, t) \leq \hat{q}$, go to Step 1. (Current subpath is fathomed.)

Step 5. If $t < T$, then for each vertex $v \in \mathcal{F}(v_t)$, calculate a bound $d_{t+1}(v, t+1)$ as well as $\bar{q}(v, t+1)$ using equation (II.5), and add $(v, t+1, \bar{q}(v, t+1))$ to $\mathcal{K}(t+1)$. Replace t by $t+1$ and go to Step 3. Else, let $\hat{q} = \bar{q}(v_t, t)$ and save the incumbent path $\{v_l\}_{l=0}^{T}$, and go to Step 1.

Clearly, a tight bound $d_t(v_t, t)$ will reduce the number of branching attempts in Step 4 of Algorithm 1. As examined in [51, 52, 32], there is a fundamental trade-off between the effort needed to compute a bound and its tightness. From these studies, it appears that the bounding technique in [32] compares favorably in most situations. We describe that bounding technique in the rest of this subsection.

Consider a subpath $\{v_l\}_{l=0}^{t}$, $t \in \mathcal{T}$, and let $p_g(\cdot, t)$ be the undetected target distribution after search along $\{v_l\}_{l=0}^{t}$, i.e.,

$$p_g(\cdot, t) = \tag{II.6}$$
$$[p(1,t), .., p(\phi(v_t)-1, t), p(\phi(v_t), t)(1 - g(v_t, t)), p(\phi(v_t)+1, t), .., p(C, t)].$$

We use subscript g to indicate that $p_g(\cdot, t)$ is obtained from $p(\cdot, t)$ by applying the glimpse detection probability corresponding to the last vertex in the "current" subpath $\{v_l\}_{l=0}^t$. For any integer $s > t$, $s \in \mathcal{T}$, we also define

$$p_\Gamma(\cdot, s; t) = p_g(\cdot, t)\Gamma^{s-t}. \tag{II.7}$$

As seen, $p_\Gamma(c, s; t)$ is the probability that the target is in cell c at time period $s > t$ and there was no detection during search along the subpath $\{v_l\}_{l=0}^t$. Hence, target distribution $p_\Gamma(\cdot, s; t)$ at time period $s > t$ ignores the effect of search after time period t. If the subpath is $\{v_0\}$, i.e., $t = 0$, we define for notational convenience

$$p_\Gamma(\cdot, s; 0) = p(\cdot, 1)\Gamma^{s-1}, \tag{II.8}$$

for any $s > 0$, $s \in \mathcal{T}$. Moreover, we define $p_\Gamma(c, t; t) = 0$ for all $c \in \mathcal{C}$ and $t = 0, 1, ..., T$.

Now, we construct a time-expanded graph from the network $(\mathcal{V}, \mathcal{E})$ as follows (see Figure 3). Each vertex $v \in \mathcal{V}$ is duplicated T times to define the nodes $\langle v, s\rangle$, $s \in \mathcal{T}$. Let $\mathcal{N}$ be the set of all such nodes as well as the nodes $n_0 = \langle v_0, 0\rangle$ and $\hat{n} = \langle \hat{v}, T+1\rangle$ representing the searcher's prior position and final position, respectively. Here, $\hat{v}$ is an artificial terminal vertex. Two nodes $n = \langle v, s-1\rangle$ and $n' = \langle v', s\rangle$, $v, v' \in \mathcal{V}$ and $s = 2, 3, ..., T$, are connected with an arc (n, n') if and only if $(v, v') \in \mathcal{E}$. Moreover, the node $n_0 = \langle v_0, 0\rangle$ is connected with an arc to a node $n' = \langle v', 1\rangle$, $v' \in \mathcal{V}$, if and only if $(v_0, v') \in \mathcal{E}$; and every node $n = \langle v, T\rangle$, $v \in \mathcal{V}$ is connected with an arc to $\hat{n}$. Let $\mathcal{A}$ be the set of all arcs. For any integer $k \leq T + 1$ and nodes $n_l = \langle v_l, l\rangle \in \mathcal{N}, l = 0, 1, ..., k$, such that $(n_{l-1}, n_l) \in \mathcal{A}$ for all $l = 1, 2, ..., k$, we let the sequence $\{n_l\}_{l=0}^k$ denote a subpath in the time-expanded graph $(\mathcal{N}, \mathcal{A})$.

For some $t \in \{0, 1, ..., T-1\}$, suppose that a subpath $\{v_l\}_{l=0}^t$ in the original graph $(\mathcal{V}, \mathcal{E})$ is given. Then, we endow each arc $(n, n') = (\langle v, s\rangle, \langle v', s+1\rangle) \in \mathcal{A}$, $s = t, t+1, ..., T-1$, in the time-expanded graph $(\mathcal{N}, \mathcal{A})$ with a "reward"

$$c_{n,n'} = [p_\Gamma(\phi(v'), s+1; t) - p_\Gamma(\phi(v), s; t)g(v, s)\Gamma(v, v')]g(v', s+1), \tag{II.9}$$

where $\Gamma(v, v')$ is the $\phi(v)$-$\phi(v')$ element of the Markov transition matrix Γ. We set $c_{n,\hat{n}} = 0$ for all $(n, \hat{n}) \in \mathcal{A}$. We observe that, multiplied out, the first term $p_\Gamma(\phi(v'), s+$

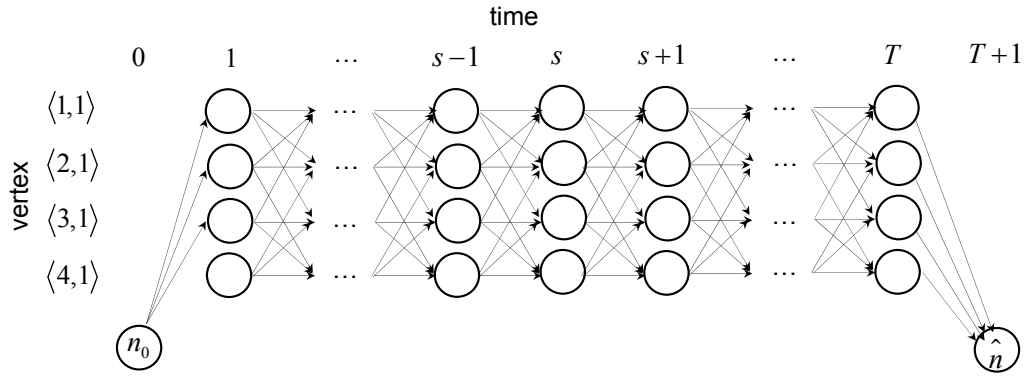

Figure 3. A time-expanded graph from the network in Figure 1 and one altitude. The searcher's initial position is $n_0 = \langle v_0, 0\rangle = \langle\langle 1, 1\rangle, 0\rangle$ and final position is $\hat{n} = \langle \hat{v}, T+1\rangle$.

$1; t)g(v', s+1)$ in (II.9) is effectively the expected number of detections during time period $s+1$, which gives rise to the so-called mean bound [34], and the second term in (II.9) improves the bound by accounting for the effect of search during time period s [32]. We refer to $(\mathcal{N}, \mathcal{A})$ with arc rewards given by (II.9) as the time-expanded network.

In Lau et al. [32], it is shown that given the subpath $\{v_l\}_{l=0}^{t}$, the optimal value of the longest-path problem in the time-expanded network from node $\langle v_t, t\rangle$ to node $\langle \hat{v}, T+1\rangle$, using the rewards in (II.9) as "arc length," provides an upper bound for Algorithm 1. Specifically, this optimal value is an upper bound on the probability of detection during time periods $t+1, t+2, ..., T$, given that the searcher starts at v_t at time t, and no prior detections occurred along the subpath $\{v_l\}_{l=0}^{t}$. We denote this bound by $d_t(v_t, t)$ and refer to it as the dynamic bound, as it needs to be recomputed

every time the *current* subpath is extended.

Since the time-expanded network is acyclic, the longest-path problem can be solved in polynomial calculation time with a standard shortest-path algorithm ([1], pages 77-79). We observe that to compute $d_t(v_t, t)$ given the subpath $\{v_l\}_{l=0}^{t}$, it is only necessary to generate the part of the graph $(\mathcal{N}, \mathcal{A})$, and the corresponding arc rewards, "after" time t and within reach from node $\langle v_t, t\rangle$, since the longest path starts at node $\langle v_t, t\rangle$. Hence, in Step 5 of Algorithm 1, it suffices to generate the time-expanded network "after" time t.

2. Algorithmic Modifications for SSP

Algorithm 1 requires bound calculation at each branching which is a principal bottleneck for that algorithm. Thus we present new bounding techniques to enhance Algorithm 1. This subsection proposes and examines four modifications of Algorithm 1. The first modification extends the bound in [32] to the case in which glimpse detection probability depends on the previous vertex, i.e., $g(v, v', t)$. The following three modifications are aimed to simplify and speed up Algorithm 1. The second modification weakens the bound but greatly simplifies its bound calculation. The third modification improves the weaker bound at little computational expense. The fourth modification takes advantage of a special, but frequently occurring, initial target distribution.

a. Bound for Edge-Dependent Glimpse Detection Probability

Previous studies (see, e.g.,[52, 32]) assume that the glimpse detection probability depends only on the searcher's current waypoint (vertex) and on time. As we argue in Section A, this is somewhat restrictive. Fortunately, we can easily extend the previous studies to the case where the glimpse detection probability also depends on the searcher's previous waypoint. The only modification that is required is to redefine the arc reward $c_{n,n'}$ in (II.9). However, a straightforward replacement of

$g(v,s)$ by $g(v'',v,s)$ in (II.9), where v'' is the vertex prior to v, would ruin the longest-path structure of the bound-calculation problem: $c_{n,n'}$ would no longer depend only on the head and tail of the arc (n,n'). Hence, it is necessary to use the smallest glimpse detection probability $\min_{v''\in\mathcal{R}(v)} g(v'',v,s)$ to eliminate the dependence on the vertex prior to v, where $\mathcal{R}(v)\subset\mathcal{V}$ is the reverse star of v, i.e., $\mathcal{R}(v)=\{v''\in\mathcal{V}\mid(v'',v)\in\mathcal{E}\}$. Consequently, we now endow each arc $(n,n')=(\langle v,s\rangle,\langle v',s+1\rangle)\in\mathcal{A}$ with the reward

$$c_{n,n'}=\left[p_\Gamma(\phi(v'),s+1;t)-p_\Gamma(\phi(v),s;t)\left(\min_{v''\in\mathcal{R}(v)}g(v'',v,s)\right)\Gamma(v,v')\right]g(v,v',s+1). \tag{II.10}$$

With this reward, the bound calculation remains a longest-path problem in an acyclic graph and it can be shown using the same arguments as in [32] which prove that the dynamic bound is valid.

b. Static Bound

Algorithm 1 requires one longest-path calculation in the time-expanded network for each vertex in the forward star of the current vertex to compute the required bounds $d_t(v_t,t)$ (see Step 5). The approach of Algorithm 1 with dynamic bound follows the traditional approach of branch-and-bound algorithms where the bound is reoptimized before each branching. In the present case, the reoptimization corresponds to the longest-path calculation and requires computing the arc rewards $c_{n,n'}$, see (II.10). These calculation can be time consuming, as they typically involve a moderately large Markov transition matrix Γ and associated matrix multiplication. Our numerical example considers that Γ is of size 225 by 225. We propose to use a static bound instead of the dynamic bound proposed in [32] and described in Subsection B.1. As shown below, all the necessary static bounds are computed prior to any branching and are *not* recomputed later.

The dynamic bound $d_t(v_t,t)$ (see Subsection B.1) uses information about search along a current subpath $\{v_l\}_{l=0}^t$. However, the bound remains valid if the effect of search along the current subpath is ignored. This follows from the

same arguments as in the proof of the validity of $d_t(v_t, t)$, see [32]. We denote the new bound $d_0(v_t, t)$, where the subscript 0 indicates that the trivial subpath $\{v_0\}$ is used in (II.10) with $t = 0$ instead of the subpath $\{v_l\}_{l=0}^{t}$, which $d_t(v_t, t)$ utilizes. Hence, the only difference between $d_t(v_t, t)$ and $d_0(v_t, t)$ is that $p_\Gamma(\cdot, \cdot; t)$ is replaced by $p_\Gamma(\cdot, \cdot; 0)$ in (II.10). However, this difference makes the bound $d_0(v_t, t)$ independent of the current subpath used to reach the vertex v_t. Hence, $d_0(v, t)$ can be computed in advance for all nodes $\langle v, t\rangle \in \mathcal{N}$, and dynamical computation of bounds is not required. Consequently, the arc rewards (II.10) and bounds are computed only once. We refer to $d_0(v, t)$ as the static bound. We observe that it is not necessary to carry out a longest-path calculation from each node $\langle v, t\rangle \in \mathcal{N}$ to $\langle \hat{v}, T + 1\rangle$ to obtain $d_0(v, t)$. It is more efficient to carry out the longest-path calculations backward from node $\langle \hat{v}, T + 1\rangle$ to all nodes. This calculation simply amounts to applying a shortest-path algorithm once to the time-expanded network with arc lengths equal to the negative rewards.

In Step 5 of Algorithm 1, we now simply use $d_0(v_t, t)$ instead of $d_t(v_t, t)$. Thus, the modified algorithm does not require any longest-path calculation in Step 5. All bound calculations are done prior to Step 0. Clearly, the modified approach results in a weaker bound and the need for more branching attempts. However, the additional branching attempts may be compensated by shorter per-iteration computing times.

In order to examine the effect of the static bound $d_0(v_t, t)$, we examine the same numerical example as in [32]: An area of interest consists of 11 by 11 cells. The searcher operates only at one altitude and its moves are restricted to vertically and horizontally adjacent cells, excluding diagonal moves. The target remains in the current cell with a probability ρ or moves to one of the vertically or horizontally adjacent cells with probability $1 - \rho$. The different moves are equally likely. The searcher departs cell 1 ($v_0 = 1$), where the cells are numbered from left to right and from top to bottom. Hence, cell 1 is the upper-left-corner cell. The target starts at the center cell, i.e., at cell 61. The time horizon $T = 17$.

We have implemented Algorithm 1 with static bound using Microsoft Visual C++ 6.0 on a desktop computer with a 3.4 GHz Intel Pentium IV processor, 1.0 gigabytes of RAM, and the Microsoft Windows XP operating system. Table 1 shows, for a range of constant glimpse detection probabilities $g(v, v', t)$ and "stay probabilities" ρ, the run times (in seconds) and numbers of bounding attempts of Algorithm 1, as well as those reported in [32]. We especially consider the case of high glimpse detection probability ($g(v, v', t) = 0.99$), in which both static and dynamic bounds tend to be relatively weak. Column 3 of Table 1 presents the resulting run times for Algorithm 1 with static bound. The corresponding run times reported from [32] are found in column 5. (The case $g(v, v', t) = 0.99$ is not considered in [32].) Those reported numbers are achieved on a 2.6 GHz Opteron 152 processor computer using Matlab. Hence, a direct comparison between the run times in columns 3 and 5 is difficult. However, we find reports of run times for other problem instances using a C++ implementation in [32]. A brief comment in [32] based on a single problem instance indicates that the Matlab implementation is 15.6 times slower than the C++ implementation. For the sake of comparison, we scale down the run times in column 5 of Table 1 with 1/15.6 to approximately account for the slower Matlab implementation. We also scale down the run times of column 5 by a factor of 2.6/3.4 to account for the slower computer used in [32]. The resulting scaled down run times are presented in column 6 of Table 1. We observe that the scaled down run times for Algorithm 1 with dynamic bound are somewhat faster than the corresponding run times with static bound. However, the static bound, which is weaker than the dynamic bound, remains fairly competitive, especially for more difficult problem instances.

We compare the strengths of the static and dynamic bounds by counting the number of branching attempts required in Algorithm 1. Columns 4 and 7 of Table 1 give the numbers of branching attempts for the static and dynamic bounds, respectively. The number of branching attempts for the static bound is, on average, 37 times larger than in the case of the dynamic bound. We observe that the greater

		Static Bound		Dynamic Bound [32]			Dynamic Bound	
		Time	Branching	Time	Scaled	Branching	Time	Branching
$g(v, v', t)$	ρ	(sec.)	attempts	(sec.)	(sec.)	attempts	(sec.)	attempts
	0.3	3.59	2,301,182	14.56	0.71	58,314	2.09	58,314
0.3	0.6	2.58	1,635,517	14.53	0.71	52,369	2.11	52,369
	0.9	11.47	7,338,492	157.30	7.71	380,889	20.75	380,889
	0.3	5.38	3,424,282	19.07	0.94	49,774	2.59	49,774
0.6	0.6	2.58	1,620,402	23.76	1.16	47,454	3.03	47,454
	0.9	59.26	38,186,809	730.96	35.84	2,185,066	103.08	2,185,066
	0.3	5.78	3,675,197	21.55	1.06	45,019	2.78	45,019
0.9	0.6	2.89	1,831,875	30.58	1.50	59,527	3.91	59,527
	0.9	176.26	113,646,357	2902.27	142.30	11,299,259	431.38	11,299,259
	0.3	6.09	3,865,002	N/A	N/A	N/A	2.91	46,339
0.99	0.6	3.00	1,896,960	N/A	N/A	N/A	3.91	58,752
	0.9	192.06	123,822,672	N/A	N/A	N/A	558.63	16,685,969

Table 1. Run times and number of branching attempts (counted in Step 4) for Algorithm 1 with static and dynamic bounds on 11 by 11 cell search problem with time horizon $T = 17$. Columns labeled "Dynamic Bound [32]" correspond to original and speed-adjusted results from [32].

number of bounding attempts is partially compensated for by avoiding dynamical reoptimization of the bound.

For a direct comparison between the static and dynamic bounds, we implement Algorithm 1 with dynamic bound in Microsoft Visual C++ 6.0. We made a significant effort to ensure that the implementation is efficient, including efficient handling of sparse matrices. Column 8 and 9 of Table 1 report the run times and the number of branching attempts for our implementation of Algorithm 1 with dynamic bound, respectively. We observe that our implementation results in identical numbers of branching attempts compared to the implementation in [32]. When comparing columns 6 and 8, we see that our implementation of the dynamic bound results in somewhat longer run times than the scaled times from [32]. However, the longer times in column 8 compared to column 6 can partially be due to an excessively aggressive scaling of run times going from column 5 to column 6.

While implementing the dynamic bound, we noted a significant challenge associated with efficient matrix multiplication and data handling. Since the

dynamic bound $d_t(v,t)$ is reoptimized a large number of times, it is paramount to carry out the related calculations efficiently. In comparison, it is rather trivial to implement a static bound. It is not critical to carry out the longest-path calculations highly efficiently in the static case as they are only done once.

We also observe that both static and dynamic bounds tend to be weaker when the target is nearly stationary (e.g., $\rho = 0.9$) and glimpse detection probability is large (e.g., $g(v,v',t) = 0.9$ or 0.99). For these problem instances, the implementation with dynamic bound is more sensitive to the change in data. In fact, comparing the instance ($g(v,v',t) = 0.9$ and $\rho = 0.9$) to the instance ($g(v,v',t) = 0.99$ and $\rho = 0.9$), we see that the run time and the number of branching attempts in the case of the dynamic bound become 1.29 and 1.48 times larger, respectively. On the other hand, the static bound is less sensitive, and its numbers become only 1.09 times larger. These numbers indicate that it becomes even less worthwhile to invest time in dynamic bound calculations when the bounds are relatively weak anyway.

c. Directional Static Bound

As seen from Table 1, the static bound is substantially weaker than the dynamic bound. We derive a stronger static bound motivated by the classical approach to handling turn-radius constraints in vehicle-routing problems [8].

In the longest-path calculations for the static bound, the reward of arc $(\langle v,s\rangle, \langle v',s+1\rangle)$ is, effectively, the probability of detection at vertex v' during time period $s+1$ and no detection at vertex v during time period s. Of course, this overestimates the probability of detection at vertex v' during time period $s+1$ and no prior detections, as detection could occur prior to time period s. We strengthen the static bound if we redefine the arc reward to be the probability of detection at vertex v' during time period $s+1$ and no detection at vertex v during time period s *and* no detection at the vertex visited during time period $s-1$. However, redefining the arc reward to depend not only on the arc's head and tail nodes, but also on a previous node, ruins the longest-path structure of the bound-calculation problem.

A similar situation arises in vehicle routing problems for vehicles with turn-radius constraints or penalties. The classical approach to handle that situation is to duplicate each node a number of times equal to the number of nodes in the node's reverse star. An arc in the resulting "node-expanded" network then carries information about three nodes, not only two, and a desirable network structure of the problem can be maintained. Fortunately, it is practical to carry out such a node-expansion approach in the problems of interest in this paper because the number of nodes in the reverse star is typically quite moderate. Hence, we proceed along the stated lines and develop a node-and-time expanded network, in which the improved static bound can be calculated by solving a longest-path problem. We refer to this improved bound as the directional static bound.

For any $n' \in \mathcal{N}$, let $\mathcal{R}(n') \subset \mathcal{N}$ be the reverse star of n', i.e., $\mathcal{R}(n') = \{n \in \mathcal{N} | (n, n') \in \mathcal{A}\}$. Then, for any $n, n' \in \mathcal{N} \backslash \{\hat{n}\}$ such that $(n, n') \in \mathcal{A}$, we define an expanded node $\xi = \langle n, n' \rangle$. We do not expand the end node, so we set $\hat{\xi} = \hat{n}$. Let Ξ be the set of all expanded nodes. Two expanded nodes $\xi, \xi' \in \Xi$ are connected by an expanded arc (ξ, ξ') if $\xi = \langle n, n' \rangle$ and $\xi' = \langle n', n'' \rangle$. Let the set of all expanded arcs be Ω. The node-and-time expanded graph is illustrated in Figure 4.

We endow each expanded arc in the node-and-time expanded graph (Ξ, Ω) with a reward similar to (II.10). To derive the exact form of this reward, we need the following building blocks. For any $v, v' \in \mathcal{V}$ and $t \in \mathcal{T}$, let $M_t(v, v')$ be a C by C identity matrix with the $\phi(v')$-th diagonal element set equal to $1 - g(v, v', t)$. We also let $\Gamma(v')$ be the $\phi(v')$-th column of the Markov transition matrix Γ.

From (II.2) and the recursive application of (II.1), we see that the probability of detection along a path $\{v_l\}_{l=0}^T$ is given by

$$\begin{aligned} q(\{v_l\}_{l=0}^T) = p(\phi(v_1), 1) g(v_0, v_1, 1) + \\ p(\cdot, 1) M_1(v_0, v_1) \Gamma(v_2) g(v_1, v_2, 2) + \\ p(\cdot, 1) M_1(v_0, v_1) \Gamma M_2(v_1, v_2) \Gamma(v_3) g(v_2, v_3, 3) + \\ p(\cdot, 1) M_1(v_0, v_1) \Gamma M_2(v_1, v_2) \Gamma M_3(v_2, v_3) \Gamma(v_4) g(v_3, v_4, 4) + \end{aligned} \quad \text{(II.11)}$$

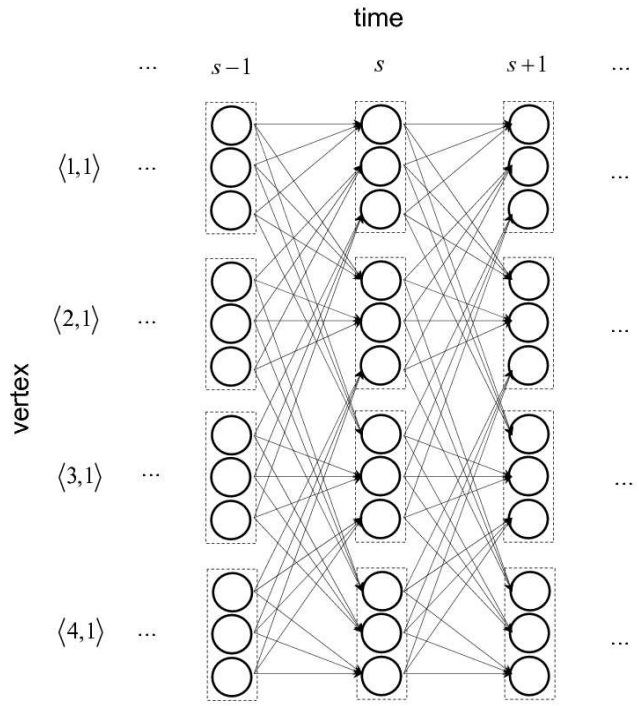

Figure 4. A node-and-time expanded network from the time-expanded graph (Figure 3).

$$\vdots$$

$$p(\cdot,1)M_1(v_0,v_1)\Gamma M_2(v_1,v_2)\cdot\ldots\cdot\Gamma M_{T-1}(v_{T-2},v_{T-1})\Gamma(v_T)g(v_{T-1},v_T,T).$$

The expression (II.11) gives insight into a class of bounds on the probability of detection, including the static bound $d_0(v_t,t)$. If we replace $M_t(\cdot,\cdot)$ by the identity matrix in (II.11), we find that

$$\begin{aligned} q(\{v_l\}_{l=0}^T) \le\ & p(\phi(v_1),1)g(v_0,v_1,1) + \\ & p(\cdot,1)\Gamma(v_2)g(v_1,v_2,2) + \\ & p(\cdot,1)\Gamma\Gamma(v_3)g(v_2,v_3,3) + \\ & p(\cdot,1)\Gamma\Gamma\Gamma(v_4)g(v_3,v_4,4) + \\ & \vdots \\ & p(\cdot,1)\Gamma^{T-2}\Gamma(v_T)g(v_{T-1},v_T,T). \end{aligned} \tag{II.12}$$

In (II.12), the "reward" received during a time period is simply the expected number of detection during that time period and depends only on the current and previous vertices. Hence, it is possible to compute an upper bound on the optimal probability of detection by finding a path $\{v_l\}_{l=0}^T$ that maximizes the right-hand side in (II.12). This calculation amounts to a longest-path problem and is, in fact, the approach in [34]. (Note, however, that [34] assumes that the glimpse detection probability is independent of the previous vertex.)

If we replace each $M_t(\cdot,\cdot)$ by the identity matrix everywhere except the last matrix of each line in (II.11), we obtain

$$\begin{aligned} q(\{v_l\}_{l=0}^T) \leq\ & p(\phi(v_1),1)g(v_0,v_1,1) + \\ & p(\cdot,1)M_1(v_0,v_1)\Gamma(v_2)g(v_1,v_2,2) + \\ & p(\cdot,1)\Gamma M_2(v_1,v_2)\Gamma(v_3)g(v_2,v_3,3) + \\ & p(\cdot,1)\Gamma\Gamma M_3(v_2,v_3)\Gamma(v_4)g(v_3,v_4,4) + \\ & \vdots \\ & p(\cdot,1)\Gamma^{T-2}M_{T-1}(v_{T-2},v_{T-1})\Gamma(v_T)g(v_{T-1},v_T,T). \end{aligned} \tag{II.13}$$

Now, the reward received during each time period also depends on the searcher's position two time periods ago, and the problem of finding a path that maximizes the right-hand side is no longer a longest-path problem. However, the bound remains valid with the following minor modification, where the maximization of a matrix with a single element different from zero or one is simply the maximization of that element:

$$\begin{aligned} q(\{v_l\}_{l=0}^T) \leq\ & p(\phi(v_1),1)g(v_0,v_1,1) + \\ & p(\cdot,1)\left(\max_{v\in\mathcal{R}(v_1)} M_1(v,v_1)\right)\Gamma(v_2)g(v_1,v_2,2) + \\ & p(\cdot,1)\Gamma\left(\max_{v\in\mathcal{R}(v_2)} M_2(v,v_2)\right)\Gamma(v_3)g(v_2,v_3,3) + \\ & p(\cdot,1)\Gamma\Gamma\left(\max_{v\in\mathcal{R}(v_3)} M_3(v,v_3)\right)\Gamma(v_4)g(v_3,v_4,4) + \\ & \vdots \end{aligned} \tag{II.14}$$

$$p(\cdot,1)\Gamma^{T-2}\left(\max_{v\in\mathcal{R}(v_{T-1})} M_{T-1}(v,v_{T-1})\right)\Gamma(v_T)g(v_{T-1},v_T,T).$$

After this modification, we see that the reward during each time period only depends on the current and previous vertices. Hence, again, it is possible to compute an upper bound on the optimal probability of detection by solving a longest-path problem. In fact, this is exactly the approach we described in Subsection B.2a and it can be shown that the reward in the longest-path problem $c_{n,n'}$, see (II.10), can be deduced from (II.14). Specifically, when the current subpath in (II.10) is $\{v_0\}$, we have for arc $(n,n') = (\langle v,s\rangle, \langle v', s+1\rangle) \in \mathcal{A}$ that

$$c_{n,n'} = p(\cdot,1)\Gamma^{s-1}\left(\max_{v''\in\mathcal{R}(v)} M_s(v'',v)\right)\Gamma(v')g(v,v',s+1). \tag{II.15}$$

Using similar arguments, we define the directional static bound as follows. Clearly,

$$\begin{aligned} q(\{v_l\}_{l=0}^T) \le\ & p(\phi(v_1),1)g(v_0,v_1,1) + \\ & p(\cdot,1)M_1(v_0,v_1)\Gamma(v_2)g(v_1,v_2,2) + \\ & p(\cdot,1)\left(\max_{v\in\mathcal{R}(v_1)} M_1(v,v_1)\right)\Gamma M_2(v_1,v_2)\Gamma(v_3)g(v_2,v_3,3) + \\ & p(\cdot,1)\Gamma\left(\max_{v\in\mathcal{R}(v_2)} M_2(v,v_2)\right)\Gamma M_3(v_2,v_3)\Gamma(v_4)g(v_3,v_4,4) + \\ & \vdots \\ & p(\cdot,1)\Gamma^{T-3}\left(\max_{v\in\mathcal{R}(v_{T-2})} M_{T-2}(v,v_{T-2})\right)\Gamma M_{T-1}(v_{T-2},v_{T-1})\Gamma(v_T)g(v_{T-1},v_T,T). \end{aligned} \tag{II.16}$$

Hence, we can compute an upper bound on the optimal probability of detection by finding a path $\{v_l\}_{l=0}^T$ that maximizes the right-hand side of (II.16). This calculation amounts to a longest-path problem in the node-and-time expanded graph (Ξ,Ω). The arc reward in this longest-path problem is deduced from (II.16). Specifically, an expanded arc $(\xi,\xi') = (\langle n'',n\rangle, \langle n,n'\rangle) \in \Omega$, with $n'' = \langle v'', s-1\rangle$, $n = \langle v,s\rangle$, and $n' = \langle v', s+1\rangle$, is endowed with the reward

$$c_{\xi,\xi'} = p(\cdot,1)\Gamma^{s-2}\left(\max_{v'''\in\mathcal{R}(v'')} M_{s-1}(v''',v'')\right)\Gamma M_s(v'',v)\Gamma(v')g(v,v',s+1). \tag{II.17}$$

We refer to the node-and-time expanded graph (Ξ, Ω) with the arc rewards $c_{\xi,\xi'}$ from (II.17) as the node-and-time expanded network. Since the node-and-time expanded graph is acyclic, longest-path problems are solvable by standard shortest-path algorithms.

In view of the above discussion, we obtain the following result.

Proposition 1 *For any $v' \in \mathcal{V}$ and $t \in \mathcal{T}$, let*

(i) $d_0(v', t)$ be the value of the longest-path from node $\langle v', t\rangle$ to node $\hat{n}$ in the time-expanded graph $(\mathcal{N}, \mathcal{A})$ with arc rewards given by (II.15), and

(ii) $\delta_0(v, v', t)$ be the value of the longest-path from expanded node $\langle\langle v, t-1\rangle, \langle v'\ t\rangle\rangle$ to expanded node $\hat{\xi}$ in the node-and-time expanded graph (Ξ, Ω) with arc rewards given by (II.17).

Then, both $d_0(v', t)$ and $\delta_0(v, v', t)$ are upper bounds on the probability of detection during time periods $t+1, t+2, ..., T$ for any path $\{v_l\}_{l=0}^{T}$ with $v_{t-1} = v$ and $v_t = v'$. Moreover, $\delta_0(v, v', t) \leq d_0(v', t)$.

We refer to $\delta_0(v, v', t)$ as the directional static bound and see from Proposition 1 that it is at least as strong as the static bound. We demonstrate in an empirical study below that it may be substantially stronger.

Clearly, building the node-and-time expanded graph (Ξ, Ω), computing the associated rewards, and calculating the longest-paths take some computing time. However, the process is only carried out once before the start of Algorithm 1 and the computed bounds are stored for later use. Hence, the time for computing the directional static bounds remains small compared to the overall run time of Algorithm 1. We conjecture that the use of the node-and-time expanded graph (Ξ, Ω) with dynamic reoptimization of bounds will not be efficient due to the small but significant effort required to build the node-and-time expanded graph, compute associated arc rewards, and carry out the longest-path calculations. We observe that this conjecture appears to be aligned with [32], which alludes to the inefficiency of a dynamic bound based on more than one-time-step look-behind.

In Table 2, we report computational results for Algorithm 1 with the directional static bound applied to the same problem instances as in Table 1. Columns 3 and 4 give run times and number of branching attempts, respectively, for Algorithm 1 using the directional static bound. We observe that, on average, the number of branching attempts has been reduced to 58.1% by using the directional static bound compared with the static bound. (Compare column 4 in Table 1 with column 4 in Table 2.) Similarly, the run times have been reduced to 58.2% by using the directional static bound. Since the reduction in branching attempts and run time are essentially identical, we conclude that the time for computing the directional static bound is small compared to the overall run time.

		Algo. 1: D-Static		Algo. 2: Static & Red.		Algo. 2: D-Static & Red.	
		Time	Branching	Time	Branching	Time	Branching
$g(v,v',t)$	ρ	(sec.)	attempts	(sec.)	attempts	(sec.)	attempts
	0.3	2.59	1,642,619	0.22	129,990	0.19	91,198
0.3	0.6	1.80	1,125,929	0.17	94,307	0.16	65,485
	0.9	6.30	4,032,951	0.58	354,677	0.41	223,435
	0.3	3.50	2,245,784	0.31	194,425	0.27	128,526
0.6	0.6	1.45	902,409	0.17	92,997	0.14	57,015
	0.9	29.67	19,321,387	2.17	1,442,612	1.37	844,747
	0.3	3.59	2,282,989	0.34	209,160	0.28	138,598
0.9	0.6	1.66	1,037,829	0.17	101,216	0.17	61,005
	0.9	80.50	52,527,302	4.95	3,323,101	2.89	1,837,647
	0.3	3.77	2,396,764	0.36	219,217	0.28	137,063
0.99	0.6	1.72	1,080,459	0.19	102,763	0.17	62,890
	0.9	87.99	57,427,410	5.39	3,592,696	3.09	1,974,871

Table 2. Run time and number of branching attempts (counted in Step 4) for Algorithm 1 on problem instances of Table 1 using directional static bound (D-Static) and for Algorithm 2 using static bound and network reduction (Static & Red.) and directional static bound and network reduction (D-Static & Red.).

d. Network Reduction

In some practical situations, the searcher's and the target's initial positions are relatively far from each other. Hence, for a number of time periods the searcher will only examine cells where the target is guaranteed not to be located. In these initial moves, the searcher is simply positioning itself for the later search. This

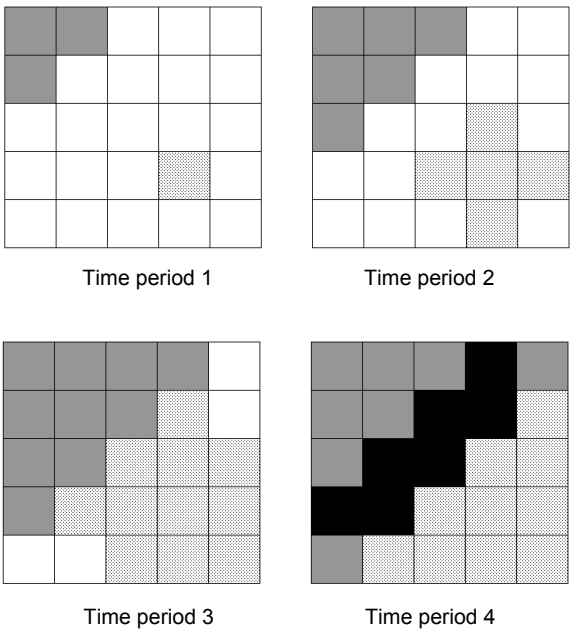

Figure 5. An area of interest composed of 5 by 5 cells. For each time period, the unshaded, lightly-shaded, heavily-shaded, and completely-shaded cells describe the regions where neither searcher nor target stay, only target possibly stays, only searcher possibly stays, and target and searcher possibly stay, respectively.

is the situation in the numerical examples of [32]. In this subsection, we derive a network reduction technique that utilizes this situation.

Consider the time-expanded graph $(\mathcal{N}, \mathcal{A})$. Suppose that the searcher flies, for the first s time periods, along a subpath $\{n_t\}_{t=0}^{s}$, with $n_t = \langle v_t, t\rangle$, such that $p(\phi(v_t), t) = 0$ for all $t < s$, and $p(\phi(v_s), s) > 0$. Then, the searcher cannot detect the target prior to time period s. We refer to the last node n_s of the subpath $\{n_t\}_{t=0}^{s}$ as a node-of-first-contact. It is typically straightforward to determine a set of nodes-of-first-contact by applying a standard search algorithm ([1], pages 73-77) on the time-expanded network. In Figure 5, we illustrate a situation where the searcher can at the earliest detect the target during time period 4, and also note that the time period of all nodes-of-first-contact $\langle v_s, s\rangle \in \mathcal{N}$ is $s \geq 4$.

We observe that there might be several different subpaths $\{n_t\}_{t=0}^{s}$, $n_t = \langle v_t, t\rangle$ to reach a node-of-first-contact $\langle v_s, s\rangle \in \mathcal{N}$. However, the subpath used to reach $\langle v_s, s\rangle$ is of no or little importance. In fact, if $g(v_{s-1}, v_s, s)$ does not vary with the choice of v_{s-1}, all subpaths to $\langle v_s, s\rangle$ have probability of detection equal to $p(\phi(v_s), s)g(v_{s-1}, v_s, s)$; with $p(\phi(v_s), s) = p(\cdot, 1)\Gamma^{s-2}\Gamma(v_s)$. Thus, it is enough to find a subpath to each node-of-first-contact $\langle v_s, s\rangle$ using a standard search algorithm, connect initial node $\langle v_0, 0\rangle$ to $\langle v_s, s\rangle$ directly with a "jump" arc representing the move to $\langle v_s, s\rangle$, and ignore time periods $1, 2, ..., s-1$ during the branch-and-bound algorithm. Clearly, this procedure may reduce the amount of branching attempts significantly. We can generalize this to the case with $g(v_{s-1}, v_s, s)$ varying with respect to v_{s-1}, as discussed in Subsection C.2c.

If a lower bound $\hat{q}$ on the optimal probability of detection is available in advance of the network reduction procedure described above, it may not be necessary to consider all nodes-of-first-contact. Fortunately in the single searcher problems (SSP), a lower bound is trivially obtained by computing the probability of detection along the path corresponding to the static bound $d_0(v_0, 0)$. Furthermore, for each node-of-first-contact $\langle v_s, s\rangle$, an upper bound on the probability of detection after time period s with no prior detections (e.g., the static bound $d_0(v_s, s)$) is available in advance of both branch-and-bound and network-reduction procedures. The upper and lower bounds can be used to eliminate some nodes-of-first-contact that cannot lie on an optimal path. Specifically, if $p(\phi(v_s), s)g(v_{s-1}, v_s, s) + d_0(v_s, s) \leq \hat{q}$, we can eliminate $\langle v_s, s\rangle$ and all incoming and outgoing arcs. Thus, the amount of branching attempts may be reduced further.

In order to take advantage of this reduced network in the branch-and-bound algorithm, we construct a second algorithm (Algorithm 2) that generalizes the branch-and-bound mechanism of Algorithm 1. Before we describe the algorithm, we need to clarify some notation. After the algorithm is presented, we describe the network reduction procedure in detail.

Given a subpath $\{n_l\}_{l=0}^{k}$, $n_l = \langle v_l, l\rangle$, $k \in \mathcal{T}$, let $\mathcal{K}(i)$, as before, be the set of triplets $(v_t, t, \bar{q}(v_t, t))$ representing extensions of $\{n_l\}_{l=0}^{k}$ yet to be explored. Now, the index i of $\mathcal{K}(i)$ refers to the depth from node $\langle v_0, 0\rangle$ to the next node $\langle v_t, t\rangle$ in the branch-and-bound tree. Since $\langle v_0, 0\rangle$ is directly connected to each node-of-first-contact $\langle v_s, s\rangle$, the corresponding depth in the branch-and-bound tree is 1. For reporting purposes, a subpath $\{n_l\}_{l=0}^{s}$ is stored for each node-of-first-contact $\langle v_s, s\rangle$.

Algorithm 2 (Solves SSP).

Input: A time-expanded network $(\mathcal{N}, \mathcal{A})$ with arc length $c_{n,n'}$ or a node-and-time expanded network (Ξ, Ω) with arc length $c_{\xi,\xi'}$. The initial target distribution $p(\cdot, 1)$, Markov transition matrix Γ, and upper bound $\hat{q}$.

Output: An optimal search path $\mathcal{P}^* = \{v_l\}_{l=0}^{T}$ and its value q^*.

Step 0. Calculate $\delta_0(v, v', t)$ for all $t \in \mathcal{T}$ and $v, v' \in \mathcal{V}$ such that $(v, v') \in \mathcal{E}$ or $d_0(v', t)$ for all $t \in \mathcal{T}$ and $v' \in \mathcal{V}$, and calculate a lower bound $\hat{q}$. Set $i = 0, \mathcal{K}(i) = \{(v_0, 0, \infty)\}$.

Step 1. If $\mathcal{K}(i)$ is empty, replace i by $i - 1$. Else, go to Step 3.

Step 2. If $i = 0$, stop: the last saved path is optimal and $\hat{q}$ is its probability of detection. Else, go to Step 1.

Step 3. Remove from $\mathcal{K}(i)$ the triplet $(v_t, t, \bar{q}(v_t, t))$ with the largest bound $\bar{q}(v_t, t)$.

Step 4. If $\bar{q}(v_t, t) \leq \hat{q}$, go to Step 1. (Current subpath is fathomed.)

Step 5. If $i = 0$, replace i by $i + 1$, and go to Step 3. ($\mathcal{K}(1)$ is populated in the network reduction procedure.)

Step 6. If $t < T$, then for each vertex $v \in \mathcal{F}(v_t)$, calculate $\bar{q}(v, t+1)$ from (II.5) using bounds $d_0(v, t+1)$ or $\delta_0(v_t, v, t+1)$, and add $(v, t+1, \bar{q}(v, t+1))$ to $\mathcal{K}(i+1)$. Replace i by $i + 1$, and go to Step 3. Else, let $\hat{q} = \bar{q}(v_t, t)$ and save the incumbent path $\{v_l\}_{l=0}^{T}$, and go to Step 1.

We now present the network reduction procedure that can be implemented as part of Step 0 of Algorithm 2. The procedure assumes that a static bound $d_0(v', t)$ is available as well as a lower bound on the optimal probability of detection $\hat{q}$. If the directional static bound is available instead of the static bound, replace $d_0(v', t)$

by $\delta_0(v, v', t)$ in the procedure below. We note again that the network reduction procedure is valid under the assumption that $g(v_{s-1}, v_s, s)$ does not vary with the choice of $v_{s-1} \in \mathcal{R}(v_s)$ among all nodes-of-first-contact $\langle v_s, s\rangle$. We generalize this in Section C.

Network Reduction Procedure R1.

Input: A time-expanded network $(\mathcal{N}, \mathcal{A})$ with arc length $c_{n,n'}$, the initial target distribution $p(\cdot, 1)$, Markov transition matrix Γ, and upper bound $\hat{q}$.

Output: Nodes-of-first-contacts $\langle v_s, s\rangle$ which are not dominated by others.

Step 1. Find all nodes-of-first-contact $\langle v_s, s\rangle \in \mathcal{N}$. If none exist with $s > 1$, then stop.

Step 2. For each $\langle v_s, s\rangle$, calculate $\bar{q}(v_s, s) = p(\phi(v_s), s)g(v_{s-1}, v_s, s)+d_0(v_s, s)$.

Step 3. Eliminate all nodes-of-first-contact $\langle v_s, s\rangle$ with $\bar{q}(v_s, s) \leq \hat{q}$.

Step 4. For each nodes-of-first-contact $\langle v_s, s\rangle$ not eliminated, store the triplet $(v_s, s, \bar{q}(v_s, s))$ in $\mathcal{K}(1)$.

Table 2 illustrates the effect of the network-reduction technique as applied to the same problem instances as in Table 1. Columns 5 and 6 of Table 2 present the run time and number of branching attempts, respectively, for Algorithm 2 with static bound and network reduction. On average, the run times and and the branching attempts are reduced to 5.3% and 5.0% of the corresponding numbers obtained without the network reduction technique (see columns 3 and 4 in Table 1), respectively. When applying both network reduction and directional static bound, we obtain the run times and numbers of branching attempts reported in columns 7 and 8 of Table 2. It is clear that network reduction and directional static bound have complementary positive effect and the run times and numbers of branching attempts are further reduced.

Table 3 presents computational results for a larger problem instance with 15 by 15 cells and a time horizon $T = 20$. Again, the searcher starts in the upper-left cell and the targets starts in the center cell. As seen from Table 3, the run

times remain rather short for Algorithm 2 with directional static bound and network reduction (columns 3 and 4) while the times increases dramatically for Algorithm 1 with dynamic bound (columns 5 and 6). Furthermore, Algorithm 2 with directional static bound and network reduction is less sensitive to the detrimental effect of a near stationary target (e.g., $\rho = 0.9$) and high glimpse detection probability (e.g., $g(v, v', t) = 0.9$ or 0.99), a case where the bounds tend to be weak.

		Algo. 2: D-Static & Red.		Algo. 1: Dynamic Bound	
		Time	Branching	Time	Branching
$g(v, v', t)$	ρ	(sec.)	attempts	(sec.)	attempts
	0.3	3.08	103,811	20.24	328,672
0.3	0.6	2.94	66,185	21.22	311,645
	0.9	3.58	238,089	107.11	1,352,503
	0.3	3.14	122,941	20.28	248,727
0.6	0.6	2.92	51,977	30.47	288,738
	0.9	4.41	571,649	701.17	8,668,034
	0.3	3.17	129,276	29.61	292,818
0.9	0.6	2.95	57,353	45.05	404,299
	0.9	5.94	1,076,948	2323.48	30,173,994
	0.3	3.13	128,776	31.97	301,498
0.99	0.6	2.92	60,449	48.42	441,664
	0.9	5.77	1,016,918	3092.63	45,329,829

Table 3. Run times and number of branching attempts for Algorithm 2, with directional static bound and network reduction, compared with Algorithm 1 with dynamic bound on 15 by 15 cell search problem with time horizon $T = 20$.

The same problem instances were also examined in [32], which reports a run time of 10.9 seconds for the case with $g(v, v', t) = 0.6$ and $\rho = 0.6$ using a C++ implementation of Algorithm 1 with a dynamic bound running on a 2.6 GHz computer. We observe that our implementation appears to be somewhat slower than the one achieved in [32] with 30.47 seconds compared to 10.9 seconds (on a presumedly slightly slower computer). However, Algorithm 2 with directional static bound and network reduction appears to offer a noticeable advantage over Algorithm 1 with dynamic bound as derived in [32]. In principle, Algorithm 1 with dynamic bound can also be speeded up by using the proposed network reduction technique. However, we

have not examined that possibility. We adopt a directional static bound with network reduction as the basis for extension to the case with side constraints discussed in the next section.

C. BRANCH-AND-BOUND ALGORITHM FOR RSSP

We now turn the attention to the full problem with side constraints, i.e., the resource-constrained single searcher problem (RSSP) formulated in Section A. We first develop a static bound based on Lagrangian relaxation that can be used within a branch-and-bound algorithm in the form of Algorithms 1 and 2. Second, we briefly discuss the development of a Lagrangian directional static bound. Third, we develop a series of network reduction techniques. Fourth, we combine the resulting procedures and present the complete algorithm.

1. Lagrangian Static Bound

Consider the time-expanded network $(\mathcal{N}, \mathcal{A})$, see Subsection B.1, with the arc rewards $c_{n,n'}$ given in (II.15). Now, we also endow each arc $(n, n') \in \mathcal{A}$, $n = \langle v, t-1 \rangle$ and $n' = \langle v', t \rangle$, with weights $r_{i,n,n'} = f_i(v, v', t), i \in \mathcal{I}$. While computing the static bound in the case of no side constraints amounts to solving a longest-path problem on the time-expanded graph, a similar bound in the case with side constraints will need to account for those constraints. Specifically, a static bound can be obtained by solving a constrained longest-path problem. We formulate this problem as an integer program on a slightly modified time-expanded graph.

We use the same time-expanded graph as in Subsection B.1, with the following modifications. We recall that $\hat{\mathcal{V}}$ is the set of vertices where the search can end. Now, every node $n = \langle v, t \rangle$, $v \in \hat{\mathcal{V}}$, $t \in \mathcal{T}$ is connected to the artificial destination node $\hat{n}$ with an arc. This modification allows the searcher to terminate the search prior to time period T to avoid violating the side constraints. Furthermore, all arcs $(n, \hat{n})$ with $n = \langle v, T \rangle$, $v \notin \hat{\mathcal{V}}$, are removed from the time-expanded network. This modification makes the searcher end at $v \in \hat{\mathcal{V}}$. We still let $\mathcal{A}$ denote the set of all arcs.

We formulate the constrained longest-path problem on the time-expanded graph $(\mathcal{N}, \mathcal{A})$ as an integer program. We consider an ordering of $\mathcal{A}$ and let $\mathbf{A}$ denote the $|\mathcal{N}|$ by $|\mathcal{A}|$ node-arc incidence matrix for the time-expanded graph. For each arc $a = (n, n') \in \mathcal{A}$, let $A_{n,a} = 1$, $A_{n',a} = -1$, and $A_{n'',a} = 0$ for any $n'' \in \mathcal{N}, n'' \neq n, n'$ be the elements of $\mathbf{A}$. Let $\mathbf{b}$ denote the $|\mathcal{N}|$-vector with $b_{n_0} = 1$, $b_{\hat{n}} = -1$ and $b_n = 0$ for all $n \in \mathcal{N} \backslash \{n_0, \hat{n}\}$. We also define the additional notation: $\hat{\mathbf{r}} = (\hat{r}_1, \hat{r}_2, \ldots, \hat{r}_I)^T$, where T denotes the transpose. We collect the rewards $c_{n,n'}$ in the $|\mathcal{A}|$-dimensional row vector $\mathbf{c}$. Also, for each $i \in \mathcal{I}$, we define the $|\mathcal{A}|$-dimensional row vector $\mathbf{r}_i$ to contain the weights $r_{i,n,n'}$ and we let $\mathbf{R}$ be the $|\mathcal{I}|$ by $|\mathcal{A}|$ matrix with $\mathbf{r}_i$ as its rows. Finally, we let $\mathbf{x}$ be a $|\mathcal{A}|$-dimensional column vector, where $x_{n,n'} = 1$ if arc (n, n') is used by a path, and zero otherwise. Then, the constrained longest-path problem, see [1], can be written as:

$$z^* \equiv \max_{\mathbf{x} \in \{0,1\}^{|\mathcal{A}|}} \mathbf{cx} \tag{II.18}$$

$$\text{s.t. } \mathbf{Ax} = \mathbf{b}$$

$$\mathbf{Rx} \leq \hat{\mathbf{r}}. \tag{II.19}$$

In principle, the solution of the constrained longest-path problem provides a static bound. However, the constrained longest-path problem is NP-complete even for the case with an acyclic graph ([22], pages 213-214). Hence, we prefer to avoid solving such problems within an algorithm. We proceed by introducing an additional relaxation that is motivated by the solution approach for the constrained shortest-path problem in [24, 9, 10].

Using the standard theory of Lagrangian relaxation (see, e.g., [1], Chapter 16) and an $|\mathcal{I}|$-dimensional row vector $\boldsymbol{\lambda} \geq \mathbf{0}$, we find that

$$z^* \leq z(\boldsymbol{\lambda}) \equiv \max_{\mathbf{x} \in \{0,1\}^{|\mathcal{A}|}} \mathbf{cx} - \boldsymbol{\lambda}(\mathbf{Rx} - \hat{\mathbf{r}}) \tag{II.20}$$

$$\text{s.t. } \mathbf{Ax} = \mathbf{b}.$$

Rewriting the objective function, we can optimize the Lagrangian upper bound $z(\boldsymbol{\lambda})$ through

$$\begin{aligned} \bar{z} &\equiv \min_{\boldsymbol{\lambda} \geq \mathbf{0}} z(\boldsymbol{\lambda}) && \text{(II.21)} \\ &= \min_{\boldsymbol{\lambda} \geq \mathbf{0}} \max_{\mathbf{x} \in \{0,1\}^{|\mathcal{A}|}} (\mathbf{c} - \boldsymbol{\lambda}\mathbf{R})\mathbf{x} + \boldsymbol{\lambda}\hat{\mathbf{r}} && \text{(II.22)} \\ &\qquad \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned}$$

For any fixed $\boldsymbol{\lambda} \geq 0$, computing the upper bound $z(\boldsymbol{\lambda})$ simply requires the solution of a longest-path problem with Lagrangian-modified arc lengths in an acyclic graph. The outer minimization over $\boldsymbol{\lambda}$ can be solved by several methods [19, 5, 14, 9, 10]. Since we anticipate only a small number of side constraints, it suffices to use a repeated coordinate search. Given an optimal or near-optimal $\boldsymbol{\lambda}$, we obtain the Lagrangian static bound by carrying out one backward longest-path calculation in the time-expanded graph from node $\hat{n}$ to all nodes $n \in \mathcal{N}$ using the Lagrangian modified arc reward $\mathbf{c} - \boldsymbol{\lambda}\mathbf{R}$. More specifically, an arc $(n, n') = (\langle v, s\rangle, \langle v', s+1\rangle) \in \mathcal{A}$, $s = 1, 2, ..., T-1$, is endowed with the reward

$$\tilde{c}_{n,n'} = c_{n,n'} - \sum_{i \in \mathcal{I}} \lambda_i r_{i,n,n'}, \tag{II.23}$$

where $c_{n,n'}$ is given by (II.15).

We also derive and implement a Lagrangian directional static bound similar to the one in Subsection B.2c. However, the derivation is a straightforward combination of Subsection B.2c and the approach described above. Hence, we omit it. This derivation results in a similar Lagrangian problem to the one in (II.21), but now defined on the node-and-time expanded network. We still refer to the Lagrangian multiplier as $\boldsymbol{\lambda}$ and the Lagrangian upper bound as $z(\boldsymbol{\lambda})$. We denote the Lagrangian directional static bound computed in this way by $\tilde{\delta}_0(v, v', t)$. Since the Lagrangian directional static bound is at least as strong as the Lagrangian static bound, we carry out the Lagrangian relaxation only in the node-and-time expanded network to find a Lagrangian multiplier $\boldsymbol{\lambda} \geq \mathbf{0}$.

2. Network Reduction

We propose and examine three techniques for reducing the size of the network prior to application of a branch-and-bound procedure. First, we use dominance rules to eliminate edges that cannot be on an optimal path. Second, we describe the application of "preprocessing" techniques frequently used prior to solving constrained shortest-path problems. Third, we modify the procedure described in Subsection B.2d.

a. Edge Dominance

There are several situation where a vertex $v' \in \mathcal{F}(v)$ can be eliminated as candidate for visit from vertex v. Such "dominance tests" are case dependent, but can be effective in reducing the number of edges. We describe one situation where we use "edge dominance."

In many practical situations, there are two resources: risk and fuel. If higher altitude implies lower risk and lower glimpse detection probability, and climbing to higher altitude consumes more fuel than level flight, then we can eliminate some edges in the graph $(\mathcal{V}, \mathcal{E})$ by edge dominance. Suppose that $f_1(v, v', t)$ and $f_2(v, v', t)$ represent risk and fuel, respectively. Also suppose that the risk $f_1(v, v', t) = f_1(v')$, i.e., only depends on v'. Let $\psi(v)$ be the altitude of waypoint $v \in \mathcal{V}$. Then, if we have the above described situation, we use the following (one-step) procedure to reduce the size of the graph $(\mathcal{V}, \mathcal{E})$.

Edge Dominance Procedure R2.

Step 1. Delete any edge $(v, v') \in \mathcal{E}$ that satisfies $f_1(v') = 0$ and $\psi(v) < \psi(v')$.

We note that Procedure R2 takes advantage of the fact that if there is no risk at v', then there is no need to increase altitude when moving from v to v'. The altitude can be increased later if need be.

b. Preprocessing

It is well known that the side constraints (II.19) can be used, prior to any main calculations, to identify nodes and arcs in the time-expanded network $(\mathcal{N}, \mathcal{A})$ that cannot lie on any feasible path [4, 15, 9, 10]. Such nodes and arcs can be eliminated from $(\mathcal{N}, \mathcal{A})$, which reduces the size of the problem that needs to be considered when computing bounds and carrying out branching. Moreover, the reduction of the time-expanded network typically also strengthens the Lagrangian relaxation, i.e., reduces the gap $\bar{z} - z^*$, (see (II.21) and (II.20)), and, hence, reduces the need for branching. We adopt the follow procedure, adapted from [9, 10], to carry out arc preprocessing:

Preprocessing Procedure R3.

Input: A time-expanded network $(\mathcal{N}, \mathcal{A})$ and arc lengths $r_{n,n'}$.

Output: The updated (or reduced) time-expanded network $(\mathcal{N}, \mathcal{A})$.

Step 1. Set number of iterations $\bar{k}$ and $k = 1$.

Step 2. For all $i \in \mathcal{I}$ and $n \in \mathcal{N}$, compute a minimum-weight n_0-n subpath distance $\underline{R}_i(n)$ and a minimum-weight n-$\hat{n}$ subpath distance $\underline{r}_i(n)$ in $(\mathcal{N}, \mathcal{A})$ with respect to weights $r_{i,n,n'}$.

Step 3. Delete any arc $(n, n') \in \mathcal{A}$ with

$$\underline{R}_i(n) + r_{i,n,n'} + \underline{r}_i(n') > \hat{r}_i \text{ for any } i \in \mathcal{I}. \tag{II.24}$$

Step 4. If $k < \bar{k}$ and at least one arc was deleted in Step 3, replace k by $k+1$, and go to Step 2. Else, stop.

If a lower bound on the probability of detection is available, we also carry out similar preprocessing with respect to arc reward $c_{n,n'}$ and the Lagrangian modified arc reward $\tilde{c}_{n,n'} = c_{n,n'} - \sum_{i \in \mathcal{I}} \lambda_i r_{i,n,n'}$, see [9, 10].

We describe the preprocessing procedure for the time-expanded network and argue that it improves the Lagrangian static bound. However, the same methodology applies to the node-and-time expanded network and it improves the

Lagrangian directional static bound. In our main algorithm (described in Subsection C.3), we also apply preprocessing to the node-and-time expanded network and denote that procedure R3′.

c. Vertex Dominance for Distant Target

As in Subsection B.2d, we consider the case where the searcher's and the target's locations are initially some distance apart and derive a network reduction procedure that takes advantage of that situation. In contrast to Subsection B.2d, the subpath used to reach a node-of-first-contact is now important since the resource consumption along different subpaths may be different. Hence, $\langle v_0, 0\rangle$ now needs to be connected to each node-of-first-contact $\langle v_s, s\rangle$ with multiple "jump" arcs representing the different possible subpaths and resource consumptions used to reach $\langle v_s, s\rangle$. A standard path enumeration algorithm (see, e.g., [11]) can enumerate the different subpaths, at least as long as s is relatively small. Multiple arcs to a node-of-first-contact can also be used to model the situation with edge-dependent glimpse detection probability, a case ignored in Subsection B.2d.

After all the arcs are generated to all the nodes-of-first-contact, a number of them can be deemed uninteresting and can be eliminated using dominance rules of the form: If an arc from $\langle v_0, 0\rangle$ to $\langle v_s, s\rangle$ has no larger probability of detection and no smaller consumption of each resource as another parallel arc and the two arcs are not identical, the first arc is dominated and can be eliminated. In sets of identical parallel arcs, we also eliminate all but one. Trivially, arcs with resource consumption greater than the specified limits are also removed. Moreover, if a lower bound on the optimal probability of detection exists, it can be used to eliminate more arcs as described in the following.

Below we formally describe this network reduction procedure based on vertex dominance for a distant target. We note that the procedure is more effective after (i) applying network reduction procedures R2, R3 and R3′, (ii) finding $\boldsymbol{\lambda}$ that (approximately) optimizes the Lagrangian upper bound $z(\boldsymbol{\lambda})$ on the node-and-time

expanded network, and (iii) computing the directional static bound $\delta_0(v, v', t)$ and the Lagrangian directional static bound $\tilde{\delta}_0(v, v', t)$ for each node $\langle v', t\rangle \in \mathcal{N}$. So we assume that these calculations have been carried out. During these calculations, feasible paths may be obtained. Such paths provide lower bounds on the optimal probability of detection. Let $\hat{q}$ denote the largest lower bound found so far.

Vertex Dominance Procedure R4.

Input: A time-expanded network $(\mathcal{N}, \mathcal{A})$, arc lengths $c_{n,n'}$ and $r_{n,n'}$.

Output: The nodes-of-first-contact $\langle v_s, s\rangle$ which are not dominated by others.

Step 1. Find all nodes-of-first-contact $\langle v_s, s\rangle \in \mathcal{N}$. If none exist with $s > 1$, then stop.

Step 2. For each node-of-first-contact $\langle v_s, s\rangle$, enumerate all subpaths $\langle v_0, 0\rangle$ to $\langle v_s, s\rangle$.

Step 3. For each node-of-first-contact $\langle v_s, s\rangle$ and subpath $\mathcal{P} = \{v_l\}_{l=0}^{s}$, carry out the tests:

If any of the following is true, then eliminate $\mathcal{P}$:

(i) For some remaining $\langle v_0, 0\rangle$-$\langle v_s, s\rangle$ subpath $\mathcal{P}'$, $r_i(\mathcal{P}) \geq r_i(\mathcal{P}')$ for all $i \in \mathcal{I}$ and $q(\mathcal{P}) \leq q(\mathcal{P}')$.

(ii) $q(\mathcal{P}) + \delta_0(v_{s-1}, v_s, s) \leq \hat{q}$.

(iii) $r_i(\mathcal{P}) + \underline{r}_i(\langle v_s, s\rangle) > \hat{r}_i$ for some $i \in \mathcal{I}$.

Step 3 can also be augmented with a test on the Lagrangian-modified probability of detection using $\tilde{\delta}_0(v, v', t)$ if a near-optimal multiplier $\boldsymbol{\lambda}$ is available.

3. Algorithm

We now state the complete algorithm for RSSP. The algorithm starts with network reductions procedures R2, R3, and R3′. The next step solves the Lagrangian problem (II.22) and determines a near-optimal $\boldsymbol{\lambda}$. (The calculations are actually carried out in the node-and-time expanded network as we prefer to use the Lagrangian directional static bound.) If a feasible path becomes available during the procedures described above, network reduction procedure R3′ is repeated now using checks with

respect to arc reward $c_{\xi,\xi'}$ and its Lagrangian modified arc reward. The next steps are to compute the directional static bound of Subsection B.2c (i.e., $\delta_0(v, v', t)$), the Lagrangian directional static bound as described in Subsection C.1 (i.e., $\tilde{\delta}_0(v, v', t)$), and bounds on resource consumption along any path extension (i.e., $\underline{r}_i(n)$, $i \in \mathcal{I}$). The final steps before the branch-and-bound procedure is to implement network reduction procedure R4.

We implement the branch-and-bound procedure as an implicit path-enumeration in the time-expanded network. The procedure amounts to a depth-first search coupled with optimality and feasibility checks using the computed bounds, which follows [9, 10]. The complete algorithm takes the following form:

Algorithm 3 (Solves RSSP).

Input: A time-expanded network $(\mathcal{N}, \mathcal{A})$ with arc lengths $c_{n,n'}$ and $r_{n,n'}$ and a node-and-time expanded network (Ξ, Ω) with arc lengths $c_{\xi,\xi'}$ and $r_{\xi,\xi'}$. The initial target distribution $p(\cdot, 1)$, Markov transition matrix Γ, and upper bound $\hat{q}$.

Output: An optimal search path $\mathcal{P}^* = \{v_l\}_{l=0}^{T}$ and its value q^*.

Step 1. Apply network reduction procedures R2, R3 and R3′.

Step 2. Find $\boldsymbol{\lambda}$ that approximately optimizes the Lagrangian upper bound $z(\boldsymbol{\lambda})$ in the node-and-time expanded graph (Ξ, Ω). If a feasible solution is found, set the probability of detection on the corresponding path equal to $\hat{q}$. Otherwise, set $\hat{q} = -\infty$.

Step 3. If a feasible solution is found so far, implement R3′ also with respect to arc reward and Lagrangian modified arc reward using $\hat{q}$.

Step 4. Ignoring side constraints, compute the directional static bound $\delta_0(v, v', t)$ for all expanded nodes $\xi = \langle n, n' \rangle$ in (Ξ, Ω), with $n = \langle v, t-1 \rangle$, $n' = \langle v', t \rangle$, $v, v' \in \mathcal{V}$.

Step 5. Using $\boldsymbol{\lambda}$ from Step 2, compute the Lagrangian directional static bound $\tilde{\delta}_0(v, v', t)$ for all expanded nodes $\xi = \langle n, n' \rangle$ in (Ξ, Ω), with $n = \langle v, t-1 \rangle$, $n' = \langle v', t \rangle$, $v, v' \in \mathcal{V}$.

Step 6. For each $i \in \mathcal{I}$, compute the minimum distance $\underline{r}_i(n)$ from each node $n \in \mathcal{N}$ back to $\hat{n}$ by solving a single, backwards, shortest-path problem in the time-expanded graph $(\mathcal{N}, \mathcal{A})$ starting from $\hat{n}$ using arc lengths $r_{i,n,n'}$.

Step 7. Apply network reduction procedure R4.

Step 8. Apply a standard path-enumeration procedure (e.g., [11]) in $(\mathcal{N}, \mathcal{A})$ with the following modifications:

(i) The path-enumeration commences from n_0, but extends a current subpath $\{n_l\}_{l=0}^{t}$ along arc $(n_t, n) = (\langle v_t, t\rangle, \langle v, t+1\rangle)$ if and only if the following conditions hold:

- For all $i \in I$, $\{n_0, n_1, ..., n_t, n\}$ can be extended to a path whose i-th resource does not exceed $\hat{r}_i$, i.e.,

$$r_i(\{n_0, n_1, ..., n_t, n\}) + \underline{r}_i(n) \leq \hat{r}_i. \tag{II.25}$$

- $\{n_0, n_1, ..., n_t, n\}$ can be extended to a path with probability of detection exceeding $\hat{q}$, i.e.,

$$q(\{n_0, n_1, ..., n_t, n\}) + \delta_0(v_t, v, t+1) > \hat{q}. \tag{II.26}$$

- $\{n_0, n_1, ..., n_t, n\}$ can be extended to a path whose Lagrangian-modified probability is no less than $\hat{q}$, i.e.,

$$\begin{aligned} &q(\{n_0, n_1, ..., n_t, n\}) - \sum_{i\in\mathcal{I}} \sum_{l=1}^{t} \lambda_i r_{i,n_{l-1},n_l} \\ &- \sum_{i\in\mathcal{I}} \lambda_i r_{i,n_t,n} + \boldsymbol{\lambda}\hat{\mathbf{r}} + \tilde{\delta}_0(v_t, v, t+1) \geq \hat{q}. \end{aligned} \tag{II.27}$$

(ii) Whenever the algorithm identifies a path $\mathcal{P}$ with $q(\mathcal{P}) > \hat{q}$ and $r_i(\mathcal{P}) \leq \hat{r}_i, i \in \mathcal{I}$, replace $\hat{q}$ by $q(\mathcal{P})$.

In Step 8, the checks (II.25), (II.26), and (II.27) prevent the enumeration of paths that can be determined, using the computed bounds, to not be optimal. Specifically, (II.25) prevents the extension of subpaths that cannot result in a feasible path with respect to the side constraints. Since $\delta_0(v_t, v, t+1)$ is a valid upper bound on the probability of detection during time periods $t+2$, $t+3$, ..., T, the left-hand side of (II.26) is an upper bound on the probability of detection along any path that starts with the subpath $\{n_0, n_1, ..., n_t, n\}$. Hence, the subpath cannot be extended to a path with larger probability of detection than $\hat{q}$ if (II.26) fails. In (II.27), the probability of detection along $\{n_0, n_1, ..., n_t, n\}$ is modified by Lagrangian terms and

the Lagrangian directional static bound is used. The resulting check can be shown to be valid using standard Lagrangian relaxation theory and the argument above.

We also implement Step 8 with a "branching strategy" based on the Lagrangian directional static bound. Specifically, we first consider extending the current subpath $\{n_l\}_{l=0}^{t}$ along the arc (n_t, n) with the largest Lagrangian directional static bound among all the nodes in the forward star of n_t. Second, we consider extending $\{n_l\}_{l=0}^{t}$ with the node corresponding to the second largest Lagrangian directional static bound, etc. This branching strategy is analogous to the one in Step 3 of Algorithms 1 and 2. We have also experimented with using the directional static bound instead of the Lagrangian directional static bound and found it to usually result in comparable run times. However, the Lagrangian directional static bound appears faster, on average.

Since Algorithm 3, in the worst case, enumerates all feasible paths, it is guaranteed to find an optimal solution of RSSP.

D. NUMERICAL EXAMPLE

This section describes computational experiments with Algorithm 3 applied to RSSP with side constraints on risk exposure and fuel consumption. We carry out all experiments on the same computational platform as in Section B.

We consider a military planning situation where a UAV is assigned a mission to search and detect a high-value moving target. Planners wish to determine a flight path over the area of interest (AOI) that maximizes the probability of detecting the target. The UAV will start its path at a known waypoint with a known fuel supply, and will return to the same waypoint before the fuel tank is empty. Doctrine specifies that the UAV cannot be assigned a path with higher risk than a specific threshold. The AOI is partially under enemy control and any aircraft flying over the AOI could be shot down by enemy surface-to-air missiles (SAMs), anti-aircraft artillery, and small-arms fire. Flying at a high altitude would reduces that risk, but it will also reduce the

quality of UAV's sensor readings. Consequently, the UAV may change altitude during the course of the mission to balance risk and sensor quality. Changing from low to high altitude consumes more fuel than level flight. Hence, the number of time periods available for search depends on the fuel consumption and therefore the vertical flight profile.

We model this situation by dividing the AOI into 10 by 10 cells (see Figure 6). The airspace over each cell is vertically discretized into two altitudes ("low and high"). The heavily shaded cells ($\mathcal{C}_1$) in Figure 6 represent an urban area over which the UAV's risk is high and its glimpse detection probability is low. The unshaded cells ($\mathcal{C}_3$) represent open terrain where there is no risk and the UAV's glimpse detection probability is high. The lightly shaded cells ($\mathcal{C}_2$) represent an area with intermediate risk and intermediate glimpse detection probability. We note that, while this problem instance does not directly relate to any real-world operational situation, we believe that it illustrate a fairly practical situation.

We assume that the risks at different edges along a path are independent. If the probability of the UAV surviving edge $(v, v') \in \mathcal{A}$ is $\sigma(v, v')$, then the probability of surviving the path $\{v_t\}_{t=0}^{k}$ is simply $\Pi_{l=1}^{k}\sigma(v_{l-1}, v_l)$. Let $\hat{\sigma}$ be a lower limit on the survival probability. Then, a standard logarithmic transformation leads to the following constraint

$$\sum_{l=1}^{k} -\log \sigma(v_{l-1}, v_l) \leq -\log \hat{\sigma}, \tag{II.28}$$

which is in the form (II.3) and (II.4) with $f_i(v_{l-1}, v_l, l) = -\log \sigma(v_{l-1}, v_l)$ and $\hat{r}_i = -\log \hat{\sigma}$.

For this computational experiment, we assume that the glimpse detection probability and the survival probability for an edge $(v, v') \in \mathcal{E}$ depend only on the cell and altitude corresponding to vertex $v' \in \mathcal{V}$ as listed in Table 4. We note that glimpse detection probability at high altitude is assumed to be 70% of that at low altitude and the failure probability (complement of the survival probability) at high altitude is 30% of that at low altitude.

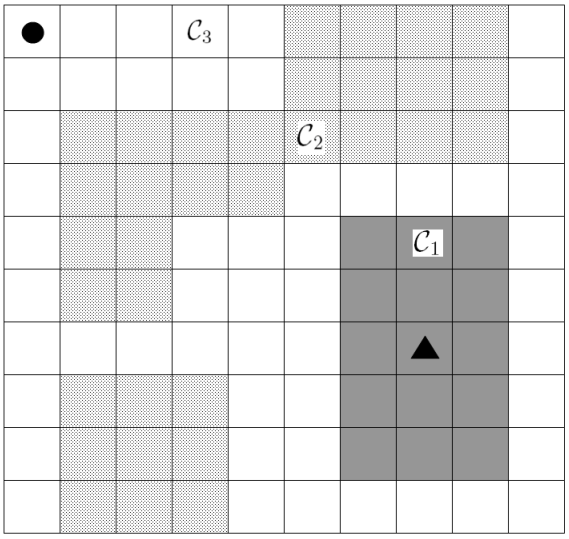

Figure 6. An area of interest composed of 10 by 10 cells and two altitudes. Heavily shaded cells ($\mathcal{C}_1$), lightly shaded cells ($\mathcal{C}_2$), and unshaded cells ($\mathcal{C}_3$) describe risky, moderately risky, and non-risky area, respectively. The circle indicates the cell over which searcher starts, and the triangle specifies the initial position of the target.

The UAV enters the airspace at high altitude over the northwest cell (cell 1; cells are numbered from left to right, and from top to bottom) and will return to the same cell at either high or low altitude at the end of the mission. The searcher is located at one vertex each time period and searches the corresponding cell. For the next time period, the searcher can stay at the same vertex, change altitude over the same cell, or move to a vertex (at any altitude) corresponding to a vertically or horizontally adjacent cell. The maximum number of time periods is $T = 40$, but the fuel-consumption constraint may limit the number of periods to less than 40. We assume that the fuel consumption at each time step is as follows: 10 units if there is no altitude change, 12(9) units if changing from low(high) altitude to high(low) altitude. The initial position of the target is the center of the high risk region (cell

Cells	Altitude	Glimpse probability	Survival probability
$\mathcal{C}_1$	low	0.20	0.960
	high	0.14	0.988
$\mathcal{C}_2$	low	0.40	0.980
	high	0.28	0.994
$\mathcal{C}_3$	low	0.60	1.000
	high	0.42	1.000

Table 4. Glimpse detection probability $g(v, v', t)$ and survival probability $\sigma(v, v')$ for different cells and altitude.

68). The target remains in the current cell with a probability $\rho = 0.6$ for the next time period or moves to one of the vertically or horizontally adjacent cells with equal probability.

The survival-probability limit is a threshold that is set by the commander or planner. A search path with lower survival probability than the threshold would not be accepted. In this experiment, we consider the survival probability limits 0.95, 0.90,...,0.70, and fuel consumption limit 300, 325, ..., 450.

We solve this problem instance using Algorithm 3. Tables 5, 6 and 7 report computational results for different combinations of survival probability and fuel limits for the UAV. When the fuel limit is tight (e.g., 300 and 325), the UAV cannot operate for the full duration of 40 time steps. We observe that increasing the fuel limit beyond 425 does not increase the probability of detection as the time limit of 40 periods becomes active. The average run time is 580 seconds, with a standard deviation of 792. All problem instances are solved within one hour and typically in much less. Figure 7 shows the optimal path given survival probability limit 0.90 and fuel limit 400. The solid lines and the dashed lines represent flight segment at low and high altitude, respectively.

We also consider the case with edge-dependent glimpse detection probability. Consider the same situation as earlier described, but now assume that a move to a new waypoint results in a lower glimpse detection probability than if the searcher already

was at that waypoint. Specifically, if $v = v'$, we let the glimpse detection probability $g(v, v', t)$ be as in Table 4; otherwise we replace $g(v, v', t)$ by $0.1g(v, v', t)$. Figure 8 shows an optimal path found given survival probability and fuel limits of 0.90 and 400, respectively. In contrast to the case with edge-independent glimpse detection probability (Figure 7), the searcher now tends to stay for multiple time periods at the same waypoints in high-probability regions to reap the benefits of the corresponding high glimpse detection probability. The run times (not reported in detail) for the case with edge-dependent glimpse detection probabilities are, on average, 53 seconds, with a standard deviation of 71 seconds. The reduction in run time compared to the edge-independent case is caused by the often lower glimpse detection probability $(0.1g(v, v', t))$, which tightens the bound.

	Survival prob. limit = 0.95					Survival prob. limit = 0.90			
Fuel limit	Prob. Detection	Survival Prob.	Fuel	Run time (sec.)	Fuel limit	Prob. Detection	Survival Prob.	Fuel	Run time (sec.)
300	0.131501	0.962466	300	26.89	300	0.135098	0.915157	300	29.58
325	0.153228	0.952892	321	98.20	325	0.166933	0.901925	322	32.66
350	0.176511	0.952892	350	3313.64	350	0.194440	0.915157	350	115.94
375	0.204686	0.952892	371	661.43	375	0.217277	0.901925	372	84.00
400	0.236651	0.962466	400	664.96	400	0.246767	0.902268	400	537.71
425	0.237081	0.952892	401	2721.68	425	0.249996	0.901925	402	361.60
450	0.237081	0.952892	401	2724.22	450	0.249996	0.901925	402	361.52

Table 5. Computational results (probability of detection, survival probability, fuel consumption and run times) for Algorithm 3 with survival probability limit = 0.95 and 0.90.

	Survival prob. limit = 0.85					Survival prob. limit = 0.80			
Fuel limit	Prob. Detection	Survival Prob.	Fuel	Run time (sec.)	Fuel limit	Prob. Detection	Survival Prob.	Fuel	Run time (sec.)
300	0.145309	0.851528	300	25.31	300	0.145809	0.805953	299	22.03
325	0.173000	0.851528	320	37.81	325	0.173218	0.839535	319	37.39
350	0.203183	0.851528	350	67.69	350	0.204891	0.805953	349	64.17
375	0.222393	0.851528	370	115.24	375	0.223377	0.805953	369	116.69
400	0.255481	0.851528	400	510.68	400	0.255813	0.827710	399	880.65
425	0.255481	0.851528	400	508.50	425	0.255813	0.827710	399	880.40
450	0.255481	0.851528	400	508.96	450	0.255813	0.827710	399	880.46

Table 6. Computational results (probability of detection, survival probability, fuel consumption and run times) for Algorithm 3 with survival probability limit = 0.85 and 0.80.

	Survival prob. limit = 0.75					Survival prob. limit = 0.70			
Fuel limit	Prob. Detection	Survival Prob.	Fuel	Run time (sec.)	Fuel limit	Prob. Detection	Survival Prob.	Fuel	Run time (sec.)
300	0.150092	0.753377	300	21.33	300	0.150277	0.742767	299	21.38
325	0.175850	0.753377	320	37.50	325	0.176068	0.742767	319	31.16
350	0.205935	0.753377	350	63.84	350	0.206200	0.742767	349	53.42
375	0.223703	0.753377	370	125.97	375	0.224003	0.713056	369	121.33
400	0.255813	0.827710	399	1220.30	400	0.255813	0.827710	399	1280.52
425	0.255813	0.827710	399	1217.21	425	0.255813	0.827710	399	1277.52
450	0.255813	0.827710	399	1218.91	450	0.255813	0.827710	399	1282.42

Table 7. Computational results (probability of detection, survival probability, fuel consumption and run times) for Algorithm 3 with survival probability limit = 0.75 and 0.70.

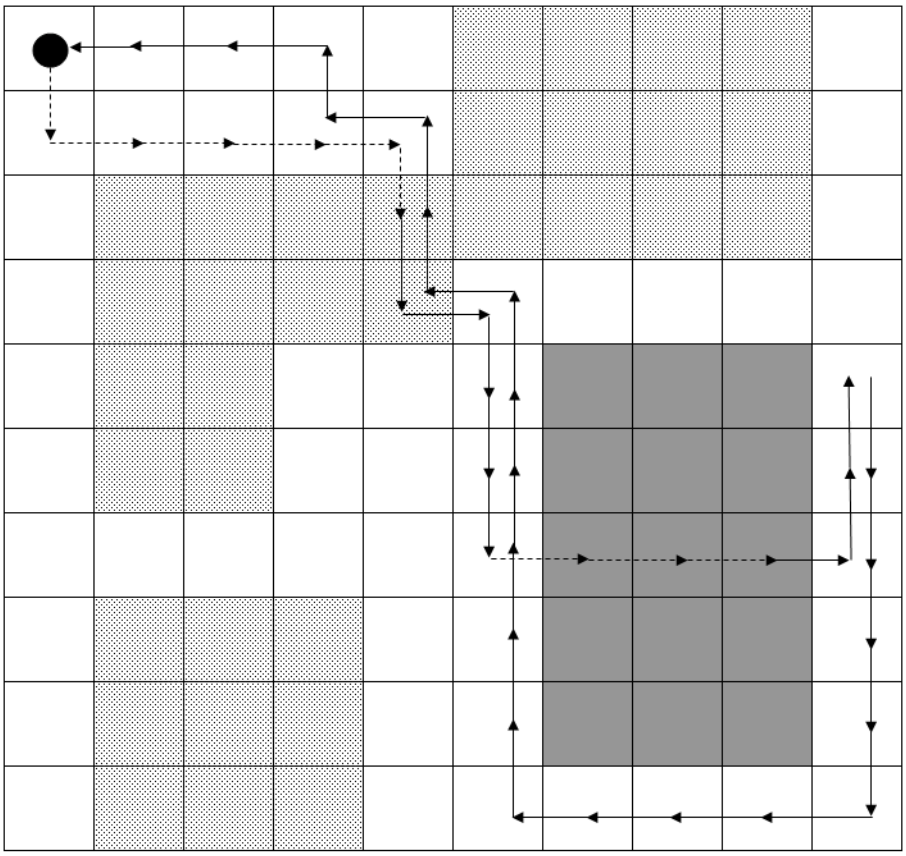

Figure 7. An optimal path for survival probability limit 0.90 and fuel limit 400. The solid lines and the dashed lines represent flight segments at low and high altitude, respectively. (See Figure 6 for a description of the underlying figure.)

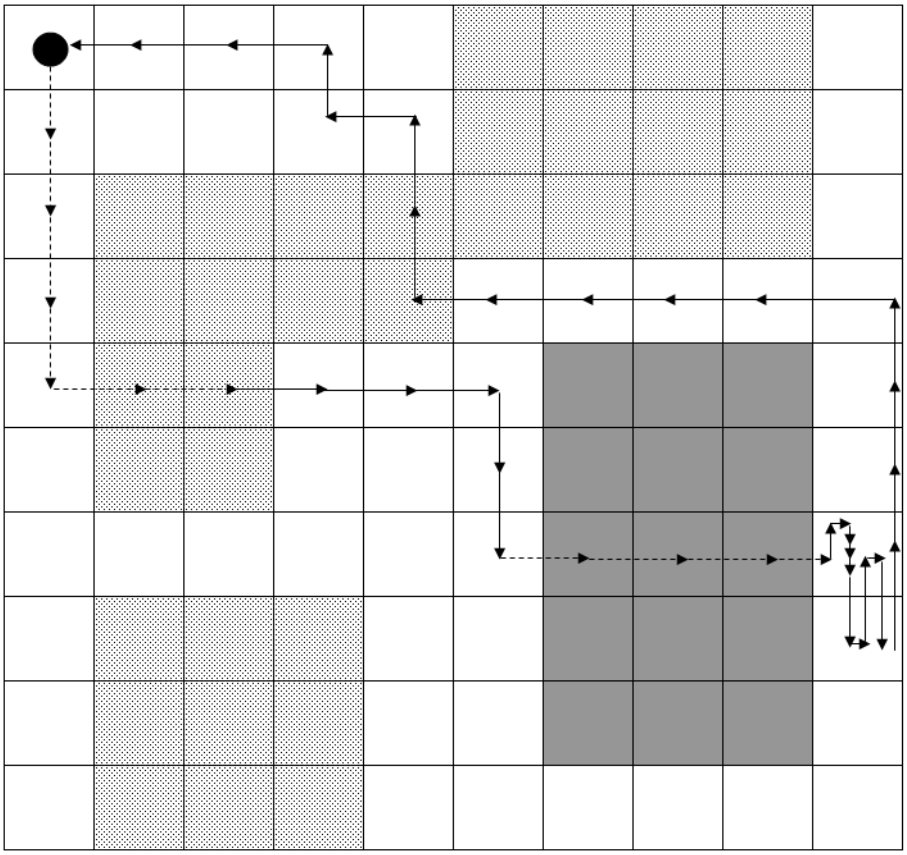

Figure 8. An optimal path for a case with edge-dependent glimpse probability, survival probability limit 0.90, and fuel limit 400. The solid lines and the dashed lines represent flight segments at low and high altitude, respectively. (See Figure 6 for a description of the underlying figure.)

THIS PAGE INTENTIONALLY LEFT BLANK

III. MULTIPLE-SEARCHER PROBLEM

This chapter addresses the optimization of multiple searchers with path- and time-constraints (MSP). We aim to determine a path for each searcher that maximize the probability of detecting a moving target within a mission time by at least one searcher. We refer to such a set of paths as a "search plan." We start by describing and formulating MSP. Then, we present three algorithms (an exact procedure and two heuristics) to find an optimal, near-optimal, or "good" search plan. The chapter also includes computational studies.

A. PROBLEM DESCRIPTION AND FORMULATION

The multiple-searcher problem (MSP) is an extension of SP which considers a finite set of searchers $\mathcal{J} = \{1, 2, ..., J\}$. In this section, for ease of reference, we formulate MSP by retracing the relevant portions of Section A in Chapter II.

The area of interest is discretized into a finite set of cells $\mathcal{C} = \{1, \ldots, C\}$ and the time horizon is discretized into a finite set of time periods $\mathcal{T} = \{1, 2, ..., T\}$. A target occupies one cell each time period and moves among cells according to a Markov process with known transition matrix Γ. Let $p(\cdot, t) = [p(1, t), p(2, t), \ldots, p(C, t)]$, where $p(c, t)$ is the probability that the target is in cell $c \in \mathcal{C}$ at the beginning of time period $t \in \mathcal{T}$ and the target has not been detected before t by any searcher. We refer to $p(\cdot, t)$ as the undetected target distribution. The initial target distribution $p(\cdot, 1)$ is known.

The multiple searchers move through a designated airspace over the area of interest with the goal of finding the moving target on the ground. The airspace over each cell $c \in \mathcal{C}$ is vertically discretized into a set of altitudes $\mathcal{H} = \{1, 2, \ldots, H\}$. For any $c \in \mathcal{C}$ and $h \in \mathcal{H}$, we refer to the cell-altitude pair $\langle c, h \rangle$ as a waypoint where the searchers can loiter and carry out search of cell c. We model the designated airspace by a directed network $(\mathcal{V}, \mathcal{E})$, with set of vertices $\mathcal{V}$ and set of directed edges $\mathcal{E}$, in

which vertices $v = \langle c, h \rangle \in \mathcal{V}$ represent waypoints and directed edges $e = (v, v') \in \mathcal{E}$ represent transition between waypoints $v, v' \in \mathcal{V}$. The searchers can only transit between two waypoints that are located physical adjacent to each other. Let $\mathcal{F}(v) \subset \mathcal{V}$ be the set of vertices that are adjacent to $v \in \mathcal{V}$. We adopt the convention that $v \in \mathcal{F}(v)$ for all $v \in \mathcal{V}$. Then, the set of edges $\mathcal{E} = \{(v, v') \in \mathcal{V} \times \mathcal{V} |\ v' \in \mathcal{F}(v)\}$.

During each time period $t \in \mathcal{T}$, each searcher is located at a particular vertex (waypoint). Any number of searchers can occupy a vertex in the same time period. We also assume that there is no transit time between waypoints. Hence, $(v, v') \in \mathcal{E}$ simply represents search at waypoint v followed by search at waypoint v' in the next time period. We note that the edge $(v, v) \in \mathcal{E}$ represents searching at waypoint v for two consecutive time periods.

Let $\phi : \mathcal{V} \rightarrow \mathcal{C}$ be the function that specifies the cell over which a vertex is located, i.e., cell $\phi(v)$ is searched from vertex v. We denote the initial vertex (waypoint) of searcher $j \in \mathcal{J}$ prior to time period 1 by $v_0^j \in \mathcal{V}$. We also define $\hat{\mathcal{V}}^j \subset \mathcal{V}$ to be a set of possible destination vertices for searcher $j \in \mathcal{J}$. When we describe an arbitrary searcher, we may omit the superscript and simply write v_0 and $\hat{\mathcal{V}}$.

For any $k \in \mathcal{T}$ and $v_l \in \mathcal{V}$, $l = 0, 1, 2, ..., k$, such that $(v_{l-1}, v_l) \in \mathcal{E}$ for all $l = 1, 2, ..., k$, let the sequence $\mathcal{P} = \{v_l\}_{l=0}^k$ denote a directed v_0-v_k subpath. If $v_k \in \hat{\mathcal{V}}$, then the directed v_0-v_k subpath is a directed v_0-v_k path. In this notation, a searcher flies from v_0 to some $v_k \in \hat{\mathcal{V}}$ along a directed v_0-v_k path. For a specific searcher $j \in \mathcal{J}$, we denote its directed v_0-v_k (sub)path as $\mathcal{P}^j = \{v_l^j\}_{l=0}^k$. Thus a search plan for the fleet of searchers is described as $\overline{\mathcal{P}} = (\mathcal{P}^1, \mathcal{P}^2, ..., \mathcal{P}^J)$.

We adopt the following target-detection model. If the target is in cell $c \in \mathcal{C}$ during time period $t \in \mathcal{T}$ and only searcher $j \in \mathcal{J}$ is at the same time at a waypoint above cell c, i.e., $\phi(v_t^j) = c$, then detection occurs with a glimpse detection probability $g^j(v_{t-1}^j, v_t^j, t)$, where $v_{t-1}^j \in \mathcal{V}$ is the waypoint of searcher j during time period $t - 1$. Hence, the glimpse detection probability during time period t depends on the previous and current waypoints for the searcher. We assume that the glimpse

detection probability of each searcher is mutually independent. In this notation, the probability of detecting the target in cell $c \in \mathcal{C}$ by any searcher during time period $t \in \mathcal{T}$ and no prior detections becomes

$$Q(c,t) = p(c,t)\left[1 - \prod_{j \in \mathcal{J} | \phi(v_t^j) = c} (1 - g^j(v_{t-1}^j, v_t^j, t))\right], \tag{III.1}$$

We refer to $Q(\cdot,t) = [Q(1,t), Q(2,t), \ldots, Q(C,t)]$ as the reward vector at time period t.

Since $p(\cdot,t)$ is the undetected target distribution at the beginning of time period t, it depends on searches up to time period $t-1$. Specifically, if $p(\cdot,t)$ is given and each searcher searches a cell from its waypoint during time period $t \in \mathcal{T}$, the undetected target distribution at the beginning of the next time period $t+1$ is

$$p(\cdot,t+1) = [p(\cdot,t) - Q(\cdot,t)]\Gamma. \tag{III.2}$$

Thus, the probability of detection for search plan $\overline{\mathcal{P}}$ is defined as

$$q(\overline{\mathcal{P}}) = \sum_{t=1}^{T}\sum_{c=1}^{C} Q(c,t). \tag{III.3}$$

We refer to the problem of maximizing (III.3) as the multiple-searcher problem (MSP).

B. ALGORITHMS FOR MSP

We develop one exact algorithm and two heuristics to solve MSP. The exact algorithm is a straightforward extension of the branch-and-bound (B/B) algorithm in Chapter II. However, the B/B algorithm for solving MSP requires an expanded network structure to account for multiple searchers. Thus, in each time period $t \in \mathcal{T}$, we consider the combined location $V_t = (v_t^1, v_t^2, \ldots, v_t^J)$ of all searchers. We call this combined location a configuration. We treat a configuration in the same way we treated a vertex in SP, and construct a time-expanded (configuration) network. The B/B algorithm for MSP is implemented in this time-expanded network. In that network, a search plan is simply a sequence of configurations $\{V_l\}_{l=0}^{k}$, $k \in \mathcal{T}$. We also

use the notation $\{n_l\}_{l=0}^{k}$ where $n_l = \langle V_l, l \rangle$. Thus we refer to a search plan also as a "configuration path." When the meaning is clear from the context, we refer to a configuration path as a path.

We present two heuristics to solve MSP. The first one is a variant of the expected detection heuristic (H1) in [13]. In H1, a configuration path (search plan) is generated by iteratively extending the current configuration subpath by the first arc in the longest-path, with respect to the expected number of detections, from the current configuration to the last time period. We note that a longest-path calculation is required every time the current configuration subpath is extended. Hence, H1 is similar to Algorithm 1 with dynamic bounds (see Chapter II) as they both require repeated longest-path calculations. As an alternative approach, we present a similar heuristic corresponding to the static bound (see Chapter II). We denote our approach the static bound heuristic (SBH).

The second heuristic algorithm is using the Cross-Entropy (CE) method [38, 39]. The CE method involves an iterative procedure where each iteration composes of two phases. First, the method generates random samples (i.e., search plans in our case) according to a probability distribution. Second, the method updates the parameters in the probability distribution based on a subset of the "best" samples, the so-called "elite" samples. This process increases the chances of an optimal or near-optimal solution to appear within the next set of samples. This dissertation appears to be the first one studying the CE method in the context of search problems.

We can assume without loss of generality that each searcher's path consists of $T + 1$ vertices. For simplicity of notations, we also assume that: (1) there is no end-point restriction for any searcher $j \in \mathcal{J}$, i.e., $\hat{\mathcal{V}}^j = \mathcal{V}$; (2) the glimpse detection probability $g^j(v, v', t)$, $j \in \mathcal{J}$ is independent of v (i.e., $g^j(v, v', t) = g^j(v', t)$); (3) the airspace has only one altitude, i.e., there is only one vertex corresponding to each cell in the area of interest. It is straightforward to generalize the proposed algorithms to account for situations without these assumptions.

1. Branch-and-Bound Algorithm

In Chapter II, we presented a specialized branch-and-bound (B/B) procedure for solving SP. We utilize the same enhancement schemes to develop a B/B algorithm for MSP. However, the B/B algorithm for solving MSP requires an expanded network structure to account for multiple searchers. We consider a directed graph $(\mathcal{V}_c, \mathcal{E}_c)$ where $\mathcal{V}_c$ is the set of possible configurations and $\mathcal{E}_c$ is the set of directed edges representing possible transitions in one time period between two configurations. Thus a search plan can be described as $\overline{\mathcal{P}} = \{V_l\}_{l=0}^{T}$ where $(V_{l-1}, V_l) \in \mathcal{E}_c$ for all $l = 1, 2, ..., T$. We note that a search plan is also described (in the previous section) as $\overline{\mathcal{P}} = (\mathcal{P}^1, ...\mathcal{P}^j, ..., \mathcal{P}^J)$, where $\mathcal{P}^j = \{v_l^j\}_{l=0}^{T}$.

The description of the B/B algorithm for solving MSP follows the presentation of the algorithms for SP presented in Chapter II. Given a configuration subpath $\{V_l\}_{l=0}^{t-1}$, $t \in \mathcal{T}$, let $\mathcal{K}(t)$ be the set of triplets of the form $(V_t, t, \bar{q}(V_t, t))$ representing extensions of $\{V_l\}_{l=0}^{t-1}$ yet to be explored. The first element V_t refers to the next configuration to form, the second element t is the time period to form the configuration V_t, and the third element $\bar{q}(V_t, t)$ is an upper bound on the probability of detection along any configuration path that starts with the configuration subpath $\{V_l\}_{l=0}^{t}$. The upper bound $\bar{q}(V_t, t)$ consists of three parts. Let $d_t(V_t, t)$ be an upper bound on the probability of detection during time periods $t+1, t+2, ..., T$, given that the configuration of time t is V_t, and no detection occurs along the configuration subpath $\{V_l\}_{l=0}^{t}$. The two other parts are the probability of detection on the configuration subpath $\{V_l\}_{l=0}^{t-1}$ and the probability of detection during t. Hence,

$$\bar{q}(V_t, t) = q(\{V_l\}_{l=0}^{t-1}) + \sum_{c \in \mathcal{C}} Q(c, t) + d_t(V_t, t). \tag{III.4}$$

We also let $\hat{q}$ denote the largest detection probability found so far among all the examined configuration paths.

Consider a configuration subpath $\{V_l\}_{l=0}^t$, $t \in \mathcal{T}$, and let $p_g(\cdot, t)$ be the undetected target distribution after search along $\{V_l\}_{l=0}^t$, i.e.,

$$p_g(\cdot, t) = [p(\cdot, t) - Q(\cdot, t)]. \tag{III.5}$$

For any integer $s > t$, $s \in \mathcal{T}$, we also define

$$p_\Gamma(\cdot, s; t) = p_g(\cdot, t)\Gamma^{s-t}. \tag{III.6}$$

As seen, $p_\Gamma(c, s; t)$ is the probability that the target is in cell c at time period $s > t$ and there was no detection during search along the subpath $\{V_l\}_{l=0}^t$. Hence, target distribution $p_\Gamma(\cdot, s; t)$ at time period $s > t$ ignores the effect of search after time period t. If the subpath is $\{V_0\}$, i.e., $t = 0$, we define for notational convenience

$$p_\Gamma(\cdot, s; 0) = p(\cdot, 1)\Gamma^{s-1}, \tag{III.7}$$

for any $s > 0$, $s \in \mathcal{T}$. Moreover, we define $p_\Gamma(c, t; t) = 0$ for all $c \in \mathcal{C}$ and $t = 0, 1, ..., T$.

We construct a time-expanded configuration graph $(\mathcal{N}_c, \mathcal{A}_c)$ from the graph $(\mathcal{V}_c, \mathcal{E}_c)$ in the same manner as the development of the time-expanded network for SP in Chapter II (for details, we refer to Section B in Chapter II). For any integer $k \leq T + 1$ and nodes $n_l = \langle V_l, l\rangle \in \mathcal{N}_c, l = 0, 1, ..., k$, such that $(n_{l-1}, n_l) \in \mathcal{A}_c$ for all $l = 1, 2, ..., k$, we let the sequence $\{n_l\}_{l=0}^k$ denote a configuration subpath in the time-expanded configuration graph $(\mathcal{N}_c, \mathcal{A}_c)$.

For some $t \in \{0, 1, ..., T - 1\}$, suppose that a configuration subpath $\{V_l\}_{l=0}^t$ in the graph $(\mathcal{V}_c, \mathcal{E}_c)$ is given. We endow each arc $(n, n') = (\langle V, s\rangle, \langle V', s + 1\rangle) \in \mathcal{A}_c$, $s = t, t + 1, ..., T - 1$, in the time-expanded configuration graph $(\mathcal{N}_c, \mathcal{A}_c)$ with a "reward"

$$c_{n,n'} = \sum_{c' \in \mathcal{C}} \left[p_\Gamma(c', s + 1; t) - \sum_{c \in \mathcal{R}(c')} Q(c, s)\Gamma(c, c') \right] \left[1 - \prod_{j \in \mathcal{J} | v_{s+1}^j = c'} (1 - g^j(v_{s+1}^j, s + 1) \right] \tag{III.8}$$

where $\Gamma(c, c')$ is the c-c' element of the Markov transition matrix Γ. Thus this time-expanded configuration network $(\mathcal{N}_c, \mathcal{A}_c)$ has the same structure as the time-expanded

network $(\mathcal{N}, \mathcal{A})$ of Chapter II and the longest paths in this network provides the static bound $d_0(V_t, t)$. The B/B for MSP is implemented in this time-expanded configuration network $(\mathcal{N}_c, \mathcal{A}_c)$.

We consider generalizations of directional static bound and network reduction from Chapter II. To obtain a directional static bounds, we would need to generate a node-and-time expanded "configuration" network corresponding to the node-and-time expanded network for SP. Since the number of arcs in the node-and-time expanded configuration network becomes extremely large (e.g., a problem instance with 5 by 5 cells, 3 searchers, and time horizon 7 has more than one hundred million arcs), we realize that the directional static bound technique is not computationally practical for MSP. Hence, we adopt the static bound in our B/B algorithm for MSP. The difficulty of network size occurs also in the time-expanded configuration network when we consider large-size problems, which means our B/B algorithm is practical only for small-size problems.

It is straightforward to generalize the network-reduction technique based on the initial positions of the searchers and the target of Chapter II to MSP. The required "configurations"-of-first-contact $\langle V_s, s\rangle \in \mathcal{N}_c$ are now defined as the first configuration where at least one searcher can obtain contact with the target. Thus our B/B algorithm utilizes this network reduction technique. The resulting algorithm takes the following form:

Algorithm 4 (Solves MSP).

Input: A time-expanded configuration network $(\mathcal{N}_c, \mathcal{A}_c)$ with nodes $n \in \mathcal{N}$ and arcs $(n, n') \in \mathcal{A}$ where $n = \langle V, t-1\rangle$, $n' = \langle V', t\rangle$. Arc lengths $c_{n,n'}$ in $(\mathcal{N}_c, \mathcal{A}_c)$, the initial target distribution $p(\cdot, 1)$, Markov transition matrix Γ, and upper bound $\hat{q}$.

Output: An optimal configuration path $\mathcal{P}^* = \{V_l\}_{l=0}^{T}$ and its value q^*.

Step 0. Calculate the static bound $d_0(V, t)$ for all $t \in \mathcal{T}$ and $V \in \mathcal{V}_c$, and calculate a lower bound $\hat{q}$. Set $i = 0, \mathcal{K}(i) = \{(V_0, 0, \infty)\}$, where V_0 is the initial configuration. Implement network reduction procedure R5 (see below).

Step 1. If $\mathcal{K}(i)$ is empty, replace i by $i-1$. Else, go to Step 3.

Step 2. If $i = 0$, stop: the last saved configuration-path is optimal and $\hat{q}$ is its probability of detection. Else, go to Step 1.

Step 3. Remove from $\mathcal{K}(i)$ the triplet $(V_t, t, \bar{q}(V_t, t))$ with the largest bound $\bar{q}(V_t, t)$.

Step 4. If $\bar{q}(V_t, t) \leq \hat{q}$, go to Step 1. (Current subpath is fathomed.)

Step 5. If $i = 0$, replace i by $i+1$, and go to Step 3. ($\mathcal{K}(1)$ is populated in the network reduction procedure.)

Step 6. If $t < T$, then for each configuration $V \in \mathcal{F}(V_t)$, calculate $\bar{q}(V, t+1)$ from (III.4) using bounds $d_0(V, t+1)$ and add $(V, t+1, \bar{q}(V, t+1))$ to $\mathcal{K}(i+1)$. Replace i by $i+1$, and go to Step 3. Else, let $\hat{q} = \bar{q}(V_t, t)$ and save the incumbent configuration path $\{V_l\}_{l=0}^{T}$, and go to Step 1.

Network Reduction Procedure R5.

Input: A time-expanded configuration network $(\mathcal{N}_c, \mathcal{A}_c)$, the initial target distribution $p(\cdot, 1)$, Markov transition matrix Γ, and upper bound $\hat{q}$.

Output: The configurations-of-first-contacts $\langle V_s, s\rangle$ which are not dominated by others.

Step 1. Find all configurations-of-first-contact $\langle V_s, s\rangle \in \mathcal{N}_c$. If none exist with $s > 1$, then stop.

Step 2. For each $\langle V_s, s\rangle$, calculate $\bar{q}(V_s, s) = \sum_{c \in \mathcal{C}} Q(c, s) + d_0(V_s, s)$,

Step 3. Eliminate all configurations-of-first-contact $\langle V_s, s\rangle$ with $\bar{q}(V_s, s) \leq \hat{q}$.

Step 4. For each configurations-of-first-contact $\langle V_s, s\rangle$ not eliminated, store the triplet $(V_s, s, \bar{q}(V_s, s))$ in $\mathcal{K}(1)$.

We implement Algorithm 4 and examine its performance on small-scale instances of MSP. The problem instances are as follows: An area of interest (AOI) consists of 5 by 5 cells. The number of searchers is 1, 2, and 3. The searchers, which have a constant glimpse detection probability 0.6, operate in the AOI and their moves are restricted to vertically or horizontally adjacent cells. The target remains in the current cell with probability 0.6 or moves to one of the vertically or horizontally adjacent cells with probability 0.4. The different moves are equally likely. All searchers

depart the upper-left corner cell, and the target starts at the center of the AOI. The time horizon is from 5 to 10. We carry out all experiments on the same computation platform as in Chapter II.

Table 8 reports the run times for different combinations of number of searchers and time horizons. Column 2, 4 and 6 of Table 8 show the run times of Algorithm 4 for the case of 1, 2 and 3 searchers, respectively. For the case with 3 searchers, the run time increases exponentially with increasing time horizon, and the case with time horizon 10 cannot be solved in several days. The case with 3 searchers and the time horizon 9 takes about 5 days to guarantee an optimal solution, however, the optimal solution is found after only 174.78 seconds. The remaining time is spent proving optimality by fathoming other possible configuration paths. Table 9 presents the optimal probability of detection obtained by Algorithm 4. Column 2, 5 and 8 of Table 9 show the optimal probability of detection for the case of 1, 2 and 3 searchers, respectively. Figure 9 illustrate an optimal search plan for the case with 3 searchers and time horizon 9.

	1 searcher		2 searchers		3 searchers	
Time	B/B	SBH	B/B	SBH	B/B	SBH
Horizon	(sec.)	(sec.)	(sec.)	(sec.)	(sec.)	(sec.)
5	0.00	0.00	0.02	0.00	4.17	1.84
6	0.00	0.00	0.03	0.02	5.67	2.17
7	0.00	0.00	0.06	0.02	58.89	2.72
8	0.00	0.00	0.44	0.02	4,341.51	2.91
9	0.00	0.00	5.19	0.03	426,874.92	3.25
10	0.00	0.00	75.77	0.03	N/A	3.70

Table 8. Run times for the branch-and-bound algorithm (B/B) and the static bound heuristic (SBH) on 5 by 5 cell search problem with 1-3 searchers and time horizon $T = 5$-10.

In view of Tables 8 and 9, it is clear that Algorithm 4 is practical only for small-size problems (with few searchers, short time horizon, and/or small area of interest). As noted earlier, the time-expanded configuration network (necessary for implementing the B/B procedure) cannot be generated for large-size problems since

Time Horizon	1 searcher B/B	1 searcher SBH	1 searcher Rel. gap	2 searchers B/B	2 searchers SBH	2 searchers Rel. gap	3 searchers B/B	3 searchers SBH	3 searchers Rel. gap
5	0.306483	0.306483	0.00	0.474213	0.474213	0.00	0.579710	0.579710	0.00
6	0.351647	0.351241	0.00	0.535954	0.521669	0.03	0.643001	0.622074	0.03
7	0.389043	0.380220	0.02	0.581175	0.561550	0.03	0.691865	0.679234	0.02
8	0.416987	0.404325	0.03	0.618416	0.574542	0.07	0.728375	0.711876	0.02
9	0.444506	0.426829	0.04	0.647400	0.620582	0.04	0.754400	0.739376	0.02
10	0.465594	0.438671	0.06	0.673168	0.648007	0.04	N/A	0.762183	N/A

Table 9. Probability of detection obtained by the branch-and-bound algorithm (B/B) and the static bound heuristic (SBH) on 5 by 5 cell search problem with 1-3 searchers and time horizons $T = 5$-10, and relative gap between SBH and B/B solutions.

the extremely large number of data (arcs) can not be stored. Thus, to solve large-size problems, alternative heuristic approaches may be necessary.

2. Static Bound Heuristic

Dell et al. [13] present seven algorithms (one exact procedure and six heuristics) to solve MSP. Among their heuristics, the expected detection heuristic (H1) performs best under a broad range of conditions. As described in the beginning of this section, H1 constructs a search plan (configuration path) by extending a configuration subpath one arc at the time. Each extension is determined by a longest-path calculation on a time-expanded configuration network where arc lengths are given by the expected number of detections.

We present an alternative approach, the static bound heuristic (SBH), which is similar to H1 in [13] but requires only one longest-path calculation and uses improved arc lengths as described below. In Algorithm 4 (i.e., the B/B procedure for MSP), we calculate the static bounds in advance of the main calculations. The bound calculation corresponds to a longest path in the time-expanded configuration network $(\mathcal{N}_c, \mathcal{A}_c)$ with arc length given by (III.8). That longest path, which specifies a search plan, is the solution provided by SBH. We observe that the longest path calculations in H1 amounts to finding a configuration path with the largest expected number of

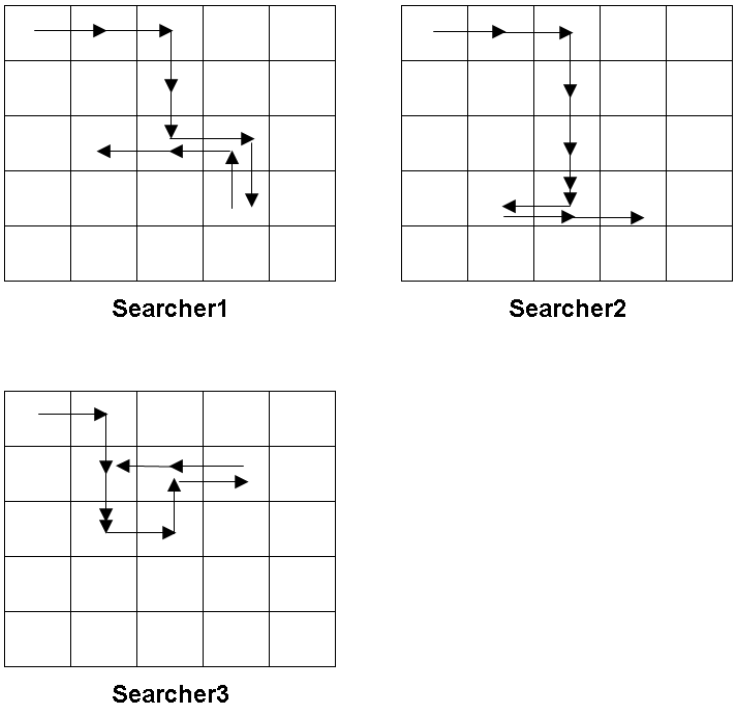

Figure 9. An optimal search plan for the 5 by 5 cell search problem with 3 searchers and the time horizon 9.

detections. In contrast, the longest path calculation in SBH uses arc lengths (III.8) which effectively amounts to finding the configuration path with the largest expected number of detections while at each node along the path accounting for the effect of search at the previous node. As demonstrated in [32] and discussed in Section 1 of Chapter II, accounting for the previous node significantly improves the configuration path found by the longest path calculation.

Column 3, 5 and 7 of Table 8 report the run times for the cases of 1, 2 and 3 searchers, respectively. Since SBH corresponds to a longest-path problem in an

acyclic network, the run times tend to be extremely short. Column 3, 6 and 9 of Table 9 show the obtained probability of detection, and column 4, 7 and 10 describe the relative gap to the corresponding optimal probability of detection. For each case, the obtained detection probability is quite close to the corresponding optimal value. Thus SBH performs well for these problem instances. However the performance tends to be worse as the problem size becomes large. Moreover, this SBH is not available for large-size problem since the time-expanded configuration cannot be generated.

3. Cross-Entropy Method

The cross-entropy (CE) method was developed by Rubinstein [38, 39] for solving rare event simulations and combinatorial optimization problems. The CE method derives its name from the cross-entropy (or Kullback-Leibler) distance between two probability distributions (e.g, an optimal importance sampling distribution and an estimated distribution). The CE method is an iterative procedure consisting of two steps: (1) random samples (i.e., search plans for our case) are generated according to a parameterized probability distribution, and (2) the generated search plans are evaluated using the objective function (i.e., detection probability), and the parameters of the sampling distribution are updated based on the "elite" samples in a manner which increases the possibility that an optimal or near-optimal solution appears in the next iteration.

The CE method is a global search heuristic, and is somewhat similar to genetic algorithms. However the CE method has a simpler scheme for the change of population generation parameter compared to genetic algorithms. Boer et al. [36] compare in details the CE method and other heuristics such as simulated annealing, genetic algorithm, and ant colony method. The CE method has been applied to a large number of combinatorial optimization problems [39, 40, 33, 44, 36, 12] but this dissertation appears to be the first one applying the CE method to search problems.

a. CE Algorithms

In the multiple-search problem (MSP), a possible search plan is describes as a sequence of configurations $\overline{\mathcal{P}} = \{V_l\}_{l=0}^{T}$ where $(V_{l-1}, V_l) \in \mathcal{E}_c$ for all $l = 1, 2, ..., T$. Thus, a crude way to find a search plan is as follows. Let $\mathcal{F}(V) = \{V' \in \mathcal{V}_c |\ (V, V') \in \mathcal{E}_c\}$ be the forward star of configuration V. Start at the given initial configuration $V_0 = (v_0^1, v_0^2, \ldots, v_0^J)$. Select an arbitrary configuration from $\mathcal{F}(V_0)$ and denote it V_1. Next, choose an arbitrary configuration from $\mathcal{F}(V_1)$ and denote it V_2. The same process is repeated until the configuration V_T is obtained.

The CE method takes similar steps, but selects the next configuration according to a probability distribution. For each node $n \in \mathcal{N}_c$ in the time-expanded configuration network $(\mathcal{N}_c, \mathcal{A}_c)$, we define a probability distribution $\sigma_{n,n'}$ over the outgoing arcs $(n, n') \in \mathcal{A}_c$ (i.e., $\sum_{(n,n')\in\mathcal{A}_c} \sigma_{n,n'} = 1$). Then, given a configuration subpath $\{n_l\}_{l=0}^{k}$, $k \in \mathcal{T}$, the configuration subpath is extended by randomly selecting an arc $(n_k, n') \in \mathcal{A}_c$ according to the probability distribution $\sigma_{n_k,n'}$.

Clearly, if $\sigma_{n,n'}$ is degenerate at each node $n \in \mathcal{N}_c$, i.e., there exists an n' such that $(n, n') \in \mathcal{A}_c$ and $\sigma_{n,n'} = 1$, then the probability distribution uniquely specifies a configuration path. The CE methods aims to iteratively update the probability distribution so that it converges to a degenerate distribution specifying the optimal configuration path of MSP. We formalize this approach in the next algorithm.

Algorithm 5 (Basic CE Algorithm).

Parameters. Sample size M, elite size M^{elite}, stopping parameter s (e.g., $s = 5$), and smoothing parameter β.

Step 0. Calculate static bounds $d_0(n)$ for all $n \in \mathcal{N}_c$ and a lower bound $\hat{q}$. Set $\bar{q}^{(0)} = \hat{q}$ and $i = 1$.

Step 1. For all $n \in \mathcal{N}_c$ and $n' \in \mathcal{F}(n)$, define probability distribution $\sigma_{n,n'}^{(i)}$ such that

$$\sigma_{n,n'}^{(i)} = \frac{d_0(n')}{\sum_{n'\in\mathcal{F}(n)} d_0(n')}. \tag{III.9}$$

Step 2. Generate M search plans based on probability distribution $\sigma_{n,n'}^{(i)}$.

Step 3. Choose M^{elite} elite samples among the generated search plans, and calculate the detection probability of the best elite $\bar{q}^{(i)}$. If $\bar{q}^{(i)} > \hat{q}$, then $\hat{q} = \bar{q}^{(i)}$

Step 4. If $\bar{q}^{(i)} = \bar{q}^{(i-1)} = \ldots = \bar{q}^{(i-s)}$, then stop. Else go to Step 5.

Step 5. Update probability distribution $\sigma_{n,n'}^{(i+1)}$ using the M^{elite} elite samples and parameter β as described below. Replace i by $i+1$ and go to Step 2.

In Step 0, the static bounds $d_0(n)$ are calculated for all node $n \in \mathcal{N}_c$ in the time-expanded configuration network. At this time, we obtain a lower bound $\hat{q}$ of the optimal detection probability (i.e., the detection probability along the configuration path corresponding to $d_0(n_0)$). The best detection probability initially (0th iteration), denoted $\bar{q}^{(0)}$, is simply $\hat{q}$. In Step 1, we define the initial probability distribution $\sigma_{n,n'}^{(1)}$ using the static bounds. Next, in Step 2, based on the probability distribution $\sigma_{n,n'}^{(1)}$, we generate M search plans $\overline{\mathcal{P}}_1, \overline{\mathcal{P}}_2, \ldots, \overline{\mathcal{P}}_M$ randomly. After that, in Step 3, the M plans are evaluated using the objective function (i.e., detection probability), and the best performing M^{elite} elite samples are extracted. If the detection probability of the current best elite sample $\bar{q}^{(1)}$ is better than the current lower bound $\hat{q}$ (the best detection probability so far), $\hat{q}$ is updated. Step 4 stops the algorithm if the best values found in the previous several iterations were identical. We hope that the repetition of values is due to a near degenerate probability distributions $\sigma_{n,n'}^{(i)}$ and that the best found search plan is close to the optimal one. However, that is not guaranteed. If the stopping criterion is not satisfied in Step 4, probability distributions are updated using the current M^{elite} elite samples as follows. Let $l_{n,n'}^{(i)}$ be the number of times one of the elite samples use the arc (n, n') in the current i^{th} iteration. The probability distribution is updated using both the current probability distributions and contribution of the elite samples by setting

$$\sigma_{n,n'}^{(i+1)} = \beta \frac{l_{n,n'}^{(i)}}{\sum_{n' \in \mathcal{F}(n)} l_{n,n'}^{(i)}} + (1-\beta)\sigma_{n,n'}^{(i)}. \tag{III.10}$$

where $0.4 \leq \beta \leq 0.9$ as suggested in [36] based on empirical studies.

We note that the parameters (M, M^{elite}, s, β) must be specified to implement Algorithm 5. Boer et al. [36] suggest that the sample size M is chosen according to the size of the problem (e.g., $M = rm$, where m is the number of arcs in the time-expanded configuration network, and r is a constant), and the elite sample size is taken to be approximately $0.01M$. Boer et al. [36] also report that an adaptive choice of the parameter settings speeds up the convergence. Thus, we implement numerical tests for a variety of cases and develop an adaptive CE algorithm. Specifically, we set the elite sample size and smoothing parameter as $M^{elite} = 0.01M$ and $\beta = 0.4$, respectively. Furthermore, we vary the sample size M in the range between M^{min} and M^{max} adaptively. M^{min} is set according to the size of the problem instance, and $M^{max} = 5M^{min}$.

The CE method has asymptotic convergence properties. Specifically, under the assumption that the optimal solution is unique, Costa et al. [3] give necessary and sufficient conditions under which the optimal solution is approached, with probability one, as the number of iterations goes to infinity. For MSP, the CE method provides good solutions (as shown below in numerical tests). However it does not guarantee an optimal solution since MSP often has multiple optimal solutions (e.g., if two searchers are identical) and since the CE method is implemented with a heuristic stopping criterion. The multi-extremal property of the MSP solutions may provide the following undesirable condition. In each iteration, probability distributions are updated using the elite samples including the best elite. Thus, in the next iteration, at least, the best elite of the previous iteration is expected to appear in the samples if the sample size is large. However, the multi-extremal property may bring unstable probability distributions and the current best elite may be worse than that in the previous iteration. In this case, Algorithm 5 is unlikely to stop in reasonable time. For that undesirable situation, we intentionally stop the algorithm and provide the best solution, but label it as "unreliable."

In view of the above discussion, we develop a specialized adaptive CE algorithm to solve the MSP. In this algorithm, we refer to the sample size at i^{th} iteration as M_i.

Algorithm 6 (Adaptive CE Algorithm).

Parameters. Minimum and maximum sample sizes M^{min} and M^{max} and smoothing parameter β.

Step 0. Calculate static bounds $d_0(n)$ for all $n \in \mathcal{N}_c$ and lower bound $\hat{q}$. Set $\bar{q}^{(0)} = \hat{q}$ and $i = 1$.

Step 1. For all $n \in \mathcal{N}_c$ and $n' \in \mathcal{F}(n)$, define probability distribution $\sigma^{(i)}_{n,n'}$ such that

$$\sigma^{(i)}_{n,n'} = \frac{d_0(n')}{\sum_{n' \in \mathcal{F}(n)} d_0(n')}. \tag{III.11}$$

Set $M_i = M^{min}$.

Step 2. Generate M_i search plans based on probability distribution $\sigma^{(i)}_{n,n'}$.

Step 3. Choose $M^{elite} = 0.01 M_i$ elite samples among the generated search plans, and calculate the detection probability of the best elite $\bar{q}^{(i)}$. If $\bar{q}^{(i)} > \hat{q}$, then $\hat{q} = \bar{q}^{(i)}$

Step 4. If $\bar{q}^{(i)} > \bar{q}^{(i-1)}$, update $\sigma^{(i+1)}_{n,n'}$ using the M^{elite} elite samples. Replace i by $i + 1$. Set $M_i = M^{min}$ and go to Step 2. Else go to Step 5.

Step 5. If the best detection probabilities are identical for three iterations (i.e., $\bar{q}^{(i)} = \bar{q}^{(i-1)} = \bar{q}^{(i-2)}$), then stop. (The obtained solution is considered reliable). Else go to Step 6.

Step 6. If $M_i < M^{max} = 5M^{min}$, increase the sample size $M_i = M_i + M^{min}$ and go to Step 2. Else go to Step 7.

Step 7. If $M_k = M_{i-1} = \ldots = M_{i-5}$, then stop. (The obtained solution is considered unreliable). Else update $\sigma^{(i+1)}_{n,n'}$ using the latest M^{elite} elite samples. Replace i by $i + 1$. $M_i = M^{min}$ and go to Step 2

Algorithm 6 is based on the time-expanded configuration network, which is problematic for large-scale problems. However, we overcome this difficulty by considering a time-expanded network for each searcher. For each searcher $j \in \mathcal{J}$, we construct a time-expanded network $\mathcal{G}^j$ as in Chapter II. For each $\mathcal{G}^j$, we define the

probability distributions $\sigma^j_{n,n'}$. We then generate a search plan by generating a path $\mathcal{P}^j = \{v^j_l\}^T_{l=0}$ for each searcher j according to the probability distributions $\sigma^j_{n,n'}$. The search plan is simply $\overline{\mathcal{P}} = (\mathcal{P}^1, \mathcal{P}^2, ..., \mathcal{P}^J)$. We update the probability distributions $\sigma^j_{n,n'}$, $j \in \mathcal{J}$, by considering the joint search effect of all searchers. Specifically, we determine as before the M^{elite} elite search plan and define $l^{j(i)}_{n,n'}$ be the number of times one of the elite samples use the arc (n, n') for search j in the current i^{th} iteration. The probability distribution for each searcher j is then updated by setting

$$\sigma^{j(i+1)}_{n,n'} = \beta \frac{l^{j(i)}_{n,n'}}{\sum_{n' \in \mathcal{F}(n)} l^{j(i)}_{n,n'}} + (1-\beta)\sigma^{j(i)}_{n,n'}. \tag{III.12}$$

An advantage of this approach is that we can treat large-scale problems since we avoid generating the time-expanded configuration network. We summarize this "decomposition" approach next.

Algorithm 7 (Adaptive CE Algorithm with Decomposition).

Parameters. Minimum and maximum sample sizes M^{min} and M^{max} and smoothing parameter β.

Step 0. Generate the time-expanded network $\mathcal{G}^j$, $j \in \mathcal{J}$. In each $\mathcal{G}^j$, calculate static bounds $d^j_0(n)$ for all nodes $n \in \mathcal{G}^j$. Generate a search plan $\overline{\mathcal{P}}$ by collecting each searcher's path corresponding to the static bound $d^j_0(n_0)$. Calculate detection probability $q(\overline{\mathcal{P}})$ from (III.3). Set $\hat{q} = \bar{q}^{(0)} = q(\overline{\mathcal{P}})$ and $i = 1$.

Step 1. For $j \in \mathcal{J}$, define the probability distributions $\sigma^{j(i)}_{n,n'}$ by

$$\sigma^{j(i)}_{n,n'} = \frac{d^j_0(n')}{\sum_{n' \in \mathcal{F}(n)} d^j_0(n')}. \tag{III.13}$$

Set $M_i = M^{min}$.

Step 2. Construct M_i search plans by generating each searcher's path based on probability distribution $\sigma^{j(i)}_{n,n'}$. For each search plan, $\overline{\mathcal{P}}$ calculate the detection probability $q(\overline{\mathcal{P}})$ from (III.3).

Step 3. Choose $M^{elite} = 0.01 M_i$ elite search plans among the generated search plans, and set the detection probability of the best elite as $\bar{q}^{(i)}$. If $\bar{q}^{(i)} > \hat{q}$, then $\hat{q} = \bar{q}^{(i)}$

Step 4. If $\bar{q}^{(i)} > \bar{q}^{(i-1)}$, update $\sigma_{n,n'}^{j(i+1)}$, $\forall j \in \mathcal{J}$ using searcher j's path contributing to M^{elite} elite samples. Replace i by $i+1$. $M_i = M^{min}$ and go to Step 2. Else go to Step 5.

Step 5. If the best detection probabilities are same for three iterations (i.e., $\bar{q}^{(i)} = \bar{q}^{(i-1)} = \bar{q}^{(i-2)}$), then stop. (The obtained solution is considered reliable). Else go to Step 6.

Step 6. If $M_i < M^{max} = 5M^{min}$, increase the sample size $M_i = M_i + M^{min}$ and go to Step 2. Else go to Step 7.

Step 7. If $M_k = M_{i-1} = \ldots = M_{i-5}$, then stop. (The obtained solution is considered unreliable). Else update $\sigma_{n,n'}^{j(i+1)}$, $j \in \mathcal{J}$, using (III.12). Replace i by $i+1$. Set $M_i = M^{min}$ and go to Step 2

In Algorithm 7, for each $j \in \mathcal{J}$, the probability distributions $\sigma_{n,n'}^{j(i)}$ are rather poor in the initial iterations compared to the situation in Algorithm 6. However, the distributions are gradually improved as the cooperative search effort is accounted for when determining the elite search plans.

The decomposition approach is practical for large-size problems since it only requires the time-expanded network for each single searcher. Thus, the approach is easily be implemented using the node-and-time expanded network with the potential for more effective generation of paths (see Chapter II for a discussion about the advantage of the note-and-time expanded network over the time-expanded network). The size of the node-and-time expanded network is larger than time-expanded network, but it is substantially smaller than the time-expanded configuration network. Since the algorithm using the node-and-time expanded network is essentially identical to Algorithm 7 (except for using the node-and-time expanded network), we omit to explicitly describe that algorithm but refer to it as Algorithm 7N.

b. Numerical Tests

We implement Algorithm 6, Algorithm 7, and Algorithm 7N using the twelve problem instances listed in Table 10. The problem instances are similar to the ones in Tables 8 and 9, and they have the same assumptions and parameter settings

(e.g., start positions, glimpse detection probability and target stationary probability, etc). The minimum sample size M^{min} is decided according to the size of problem instance, see columns 5 and 6 of Table 10 where m is the number of arcs in the network corresponding to the algorithms. As before, we set $M^{max} = 5M^{min}$ and $\beta = 0.4$. Since Algorithm 6 uses the time-expanded configuration network, it is available only for small problem instances, i.e., cases A2, A3, B2, and B3. The CE heuristics (Algorithms 6, 7, and 7N) are compared with Algorithm 4 (i.e., the B/B algorithm). When available, we compare the optimal value from Algorithm 4 with the search plan obtained by the CE heuristics. We also compare with incumbent solution available in Algorithm 4 at the point in time when the CE heuristics terminate.

Case	Area size	Time horizon	Number of searchers	Min. sample size Algo. 6	Min. sample size Algo. 7/7N
A2	5 by 5	9	2	$0.1m$	$10000m$
A3	5 by 5	9	3	$0.01m$	$10000m$
B2	5 by 5	18	2	$0.1m$	$10000m$
B3	5 by 5	18	3	$0.01m$	$10000m$
C2	15 by 15	18	2	N/A	$1000m$
C3	15 by 15	18	3	N/A	$1000m$
D1	15 by 15	27	1	N/A	$100m$
D2	15 by 15	27	2	N/A	$100m$
D3	15 by 15	27	3	N/A	$100m$
E1	15 by 15	28	1	N/A	$100m$
F1	15 by 15	29	1	N/A	$100m$
G1	15 by 15	30	1	N/A	$100m$

Table 10. Test cases with Algorithm 6, Algorithm 7 and Algorithm 7N with area size, time horizon, number of searchers, and minimum sample size. m is the number of arcs in the network used in the corresponding algorithm.

(a) Single searcher

Tables 11 and 12 report the numerical results for problem instances with a single searcher. Columns 2 and 3 in Table 11 present the detection probabilities obtained by the CE heuristics and column 4 shows the corresponding optimal solution obtained by Algorithm 4. We find that Algorithms 7 and 7N have comparable solution quality. The last column in Table 11 reports the probability of detection of the incumbent search plan of Algorithm 4 at the time Algorithm 7 terminates. For example, for case D1, the best detection probability available to Algorithm 4 at time 50.98 seconds (see Table 12) is 0.304046. Table 12 reports the corresponding run times (columns 2 and 3 for Algorithms 7 and 7N, respectively, and column 5 for Algorithm 4). In Table 12, we have also included the time at which Algorithm 4 finds the optimal solution (but not yet proven optimal), see column 4. The run time of Algorithm 4 is clearly slower than that of the CE heuristics. We also observe that the run time of Algorithm 7 is much faster than that of Algorithm 7N. In a comparison of columns 2, 3, and 5 in Table 11, we find that the CE heuristics terminates with a better search plan than what is available from Algorithm 4 at the same time. The solution quality of the CE heuristics is quite good for all the instances examined.

	CE heuristic		Branch-and-bound (Algo. 4)	
Case	Algo. 7	Algo. 7N	Optimal	Early term.
D1	0.305254	0.305254	0.305254	0.304046
E1	0.311225	0.313101	0.313101	0.307411
F1	0.320277	0.320716	0.320719	0.313043
G1	0.325968	0.325968	0.327823	0.320131

Table 11. Probability of detection for problem instances with a single searcher.

(b) Two searchers

Tables 13 and 14 describe the computational results for problem instances with two searchers. The columns show the same content as the corresponding columns in Tables 11 and 12. Algorithm 4 performs well for the smallest problem instance (A2), but is not available for large instances. Algorithms 7 and 7N obtain

Case	CE heuristic Algo. 7 (sec.)	Algo. 7N (sec.)	Branch-and-bound (Algo. 4) Found optimal (sec.)	Proved optimal (sec.)
D1	50.98	226.08	151.14	752.32
E1	47.28	229.17	536.40	2732.36
F1	79.26	302.88	1955.64	9690.79
G1	60.17	309.08	6255.47	34820.26

Table 12. Run times on problem instances with a single searcher.

solutions in practical times and their solution quality is better than that of Algorithm 6. In addition, they can obtain solutions for large problems. Hence, it appears that Algorithms 7 and 7N are superior to Algorithm 6 for problem instances with two searchers. Algorithm 7 and Algorithm 7N are comparable, but the former takes less times to obtain a solution. The run time of the CE heuristics is smaller for case D2 than for C2, see Table 14. This is counterintuitive as cases D2 have a longer time horizon than C2. However, the randomness in the algorithms may cause such effects. Overall, it appears that Algorithm 7 is the algorithm of choice as in the case of two searchers.

Case	CE heuristic Algo. 6	Algo. 7	Algo. 7N	Branch-and-bound (Algo. 4) Optimal	Early term.
A2	0.646586	0.647400	0.647400	0.647400	0.647400
B2	0.799565	0.801566	0.801552	N/A	N/A
C2	N/A	0.336465	0.336483	N/A	N/A
D2	N/A	0.474782	0.476186	N/A	N/A

Table 13. Probability of detection for problem instances with two searchers.

(c) Three searchers

Tables 15 and 16 report the numerical results for the cases with three searchers. The columns are the same as the corresponding columns in Tables 11 and 12. We observe that Algorithm 7 dominates Algorithm 6 in solution quality and run times. Furthermore, it is available for the large-scale problem instances. As in the

	CE heuristic			Branch-and-bound (Algo. 4)	
Case	Algo. 6 (sec.)	Algo. 7 (sec.)	Algo. 7N (sec.)	Found optimal (sec.)	Proved optimal
A2	6.78	11.09	48.45	0.67	5.19
B2	35.23	169.89	517.05	N/A	N/A
C2	N/A	650.93	1666.43	N/A	N/A
D2	N/A	210.43	951.36	N/A	N/A

Table 14. Run times on problem instances with two searchers.

case of two searchers, Algorithms 7 and 7N are comparable, but the former tends to require less computing time to obtain a reasonable solution.

For the smallest size problem instance (A3), Algorithm 7 obtains an optimal solution (however it is not guaranteed) in 49 seconds. On the other hand, Algorithm 4 obtains an optimal solution in 174.78 seconds and proves it optimal in about 5 days. The best detection probability after 49 seconds in Algorithm 4 is 0.749631. Thus the CE heuristic Algorithm 7 appears to generate good solutions quicker than the B/B based Algorithm 4.

Comparing the CE heuristic (Tables 11-16) with the Static Bound Heuristic (SBH) (Tables 8 and 9 second to last row), we find that the SBH dominates the CE heuristic in run time (3.25 seconds compared to 49 seconds for case A3). However, the SBH is available only for small-size problem. In addition, the solution quality of the CE heuristic is better than that of the SBH based on the limited tests .

In view of the above results, Algorithm 7 appears to be an overall efficient heuristic for solving MSP.

	CE heuristic			Branch-and-bound (Algo. 4)	
Case	Algo. 6	Algo. 7	Algo. 7N	Optimal	Early term.
A3	0.753688	0.754400	0.754400	0.754400	0.749631
B3	0.889145	0.891720	0.893104	N/A	N/A
C3	N/A	0.436528	0.436528	N/A	N/A
D3	N/A	0.587159	0.593178	N/A	N/A

Table 15. Probability of detection for problem instances with three searchers.

	CE heuristic			Branch-and-bound (Algo. 4)	
	Algo. 6	Algo. 7	Algo. 7N	Found optimal	Proved optimal
Case	(sec.)	(sec.)	(sec.)	(sec.)	
A3	144.75	49.00	249.42	174.78	426,874.92
B3	767.39	271.43	1567.62	N/A	N/A
C3	N/A	674.85	3815.69	N/A	N/A
D3	N/A	297.30	1454.73	N/A	N/A

Table 16. Run times on problem instances with three searchers.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. MULTIPLE HOMOGENEOUS SEARCHER PROBLEM

This chapter focuses on the multiple-searcher problems where all searchers are identical, i.e., the multiple homogenous searcher problem (MHSP). The goal for the searchers is to minimize the probability of not detecting the target within a mission time, which is equivalent to maximize the probability of detecting the target within the same time. We utilize the convexity of the nonlinear objective function (the non-detection probability), and introduce an exact algorithm using cutting planes (outer approximations). Under certain assumptions, the problem becomes equivalent to a large-scale linear mixed-integer program. We also present several new cuts for MHSP and demonstrate their effect in computational tests.

A. PROBLEM DESCRIPTION AND FORMULATION

Chapter III presented three algorithms (the branch-and-bound procedure and two heuristics) to solve the multiple searcher problem (MSP), which are also applicable for solving MHSP. Here we introduce an alternative approach especially tailored to solve MHSP. This section formulates MHSP by following [7, 18]. We use the same notations and assumptions as in MSP. Specifically, we assume that: (1) there is no end-point restriction for any searcher; (2) the airspace has only one altitude. i.e., there is only one vertex corresponding to each cell in the discretized area of interest; and (3) the search effect at the current waypoint is independent of the previous waypoint. In addition, we assume that the "search effect" is described by an exponential detection function instead of an arbitrary function as in earlier chapters. That is, when the number of searchers in cell c at time period t is $y_{c,t}$, the probability of detecting the target in that cell during that time period given the target occupies cell c at time period t is described as $1-\exp(-\alpha_{c,t}y_{c,t})$ instead of $1-(1-g(c,t))^{y_{c,t}}$ as used in Chapter III. Here, the term $\alpha_{c,t}$ is the detection rate for a single searcher in cell c

and time period t defined by $\alpha_{c,t} = -\log(1 - g(c,t))$. We assume that $g(c,t) < 1$ (i.e., the sensor is not perfect) so that $\alpha_{c,t}$ is finite. Additionally, we let $\omega(t)$ be the target's cell at time period $t \in \mathcal{T}$, the sequence of cells $\omega = (\omega(1), \omega(2), \ldots, \omega(T))$ denote a target path, and p_ω be the probability that the target takes that path ω. The set of all possible target paths is denoted as Ω. In this notation, MHSP is formulated as follows.

Formulation of MHSP

Indices

c, c'	cells ($c, c' \in \mathcal{C} = \{1, \ldots, C\}$)
t	time step ($t \in \mathcal{T} = \{0, 1, ..., T\}$)
ω	target path ($\omega \in \Omega$)

Parameters

$\alpha_{c,t}$	detection rate for a single searcher in cell c and time period t
$\alpha^{\omega}_{c,t}$	$\alpha_{c,t}$ if cell c is on target path ω at time period t, zero otherwise.
$y_{c,0}$	number of searchers in cell c at time period 0
p_ω	probability that the target takes path ω

Variables

$x_{c,c't}$	number of searchers that is redistributed from cell c in time period t to cell c' in time period $t+1$
$y_{c,t}$	number of searchers in cell c in time period t

Formulation

$$\min \ f(y) = \sum_{\omega \in \Omega} p_\omega \exp\left(-\sum_{c,t} \alpha^{\omega}_{c,t} y_{c,t}\right)$$

s.t.

$$\sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T$$

$$y_{c,t} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T}$$

$x_{c,c',t}$, $y_{c,t}$: integer

We refer to this problem which minimizes the probability that the target is not detected during time horizon T subject to network flow-balance constraints as the multiple homogeneous searcher problem (MHSP). The objective function of this nonlinear mixed-integer program is clearly convex. For reference later, elements of

the gradient $\nabla f(y)$ of the objective function $f(y)$ are

$$\frac{\partial f(y)}{\partial y_{c,t}} = -\sum_{\omega \in \Omega} p_\omega \alpha^\omega_{c,t} \exp(-\sum_{c,t} \alpha^\omega_{c,t} y_{c,t}). \tag{IV.1}$$

Since the formulation uses all possible target paths $\omega \in \Omega$, if the number of possible target paths is extremely large, calculating the objective function value $f(y)$ or its gradient $\nabla f(y)$ for any solution y becomes extremely computationally expensive. Brown [7] introduces an alternative formulation for the case of Markovian target movement using conditioning on the target's position at a time period as follows.

Given a solution y, let $r_{c,t}(y)$ be the probability of the target visiting cell c at time period t without being detected in time periods 1, 2, ..., $t-1$, and $s_{c,t}(y)$ be the probability that the target departs cell c in time period t and is not detected by any searches in time periods $t+1$, $t+2$, ..., T. Let $r_{\cdot,t}(y) = [r_{1,t}(y), r_{2,t}(y), \ldots, r_{C,t}(y)]$ and $s_{\cdot,t}(y) = [s_{1,t}(y), s_{2,t}(y), \ldots, s_{C,t}(y)]$. We define $r_{\cdot,1}(y) = p(\cdot, 1)$, where $p(\cdot, 1)$ is a given initial target distribution, and $s_{c,T}(y) = 1$ for any cell $c \in \mathcal{C}$. Let Γ be a target transition matrix. Thus $r_{\cdot,t}(y)$ and $s_{\cdot,t}(y)$ are calculated recursively by

$$r_{\cdot,t}(y) = [r_{1,t-1}(y) \exp(-\alpha_{1,t-1} y_{1,t-1}), \ldots, r_{C,t-1}(y) \exp(-\alpha_{C,t-1} y_{C,t-1})]\Gamma, \tag{IV.2}$$

$$s_{\cdot,t}(y) = [s_{1,t+1}(y) \exp(-\alpha_{1,t+1} y_{1,t+1}), \ldots, r_{C,t+1}(y) \exp(-\alpha_{C,t+1} y_{C,t+1})]\Gamma', \tag{IV.3}$$

where Γ' is the transpose matrix of Γ. Since we only consider a target moving according to a Markov transition matrix, we can apply this result. In this notation, for any $t = 1, 2, ..., T$, the objective function is alternatively expressed as

$$f(y) = \sum_{c \in \mathcal{C}} r_{c,t}(y) \exp(-\alpha_{c,t} y_{c,t}) s_{c,t}(y), \tag{IV.4}$$

and elements of the gradient $\nabla f(y)$ are

$$\frac{\partial f(y)}{\partial y_{c,t}} = -\alpha_{c,t} r_{c,t}(y) \exp(-\alpha_{c,t} y_{c,t}) s_{c,t}(y). \tag{IV.5}$$

Thus, for any solution y, the objective function value and the gradient can be evaluated with a moderate computational effort.

B. ALGORITHMS FOR MHSP

As mentioned, MHSP is a convex nonlinear mixed-integer program. This section introduces an exact outer approximation (OA) algorithm to solve MHSP using the convexity of the objective function [30]. The OA algorithm refers to the fact that the surface described by a convex function lies above the supporting hyperplane to the convex function at any point. A supporting hyperplane at a point $y^{(i)}$ takes the form $f(y^{(i)}) + \nabla f(y^{(i)})'(y - y^{(i)})$ (see Figure 10). The algorithm iteratively generates supporting hyperplanes (cuts) and accumulates them to provide successively improving linear approximations of the nonlinear convex function (see Figure 10). The linear approximation underestimates the objective function. We start by describing a basic OA algorithm. After that we introduce several new cuts and present computational results in the subsequent subsections.

1. Basic OA Algorithm

The basic OA algorithm is described as Algorithm 8 (see below), which solves the following master problem as part of its calculations. Specifically, we denote that the master problem of the k-th iteration of Algorithm 8 for MP1(k) and its optimal value and optimal solution for $z^{(k)}$ and $y^{(k)}$, respectively.

Formulation of Master problem : MP1(k)

$$
\begin{aligned}
&\min \ z \\
&\ s.t. \\
&\ z \geq f(y^{(i)}) + \nabla f(y^{(i)})'(y - y^{(i)}) \quad i = 0, 1, ..., k-1 \\
&\ \sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T \\
&\ y_{c,t} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T} \\
&\ x_{c,c',t},\ y_{c,t}\text{: integer}
\end{aligned}
$$

Algorithm 8 (Basic OA Algorithm).

Step 0. Set a relative optimality tolerance $\delta \geq 0$. Choose an initial feasible solution $y^{(0)}$.

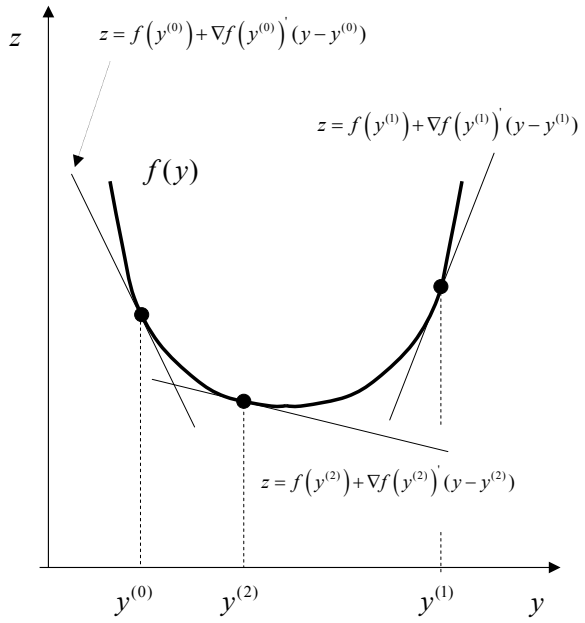

Figure 10. Linear supporting functions.

Step 1. Calculate $f(y^{(0)})$ and $\nabla f(y^{(0)})$ (see (IV.4) and (IV.5)). Set lower bound $\underline{q} = 0$, upper bound $\bar{q} = f(y^{(0)})$, and $k = 1$.

Step 2. If the gap $(\bar{q} - \underline{q})/\underline{q} \leq \delta$, stop. Else, solve the master problem MP1(k), and obtain its optimal value $z^{(k)}$ and optimal solution $y^{(k)}$. If $z^{(k)} > \underline{q}$, then $\underline{q} = z^{(k)}$.

Step 3. Calculate $f(y^{(k)})$ and $\nabla f(y^{(k)})$ (see (IV.4) and (IV.5)).

Step 4. If $f(y^{(k)}) < \bar{q}$, then $\bar{q} = f(y^{(k)})$. Replace k by $k + 1$, and go to Step 2.

As seen, Algorithm 8 generates one cut in each iteration (a strategy we refer to as "single-cut") at a solution determined by the master problem MP1(k).

It is well known that using a nonzero optimality tolerance when solving, at least the initial master problems in Algorithm 8 may reduce overall computing time

as cuts can be generated more cheaply. We examine this possibility. Let $\epsilon^{(k)} \geq 0$ be a relative optimal tolerance for the solution of the master problem MP1(k). Then, the value $z^{(k)}_{\epsilon^{(k)}}$ obtained from the master problem may not be a valid lower bound on the optimal value of MHSP. However, $(1-\epsilon^{(k)})z^{(k)}_{\epsilon^{(k)}}$ is a valid lower bound and we use that instead of $z^{(k)}$ in step 2 of Algorithm 8. We note that when $\epsilon^{(k)}$ is constant at zero for iterations after some finite iteration, Algorithm 8 is an exact algorithm and provides an optimal solution after a finite number of iterations [16].

We examine Algorithm 8 using the problem instance "case C3" in Table 10 of Chapter III which involves 15 by 15 cells, a time horizon of 18, and 3 searchers. We also adopt the same assumptions and parameter settings as in Chapter III. Our program is coded using the General Algebraic Modeling System (GAMS) Distribution 22.5 [21] and is implemented on the same computational platform as in Chapters II and III. The master problem MP1(k) is solved by the CPLEX 10.0 solver [27]. In the master problem, we assume that the optimality tolerance $\epsilon^{(k)}$ is constant with value 0 (version 1) or with value 0.03 (version 2).

All searchers depart the upper-left corner cell. Thus, as a choice of initial feasible solution (search plan), we consider the situation that all searchers loiter at the depot during the whole time horizon. We refer to this search plan as a "trivial search" plan. This trivial plan is clearly unwise, but we apply this plan as an initial feasible solution in the preliminary tests. Later we introduce a procedure to obtain a better initial feasible solution.

In the initial numerical tests, the goal is to compare various approaches thus we terminate the calculations after one hour. Table 17 reports lower/upper bounds on the optimal value and the relative gap between the bounds at every ten minute during one hour calculations using Algorithm 8. As reference, the best known non-detection probability is 0.563472 (=1-0.436528) which is found in Table 15 (case C3) in Chapter III. The one hour run time includes the solution time for the master problems (Step 2) and the overhead time for generating cuts, etc. (Steps 1 and 3).

Algorithm 8 versions 1 and 2 use 96.4% and 85.1% of the run time to solve the master problems, respectively. During one hour, version 1 executes only 25 iterations because the tighter optimality tolerance in the master problems, while version 2 executes 96 iterations. Even though the cuts in version 2 may not be as "deep" as the one generated in version 1, it is clear from Table 17 that the extra effort to solve the master problem optimally may not be worth it.

In these test, the relative optimal tolerance $\epsilon^{(k)}$ of the master problem is constant (0 or 0.03) for the one hour calculations. We note that if $\epsilon^{(k)}$ is not zero, Algorithm 8 may regeneration a cut. This can be prevented in many ways, especially if we access the master problem solver. However, in these tests, as well as in tests below, we adopt the simple approach of adjusting the relative optimality tolerance $\epsilon^{(k)}$. For example, $\epsilon^{(k)}$=0.03 appears to be a practical number if the goal is a 5% near-optimal solution.

Time	Algo. 8 version 1: $\epsilon^{(k)} = 0$			Algo. 8 version 2: $\epsilon^{(k)} = 0.03$		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.410705	0.620493	0.511	0.467784	0.576832	0.233
20	0.422249	0.599865	0.421	0.471399	0.576832	0.224
30	0.428358	0.599865	0.400	0.473591	0.576832	0.218
40	0.437659	0.585815	0.339	0.475205	0.574215	0.208
50	0.448581	0.585815	0.306	0.475316	0.574215	0.208
60	0.449400	0.585815	0.304	0.475316	0.572823	0.205

Table 17. Numerical results for Algorithm 8 versions 1 and 2 for every ten minutes of calculations. The best known non-detection probability is 0.563472 (=1-0.436528) found in Table 15 (case C3) in Chapter III.

2. New Cuts for OA Algorithm

This subsection introduces several new cuts and demonstrates their effect. Initially two new cuts (multi-cut and strong-cut) are presented. After that we prove that, under certain assumption, MHSP is equivalent to a large-scale linear mixed-integer program, which motivates an approach for obtaining a good initial solution.

Next, two additional cuts (relative-cut and symmetric-cut) are presented. Finally, we develop a specific OA algorithm by combining these cuts effectively.

(a) Multi-cut

In Table 17, we learned that at least a moderate number of cuts is necessary to bring up the lower bound. The aim of the multi-cut presented here is that, in each iteration, we generate multiple cuts instead of single cut to accumulate many cuts fast and push up the lower bound quickly. The multi-cut utilizes a similar technique to the multicut version of the L-shaped method in stochastic programming with two-stage recourse (see Chapter 5 in [6]). The basic idea is that, for the objective function $f(y) = \sum_{\omega\in\Omega} p_\omega f_\omega(y)$ where $f_\omega(y) = \exp(-\sum_{c,t} \alpha^\omega_{c,t} y_{c,t})$, we consider outer approximation of $f_\omega(y)$ for all $\omega \in \Omega$ instead of for $f(y)$, and generate $|\Omega|$ cuts at each iteration. However, the number of possible target paths $|\Omega|$ is extremely large. Thus, according to the target movement during the initial few time steps, we define U (typically $U << |\Omega|$) "scenarios" $\tilde{\Omega}_1$, $\tilde{\Omega}_2$, ..., $\tilde{\Omega}_U$, where a scenario $\tilde{\Omega}_u$ is a subset of target paths (i.e., $\tilde{\Omega}_u \subset \Omega$, $u = 1, 2, ..., U$). Moreover, each scenario is mutually exclusive (i.e., $\tilde{\Omega}_u \cap \tilde{\Omega}_{u'} = \emptyset$, $u \neq u'$) and each possible target path ω belongs to some scenario (i.e., $\cup_{u=1}^{U} \tilde{\Omega}_u = \Omega$). For example, from a known target location at time period $t = 1$ there are five possible target movements (stay in the initial position or go up/down/left/right) for the next time period. Thus we can define five ($U = 5$) scenarios based on the target movement conditioned during the initial two time steps. Similarly, for the initial three time steps (t=1,2 and 3), twenty five scenario ($U = 25$) are defined if boundary effects are ignored. Let $\tilde{p}_u$ be the probability that the target takes any path ω such that $\omega \in \tilde{\Omega}_u$.

In order to apply this multi-cut approach, the objective function (IV.4) and the gradient (IV.5) need to be expressed differently. When the solution is y, let $r^u_{c,t}(y)$ be the probability of target visiting cell c at time period t without being detected in time periods 1, 2, ..., $t-1$ given the target takes a path ω corresponding to scenario u. Similarly, $s^u_{c,t}(y)$ is defined as the probability that the target departs cell

c in time period t and is not detected by any searches in time periods $t+1$, $t+2$, ..., T given the target chooses a path ω corresponding to scenario u. Let $r^u_{\cdot,t}(y) = [r^u_{1,t}(y), r^u_{2,t}(y), \ldots, r^u_{C,t}(y)]$ and $s^u_{\cdot,t}(y) = [s^u_{1,t}(y), s^u_{2,t}(y), \ldots, s^u_{C,t}(y)]$. By considering the conditioning on the target's position with respect to scenario u, both $r^u_{\cdot,t}(y)$ and $s^u_{\cdot,t}(y)$ are recursively calculated similar as $r_{\cdot,t}(y)$ and $s_{\cdot,t}(y)$. In this notation, the objective function is redefined as

$$f(y) = \sum_{u=1}^{U} \tilde{p}_u f_u(y), \tag{IV.6}$$

where, for any $t = 1, 2, ..., T$, $f_u(y) = \sum_{c\in\mathcal{C}} r^u_{c,t}(y) \exp(-\alpha_{c,t} y_{c,t}) s^u_{c,t}(y)$, and elements of the gradient $\nabla f_u(y)$ are

$$\frac{\partial f_u(y)}{\partial y_{c,t}} = -\alpha_{c,t} r^u_{c,t}(y) \exp(-\alpha_{c,t} y_{c,t}) s^u_{c,t}(y). \tag{IV.7}$$

The OA algorithm with U cuts per iteration is described as Algorithm 9. The algorithm solves the following master problem MP2(k) in the k-th iteration. We denote the corresponding optimal value and optimal solution for $z^{(k)}$ and $y^{(k)}$, respectively.

Formulation of Master problem : MP2(k)

$$\begin{aligned}
&\min\ z = \textstyle\sum_{u=1}^{U} \tilde{p}_u z_u \\
&\ s.t. \\
&\ z_u \geq f_u(y^{(i)}) + \nabla f_u(y^{(i)})'(y - y^{(i)}) \quad u = 1, 2, ..., U,\ i = 0, 1, ..., k-1 \\
&\ \textstyle\sum_{c'\in\mathcal{R}(c)} x_{c',c,t-1} = \sum_{c'\in\mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T \\
&\ y_{c,t} = \textstyle\sum_{c'\in\mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T} \\
&\ x_{c,c',t},\ y_{c,t}\text{: integer}
\end{aligned}$$

Algorithm 9 (OA Algorithm with multi-cut).

Step 0. Set a relative optimality tolerance $\delta \geq 0$. Choose an initial feasible solution $y^{(0)}$.

Step 1. Calculate $f_u(y^{(0)})$ and $\nabla f_u(y^{(0)})$, $u = 1, 2, ..., U$ (see (IV.6) and (IV.7)). Set lower bound $\underline{q} = 0$, upper bound $\bar{q} = \sum_{u=1}^{U} \tilde{p}_u f_u(y^{(0)})$, and $k = 1$.

Step 2. If the gap $(\bar{q}-\underline{q})/\underline{q} \leq \delta$, stop. Else, solve the master problem MP2(k), and obtain its optimal value $z^{(k)}$ and optimal solution $y^{(k)}$. If $z^{(k)} > \underline{q}$, then $\underline{q} = z^{(k)}$.

Step 3. Calculate $f_u(y^{(k)})$ and $\nabla f_u(y^{(k)})$, $u = 1, 2, ..., U$ (see (IV.6) and (IV.7)).

Step 4. If $f(y^{(k)}) = \sum_{u=1}^{U} \tilde{p}_u f_u(y^{(k)}) < \bar{q}$, then $\bar{q} = f(y^{(k)})$. Replace k by $k+1$, and go to Step 2.

Similar to the master problem MP1(k) in Algorithm 8, the relative optimality tolerance of the master problem MP2(k) is denoted by $\epsilon^{(k)} \geq 0$. Again we note that, if $\epsilon^{(k)} > 0$, $z^{(k)}$ must be replaced by $(1-\epsilon^{(k)})z^{(k)}_{\epsilon^{(k)}}$ to obtain a valid lower bound in Step 2 of Algorithm 9. Here, $z^{(k)}_{\epsilon^{(k)}}$ is the value turn by the solver when using relative optimality tolerance $\epsilon^{(k)} > 0$ when solving MP2(k).

We examine Algorithm 9 using the same problem instance as in Table 17. In these instances, conditioning on the target movement in the initial two time steps (t=1 and 2) results in U=5 scenarios. Conditioning on the initial three time steps results in $U = 25$ scenarios. We refer to these two versions of Algorithm 9 as "multi-t2-cut" and "multi-t3-cut," respectively. As above, the relative optimality tolerance $\epsilon^{(k)}$ when solving the master problem MP2(k) is set to 0 or 0.03. This results in a total of four versions of Algorithm 9:

- version 1: multi-t2-cut, and MP2(k) solved with $\epsilon^{(k)} = 0$
- version 2: multi-t2-cut, and MP2(k) solved with $\epsilon^{(k)} = 0.03$
- version 3: multi-t3-cut, and MP2(k) solved with $\epsilon^{(k)} = 0$
- version 4: multi-t3-cut, and MP2(k) solved with $\epsilon^{(k)} = 0.03$

We note that the initial feasible solution is still set to be the trivial search plan (i.e., all searchers keep searching the initial cell for the whole duration). Table 18 shows the numerical results for versions 1 and 2 of Algorithm 9. Versions 1 and 2 spend 96.3% and 80.1% of the run time to solve the master problems (in Step 2), respectively. The remaining time is used to generate cuts and build models.

During one hour, the version 1 permits only 20 iterations while version 2 executes 68 iterations. Since the multi-cut requires much more overhead time to generate (multiple) cuts than Algorithm 8 (single-cut) in each iteration, the number of possible iterations is reduced compared to that of Algorithm 8 (i.e., 25 iterations in version 1 of Algorithm 8 are reduced to 20 in version 1 of Algorithm 9; 96 iterations in version 2 of Algorithm 8 are reduced to 68 in version 2 of Algorithm 9). However, for both version 1 and version 2 of Algorithm 9, the total number of accumulated cuts becomes larger than the corresponding versions using a single-cut approach. As a result, the lower bound of the multi-$t2$-cut (versions 1 and 2) improves quicker than that of the single-cut (refer to Table 17). For the upper bound, there is no significant difference between the multi-$t2$-cut and the single-cut approaches. After one hour, the multi-$t2$-cut versions provides better relative gaps than the single-cut versions (i.e., 30.4% (version 1 of Algorithm 8) compared to 28.3% (version 1 of Algorithm 9), and 20.5% (version 2 of Algorithm 8) compared to 19.6% (version 2 of Algorithm 9)). We also note that version 2 ($\epsilon^{(k)} = 0.03$) performs better than the version 1 ($\epsilon^{(k)} = 0$)) as seen from Table 17.

Time	Algo. 9 version 1: $\epsilon^{(k)} = 0$			Algo. 9 version 2: $\epsilon^{(k)} = 0.03$		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.436285	0.588137	0.348	0.470301	0.569675	0.211
20	0.454153	0.588137	0.295	0.474848	0.569675	0.200
30	0.457303	0.588137	0.286	0.475066	0.569675	0.199
40	0.458369	0.588137	0.283	0.475835	0.569675	0.197
50	0.458369	0.588137	0.283	0.476396	0.569675	0.196
60	0.458369	0.588137	0.283	0.476396	0.569675	0.196

Table 18. Numerical results for Algorithm 9 using multi-$t2$-cut (versions 1 and 2) for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17.

Table 19 presents the numerical results for Algorithm 9 using multi-$t3$-cut. In each iteration, the multi-$t3$-cut requires much more time to generate 25 cuts. Thus the versions 3 and 4 spend only 40.1% and 4.7% of the run time on solving the

master problems (in Step 2) and the remaining time generating cuts. During one hour tests, versions 3 and 4 permit only 12 and 19 iterations, respectively. Typically, the multi-$t3$-cut versions provide a significant improvement in the lower bound in each iteration. However, the long run time per iteration prevents them from carrying out "enough" iterations. Thus it appears that the multi-$t3$-cut versions are inferior to the single-cut and multi-$t2$-cut versions (see Tables 17 and 18). We note however that our implementation of Algorithm 9 in GAMS could be improved by implementing the time-consuming cut generation in C++ or some other faster programming language. If the cut generation time could be reduces, the multi-$t3$-cut versions may prove competitive. However, in this dissertation, we do not consider this programming enhancement.

Time	Algo. 9 version 3: $\epsilon^{(k)} = 0$			Algo. 9 version 4: $\epsilon^{(k)} = 0.03$		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.255538	0.711572	1.785	0.247878	0.716509	1.891
20	0.359439	0.624907	0.739	0.343276	0.662075	0.929
30	0.399356	0.604164	0.513	0.384563	0.608274	0.582
40	0.424481	0.604164	0.423	0.419195	0.605206	0.444
50	0.424481	0.604164	0.423	0.435430	0.589771	0.354
60	0.429744	0.594532	0.383	0.450207	0.581592	0.292

Table 19. Numerical results for Algorithm 9 using multi-$t3$-cut (versions 3 and 4) for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17.

Based on the test results in Tables 17, 18 and 19, we conclude that Algorithm 8, version 2 (single-cut with $\epsilon^{(k)}$=0.03), and Algorithm 9, version 4 (multi-$t2$-cut with $\epsilon^{(k)}$=0.03), are the most promising. Hence, we adopt these versions as baselines for further comparisons. When the meaning is clear from the context, we refer to those versions simply as single-cut and multi-cut, respectively.

(b) Strong-cut

In Algorithms 8 and 9, at each iteration, cuts are generated at the optimal solutions $\hat{y}$ obtained from the master problems. The cuts are the tangent hyper-planes

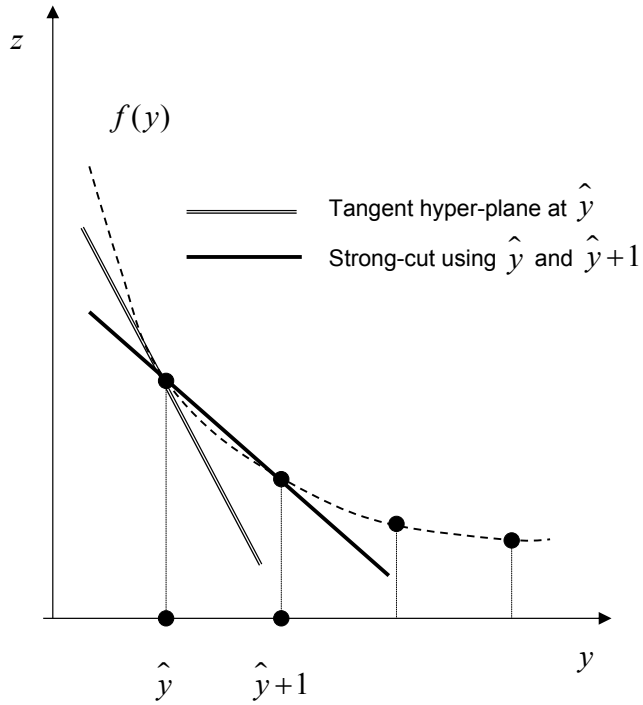

Figure 11. Strong cut and tangent hyper-plane on an exponential objective function in mixed-integer program.

and utilize that $f(y) \geq f(\hat{y}) + \nabla f(\hat{y})'(y - \hat{y})$ for all y. (For simplicity, we now argue using single-cut, but the same arguments can be build for multi-cut versions). We now use the fact we are only concern with integer values of y to build a stronger cut. Figure 11 illustrates the basic idea of the new cut. Since the MHSP is an integer program, we utilize finite differences of the objective function $f(y)$ by considering the perturbation from $\hat{y}_{c,t}$ to $\hat{y}_{c,t} + 1$ while keep all other variables fixed. Theorem IV.1 presents this new cut. Let $\triangle_{c,t} \in \mathcal{Z}^{\mathcal{CT}}$ be a vector in which the (c,t) element is one and the other elements are all zero, where $\mathcal{Z}$ is the set of integers.

Theorem IV.1 *For any* y, $\hat{y} \in \mathcal{Z}^{\mathcal{CT}}$, $f(y) \geq f(\hat{y}) + \sum_{c,t}[f(\hat{y}+\triangle_{c,t}) - f(\hat{y})](y_{c,t} - \hat{y}_{c,t})$.

Proof. $f(y) = \sum_\omega p_\omega \exp(-\sum_{c,t} \alpha^\omega_{c,t} y_{c,t}) = \sum_\omega \exp(-\sum_{c,t} \alpha^\omega_{c,t} y_{c,t} + \log p_\omega)$. Let a^ω be a CT-dimensional vector defined by $a^\omega = (\alpha^\omega_{c,t})$ and $b^\omega = (-\log p_\omega)$. Then, $a^\omega \geq 0$ and $b^\omega > 0$. Hence, $f(y) = \sum_\omega f_\omega(y)$ where $f_\omega(y) = \exp(-a^\omega y - b^\omega)$. In this notation, the result is equivalent to $f_\omega(\hat{y}) + \sum_{c,t}[f_\omega(\hat{y} + \triangle_{c,t}) - f_\omega(\hat{y})](y_{c,t} - \hat{y}_{c,t}) \leq f_\omega(y)$ for all ω. Consequently, $f_\omega(\hat{y})[1 + \sum_{c,t}(\exp(-\alpha^\omega_{c,t}) - 1)(y_{c,t} - \hat{y}_{c,t}) - \exp(-a^\omega(y - \hat{y}))] \leq 0$. Let $\beta^\omega_{c,t} = \exp(-\alpha^\omega_{c,t})$, and $\mathcal{N}$ be the set of the cell-time pair (c,t) such that $y_{c,t} - \hat{y}_{c,t} \neq 0$ and $\alpha^\omega_{c,t} > 0$ (i.e., cell c is on path ω at time period t). Then we only need to show that $\sum_{i\in\mathcal{N}}(1-\beta_i)(y_i - \hat{y}_i) + \prod_{i\in\mathcal{N}}(\beta_i)^{(y_i - \hat{y}_i)} \geq 1$, where $i = (c,t) \in \mathcal{N}$, $0 < \beta_i < 1$. Let $\hat{k}_i = y_i - \hat{y}_i$, then we can define $\varphi(\beta) = \sum_{i\in\mathcal{N}}(1-\beta_i)\hat{k}_i + \prod_{i\in\mathcal{N}} \beta_i^{\hat{k}_i}$, where $\beta = (\beta_i)$ is a $|\mathcal{N}|$-dimensional vector. Let $\psi(\beta) = \prod_{i\in\mathcal{N}} \beta_i^{\hat{k}_i}$. Then $\nabla\psi(\beta) = (\log(\beta_1), ..., \log(\beta_{|\mathcal{N}|}))'\psi(\beta)$, and $\nabla^2\psi(\beta) = A'A\psi(\beta)$, where A is a $|\mathcal{N}|$ by $|\mathcal{N}|$ matrix in which the first low is $(\log(\beta_1), ..., \log(\beta_{|\mathcal{N}|}))$ and the other elements are all zero. Since the matrix $A'A$ is positive semi-definite and $\psi(\beta) > 0$, $\psi(\beta)$ is convex on the set with $\beta_i > 0$ for all $i \in \mathcal{N}$. Thus $\varphi(\beta)$ is also convex on the set with $\beta_i > 0$ for all $i \in \mathcal{N}$. When β_i=1 for $\forall i \in \mathcal{N}$, $\nabla\varphi(\beta) = (\hat{k}_1(-1 + \beta_1^{-1}), ..., \hat{k}_{|\mathcal{N}|}(-1 + \beta^{-1}_{|\mathcal{N}|})) = 0$. Thus, the minimum value of $\varphi(\beta)$ is 1 when β_i=1 for $\forall i \in \mathcal{N}$. Consequently, we obtain that $\varphi(\beta) \geq 1$ which proves the result. □

We refer to this type of new cut as 'strong-cut'. The calculation of finite differences $[f(\hat{y} + \triangle_{c,t}) - f(\hat{y})]$ is as easy as that of the gradient (IV.5). Specifically,

$$
\begin{aligned}
&[f(\hat{y} + \triangle_{c,t}) - f(\hat{y})] \\
&= \sum_{c' \in \mathcal{C}} r_{c',t}(y)[\exp(-\alpha_{c',t} y_{c',t} - \alpha_{c,t}) - \exp(-\alpha_{c',t} y_{c',t})] s_{c',t}(y) \\
&\quad = r_{c,t}(y)[\exp(-\alpha_{c,t}(y_{c,t} + 1)) - \exp(-\alpha_{c,t} y_{c,t})] s_{c,t}(y). \qquad \text{(IV.8)}
\end{aligned}
$$

For the multi-cut version, for each scenario $u = 1, 2, ..., U$, the finite differences $[f_u(\hat{y} + \triangle_{c,t}) - f_u(\hat{y})]$ are calculated similarly:

$$[f_u(\hat{y} + \triangle_{c,t}) - f_u(\hat{y})]$$

$$= \sum_{c' \in \mathcal{C}} r^u_{c',t}(y)[\exp(-\alpha_{c',t} y_{c',t} - \alpha_{c,t}) - \exp(-\alpha_{c',t} y_{c',t})] s^u_{c',t}(y)$$

$$= r^u_{c,t}(y)[\exp(-\alpha_{c,t}(y_{c,t} + 1)) - \exp(-\alpha_{c,t} y_{c,t})] s^u_{c,t}(y). \quad \text{(IV.9)}$$

Algorithm 10 describes a single-cut OA algorithm using the strong-cut. Algorithm 10 is a modification from Algorithm 8 since at each iteration a strong cut is generated instead of a tangent hyper-plane. In the k-th iteration of Algorithm 10, we solve a master problem MP3(k) defined below, where the optimal value and optimal solution are denoted $z^{(k)}$ and $y^{(k)}$, respectively.

Formulation of Master problem : MP3(k)

$min\ z$
$s.t.$
$z \geq f(y^{(i)}) + \sum_{c,t}[f(y^{(i)} + \triangle_{c,t}) - f(y^{(i)})](y_{c,t} - y^{(i)}_{c,t}) \quad i = 0, 1, ..., k-1$
$\sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T$
$y_{c,t} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T}$
$x_{c,c',t},\ y_{c,t}$: integer

Algorithm 10 (OA Algorithm with single, strong-cut).

Step 0. Set a relative optimality tolerance $\delta \geq 0$. Choose an initial feasible solution $y^{(0)}$.

Step 1. Calculate $f(y^{(0)})$ and $f(y^{(0)} + \triangle_{c,t}) - f(y^{(0)})$ for $\forall(c,t)$ (see (IV.4) and (IV.8)). Set lower bound $\underline{q} = 0$, upper bound $\bar{q} = f(y^{(0)})$, and $k = 1$.

Step 2. If the gap $(\bar{q} - \underline{q})/\underline{q} \leq \delta$, stop. Else, solve the master problem MP3(k), and obtain its optimal value $z^{(k)}$ and optimal solution $y^{(k)}$. If $z^{(k)} > \underline{q}$, then $\underline{q} = z^{(k)}$.

Step 3. Calculate $f(y^{(k)})$ and $f(y^{(k)} + \triangle_{c,t}) - f(y^{(k)})$ for $\forall(c,t)$ (see (IV.4) and (IV.8)).

Step 4. If $f(y^{(k)}) < \bar{q}$, then $\bar{q} = f(y^{(k)})$. Replace k by $k+1$, and go to Step 2.

Algorithm 11 shows the multi-cut version of Algorithm 10. In the k-th iteration of Algorithm 11, the master problem MP4(k) defined below is solved. The optimal value and optimal solution of MP4(k) are denoted $z^{(k)}$ and $y^{(k)}$, respectively. In each iteration of Algorithm 11, U cuts are generated at once.

Formulation of Master problem : MP4(k)

$$
\begin{aligned}
&min\ z = \textstyle\sum_{u=1}^{U} \tilde{p}_u z_u \\
&\ s.t. \\
&\ z_u \geq f_u(y^{(i)}) + \textstyle\sum_{c,t}[f_u(y^{(i)} + \triangle_{c,t}) - f_u(y^{(i)})](y_{c,t} - y_{c,t}^{(i)}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad u = 1, 2, ..., U,\ i = 0, 1, ..., k-1 \\
&\ \textstyle\sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T \\
&\ y_{c,t} = \textstyle\sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T} \\
&\ x_{c,c',t},\ y_{c,t}\text{: integer}
\end{aligned}
$$

Algorithm 11 (OA Algorithm with multi/strong-cut).

Step 0. Set a relative optimality tolerance $\delta \geq 0$. Choose an initial feasible solution $y^{(0)}$.

Step 1. Calculate $f_u(y^{(0)})$ and $f_u(y^{(0)} + \triangle_{c,t}) - f_u(y^{(0)})$, $u = 1, 2, ..., U,\ \forall (c,t)$ (see (IV.6) and (IV.9)). Set lower bound $\underline{q} = 0$, upper bound $\bar{q} = \sum_{u=1}^{U} \tilde{p}_u f_u(y^{(0)})$, and $k = 1$.

Step 2. If the gap $(\bar{q} - \underline{q})/\underline{q} \leq \delta$, stop. Else, solve the master problem MP4(k), and obtain its optimal value $z^{(k)}$ and optimal solution $y^{(k)}$. If $z^{(k)} > \underline{q}$, then $\underline{q} = z^{(k)}$.

Step 3. Calculate $f_u(y^{(k)})$ and $f_u(y^{(k)} + \triangle_{c,t}) - f_u(y^{(k)})$, $u = 1, 2, ..., U,\ \forall (c,t)$ (see (IV.6) and (IV.9)).

Step 4. If $f(y^{(k)}) = \sum_{u=1}^{U} \tilde{p}_u f_u(y^{(k)}) < \bar{q}$, then $\bar{q} = f(y^{(k)})$. Replace k by $k+1$, and go to Step 2.

We demonstrate the effect of the strong-cut using the same problem instance accompanied with the same assumptions and parameter settings as in the previous subsection. The trivial search plan is still used as the initial feasible solution $y^{(0)}$. The relative optimality tolerance of the master problems is assumed constant at 0.03.

Table 20 describes the effect of the strong-cut. Compared with the previous algorithms (Algorithms 8 and 9), Algorithms 10 and 11 implement 1.4 times more iterations: 96 for Algorithm 8 compared to 135 for Algorithm 10, and 68 for Algorithm 9 compared to 95 for Algorithm 11. Furthermore, the relative gap after one hour is drastically reduced from about 20% to 10.5% for both versions. Since the increase of the number of iterations and the progress in the gap are essentially identical for both versions, we conclude that the strong-cut is quite effective. The performance of Algorithms 10 and 11 during one hour of calculations is almost parallel to each other.

Time	Algo. 10 : single-cut			Algo. 11 : multi-cut		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.509594	0.570631	0.120	0.510988	0.571003	0.117
20	0.511369	0.570631	0.116	0.512539	0.569664	0.111
30	0.512452	0.568363	0.109	0.513929	0.569664	0.108
40	0.512819	0.568363	0.108	0.514792	0.569664	0.107
50	0.512847	0.568363	0.108	0.515407	0.569664	0.105
60	0.514277	0.568363	0.105	0.515407	0.569664	0.105

Table 20. Numerical results for Algorithms 10 and 11 using strong-cut for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17.

(c) Choice of Initial Solution

So far, as a choice of initial feasible solution, we have used the trivial search plan (i.e., all searchers keep loitering at the initial cell). The choice of the initial solution certainly influences the OA algorithms, especially during the initial calculations. In this section, we aim to develop a "good" initial solution. In advance of this, we first introduce a theoretical property of the MHSP, which facilitates the calculation of a good initial solution.

We now assume that the detection rate is time and space independent, i.e., $\alpha = \alpha_{c,t}$ for all c and t. Under this assumption, we show that the MHSP is equivalent to a large-scale linear mixed-integer program. Let $W_\omega = \sum_{c,t} \alpha^\omega_{c,t} y_{c,t}$ be the total effective search effort given search plan y when the target takes path $\omega \in \Omega$. Since

the detection rate is assumed identical, W_ω is a multiple of α and thus $W_\omega = m\alpha$ for some $m = 0, 1, ..., JT$. This is a similar idea to [52]. The maximum possible total effective search effort is the number of searchers (J) times the time horizon (T), which occurs when all searchers take the path ω. In this notation, we claim that the MHSP is formulated as the following large-scale linear mixed-integer program.

Formulation in linear mixed-integer program: L1

Indices

c, c'	cells ($c, c' \in \mathcal{C} = \{1, \ldots, C\}$)
t	time step ($t \in \mathcal{T} = \{0, 1, ..., T\}$)
j	searchers ($j \in \mathcal{J} = \{1, ..., J\}$)
ω	target path ($\omega \in \Omega$)

Parameters

α	detection rate for a single searcher in cell c and time period t
$\alpha^\omega_{c,t}$	$\alpha_{c,t}$ if cell c is on target path ω at time period t, zero otherwise.
$y_{c,0}$	number of searchers in cell c in time period 0
p_ω	probability that the target takes path ω

Variables

$x_{c,c't}$	number of searchers that is redistributed from cell c in time period t to cell c' in time period $t+1$
$y_{c,t}$	number of searchers in cell c in time period t

Formulation

$$min \ \sum_{\omega \in \Omega} p_\omega z_\omega$$

s.t.

$$W_\omega = \sum_{c,t} \alpha^\omega_{c,t} y_{c,t}$$

$$z_\omega \geq e^{-m\alpha}[1 + m - me^{-\alpha}] + (1/\alpha)e^{-m\alpha}(e^{-\alpha} - 1)W_\omega \qquad \forall \omega \in \Omega, \quad m = 0, 1..., JT$$

$$\sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T$$

$$y_{c,t} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T}$$

$$x_{c,c',t}, \ y_{c,t}: \text{ integer}$$

Theorem IV.2 *If the detection rate is identical (i.e., $\alpha = \alpha_{c,t}$ for all c and t), the convex nonlinear mixed-integer MHSP and the linear mixed-integer program L1 have identical global minimum solutions.*

Proof. Let $f_\omega(W_\omega) = e^{-W_\omega}$, where $W_\omega = \sum_{c,t} \alpha^\omega_{c,t} y_{c,t}$ and $f(y) = \sum_{\omega \in \Omega} p_\omega f_\omega(W_\omega)$. W_ω only takes a finite number of discrete value $W_\omega = m\alpha$, $m = 0, 1, ..., JT$. Thus $f_\omega(W_\omega)$ can be described as a piece-wise linear function (see Figure 12). The linear

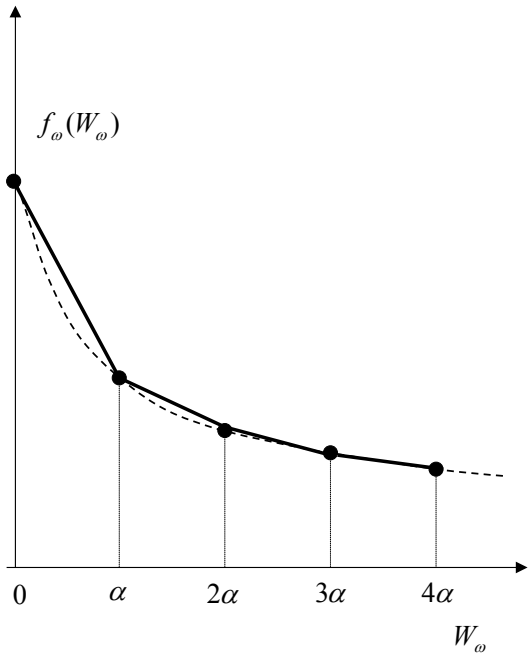

Figure 12. Piece-wise linearlization of the exponential function $f_\omega(W_\omega)$.

approximation between $m\alpha$ and $(m+1)\alpha$ is obtained as $f(W_\omega) = e^{-m\alpha}[1 + m - me^{-\alpha}] + (1/\alpha)e^{-m\alpha}(e^{-\alpha} - 1)W_\omega$ by solving the linear equation $(b_0 + b_1[m\alpha] = e^{-m\alpha}$ and $b_0 + b_1[(m+1)\alpha] = e^{-(m+1)\alpha})$ for parameters b_0 and b_1. □

The linearization of MHSP under the assumption of identical detection rates explicitly depends on all possible target paths. Hence, the program L1 may become extremely large. Thus we cannot directly apply this formulation to solve most instances of MHSP. However, this linearization provides useful insight to obtain a good initial solution.

The linearlization of MHSP considers the total effective search effort W_ω on each target path $\omega \in \Omega$ individually. The alternative approach is to consider the

total effective search effort aggregated over all target paths. This approach allow us to utilize the Markovian property of the target movement, but it leads to a relaxation as we now described. Recall that the target moves between cells according to a known transition matrix Γ. Let $p(\cdot,t) = p(\cdot,1)\Gamma^{t-1}$, $t > 1$ be the undetected target distribution in time period t, ($p(\cdot,1)$ is known). Thus the aggregated total effective search effort is described as $W = \sum_{c,t} p(c,t)\alpha y_{c,y}$. In this notation, the aggregation of the large-scale linear mixed-integer program L1 takes the form of the moderately sized linear mixed-integer program L2:

Formulation in linear mixed-integer program (aggregation): L2

Indices

c, c'	cells ($c, c' \in \mathcal{C} = \{1, \ldots, C\}$)
t	time step ($t \in \mathcal{T} = \{0, 1, ..., T\}$)
j	searchers ($j \in \mathcal{J} = \{1, ..., J\}$)

Parameters

α	detection rate for a single searcher in any cell and time
$y_{c,0}$	number of searchers in cell c in time period 0
$p(\cdot,t)$	undetected target distribution at time period t

Variables

$x_{c,c't}$	number of searchers that is redistributed from cell c in time period t to cell c' in time period $t+1$
$y_{c,t}$	number of searchers in cell c in time period t

Formulation

$min\ z$

s.t.

$W = \sum_{c,t} p(c,t)\alpha y_{c,y}$

$z \geq e^{-m\alpha}[1 + m - me^{-\alpha}] + (1/\alpha)e^{-m\alpha}(e^{-\alpha} - 1)W,\ m = 0, 1..., JT$

$\sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T$

$y_{c,t} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T}$

$x_{c,c',t},\ y_{c,t}$: integer

We observe that solutions of L2 is not identical to solutions of L1 as L2 is a relaxation of L1 due to the aggregation of all the target paths. However, L2 is easily solved and typically provides a good initial solution for the main calculations. We will use this aggregated program L2 to obtain an initial solution.

In addition, we also consider a simple condition (constraint) to obtain a better initial solution. If the number of searchers is moderate (e.g., 3 searchers) and the size of the area is large, it is typically better that the searchers visit cells not previously searched. (Of course, the effect of this strategy also depends on the movement of the target.) Given a specific $t' \in \mathcal{T}$ (e.g., t'=1 for the problem instance with 3 searchers), a non-revisit constraint is described as $\sum_{t>t'} y_{c,t} \leq 1$, $\forall c \in \mathcal{C}$. Therefore, in the initial step (Step 0) of the OA algorithms, we solve the aggregation problem (L2) with this non-revisit constraint to obtain an initial solution. This initial problem is referred as L3 as follows.

Initial problem: L3

Indices

c, c' cells ($c, c' \in \mathcal{C} = \{1, \ldots, C\}$)
t time step ($t \in \mathcal{T} = \{0, 1, ..., T\}$)
j searchers ($j \in \mathcal{J} = \{1, ..., J\}$)

Parameters

α detection rate for a single searcher in any cell and time
$y_{c,0}$ number of searchers in cell c in time period 0
$p(\cdot, t)$ undetected target distribution at time period t

Variables

$x_{c,c't}$ number of searchers that is redistributed from cell c in time period t to cell c' in time period $t+1$
$y_{c,t}$ number of searchers in cell c in time period t

Formulation

$min\ z$
s.t.
$W = \sum_{c,t} p(c,t)\alpha y_{c,y}$
$z \geq e^{-m\alpha}[1 + m - me^{-\alpha}] + (1/\alpha)e^{-m\alpha}(e^{-\alpha} - 1)W$, $m = 0, 1..., JT$
$\sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t}$ $\forall t = 1, ..., T$
$y_{c,t} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t}$ $\forall t \in \mathcal{T}$
$\sum_{t>t'} y_{c,t} \leq 1$, $\forall c \in \mathcal{C}$
$x_{c,c',t}$, $y_{c,t}$: integer

In the OA algorithms, the first cut(s) is generated at the initial solution obtained from the initial problem L3. In addition, we also add the cut at the solution $y = 0$ (i.e., $f(y) \geq f(0) + \nabla f(0)'y$) which can be shown to relate to the mean bound

in a branch-and-bound procedure [34, 52]. The cut at $y = 0$ linearlizes the probability of non-detection at the point of "no search" effect so it gives an estimate of initial effect of increasing the search effort above zero in a cell. We apply the strong-cut, not the gradient-based hyper plane, at $y = 0$ and the initial solution found when solving L3. We note that this initial process is also available for the algorithm using multi-cut (Algorithm 11) and is referred as the initial problem L4 as follows.

Initial problem: L4

Indices

c, c'	cells ($c, c' \in \mathcal{C} = \{1, \ldots, C\}$)
t	time step ($t \in \mathcal{T} = \{0, 1, ..., T\}$)
j	searchers ($j \in \mathcal{J} = \{1, ..., J\}$)
u	scenario ($u = 1, 2, ..., U$)

Parameters

α	detection rate for a single searcher in any cell and time
$y_{c,0}$	number of searchers in cell c in time period 0
$\tilde{p}_u$	probability that the target takes scenario u
$p^u(\cdot, t)$	undetected target distribution at time period t when scenario u

Variables

$x_{c,c't}$	number of searchers that is redistributed from cell c in time period t to cell c' in time period $t + 1$
$y_{c,t}$	number of searchers in cell c in time period t
W_u	total effective search effort when scenario u
z_u	

Formulation

$$min \ \sum_{u=1}^{U} \tilde{p}_u z_u$$

s.t.

$$W_u = \sum_{c,t} p^u(c,t) \alpha y_{c,y}, \quad u = 1, 2, ..., U$$

$$z_u \geq e^{-m\alpha}[1 + m - me^{-\alpha}] + (1/\alpha)e^{-m\alpha}(e^{-\alpha} - 1)W_u \qquad u = 1, 2, ..., U, \ m = 0, 1..., JT$$

$$\sum_{c' \in \mathcal{R}(c)} x_{c',c,t-1} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t = 1, ..., T$$

$$y_{c,t} = \sum_{c' \in \mathcal{F}(c)} x_{c,c',t} \quad \forall t \in \mathcal{T}$$

$$\sum_{t > t'} y_{c,t} \leq 1, \ \forall c \in \mathcal{C}$$

$x_{c,c',t}, \ y_{c,t}$: integer

Table 21 reports the effect of the choice of an improved initial solution using the same problem instance with the same assumptions and parameter settings as above. We define Algorithm 10.2 (single-cut) and Algorithm 11.2 (multi-cut) when

an initial solution is computed from L3 in Algorithm 10 and from L4 in Algorithm 11, respectively. For both Algorithms 10.2 and 11.2, the relative gap decreases after 10 minutes from 0.107 to 0.103 for Algorithm 10.2, and from 0.105 to 0.098 for Algorithm 11.2. However, after 30 minutes of calculations, the advantage of the improved initial solution becomes insignificant since the numbers of cuts are added and the cuts largely influence the solution quality. Algorithms 10.2 and 11.2 perform better than the previous cases using trivial initial solution, thus we apply them as the baseline for the OA algorithms which include the next set of new cuts.

Time	Algo. 10.2 : single-cut			Algo. 11.2 : multi-cut		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.510520	0.568875	0.114	0.512064	0.567808	0.109
20	0.512370	0.568875	0.110	0.513271	0.567808	0.106
30	0.512422	0.568875	0.110	0.514057	0.567808	0.105
40	0.513546	0.568875	0.108	0.514660	0.567808	0.103
50	0.513747	0.568875	0.107	0.515650	0.567808	0.101
60	0.513747	0.568875	0.107	0.515900	0.567808	0.101

Table 21. Numerical results for Algorithms 10.2 and 11.2 using an improved initial solution for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17.

(d) Relative-cut

In Table 21, the upper bound (i.e., the best feasible value) does not change after ten minutes. Thus it is possible that the upper bound could have been optimal. (Of course, it is not in this case as we know a better value from Chapter III). Let $\bar{y}$ be the best solution so far providing the upper bound $\bar{q} = f(\bar{y})$. Since the objective function is convex, $\bar{y}$ is guaranteed to be optimal if $f(\bar{y})$ is still the upper bound after cuts are added at all neighbor integer points $\tilde{y}$ of $\bar{y}$ (the Euclidian distance between $\bar{y}$ and $\tilde{y}$ is one). Thus, we explore the strategy of generating cuts around the best solution.

Figure 13 illustrates the rule that determines a neighboring integer point $\tilde{y}$ where we generate a cut. In earlier the OA algorithms, each iteration obtains a

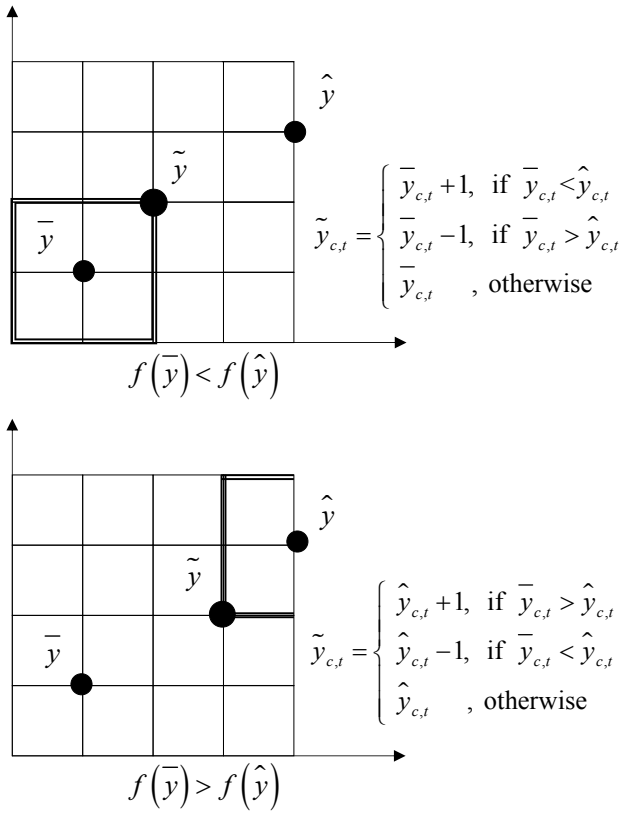

Figure 13. Illustration of cut generation around the best solution.

solution $\hat{y}$ and generates one or more cuts at $\hat{y}$. Now, we also generate an additional cut at $\tilde{y}$. Thus, in the algorithms using the single-cut, two cuts are generated in each iteration. The upper picture in Figure 13 shows a case where the best solution so far $\bar{y}$ is still better than the current solution $\hat{y}$ (i.e., $f(\bar{y}) < f(\hat{y})$). Then, we determin an integer point $\tilde{y}$ which is a neighbor of $\bar{y}$ *and* is the closest point to $\hat{y}$ measured in the Euclidean distance. The lower picture illustrates the opposite case. In this case, after generating the cuts at $\hat{y}$ and $\tilde{y}$, the best solution $\bar{y}$ is updated as $\hat{y}$. We denote the cut at the solution $\tilde{y}$ as a "relative-cut. " Moreover, for the relative-cut, we also apply the strong-cut instead of the gradient-based hyper-plane cut.

The OA algorithms using the relative-cut is described as Algorithm 12 and Algorithm 13. Algorithm 12 is the single-cut version, and Algorithm 13 is the multi-

cut version. Algorithms 12 and 13 also use the initial problem L3 and L4 to obtain a reasonable initial solution, respectively. Thus Algorithms 12 and 13 are extension of Algorithms 10.2 and 11.2 to the ones with relative-cut, respectively.

Algorithm 12 (OA Algorithm with single/strong/relative-cut).

Step 0. Set a relative optimality tolerance $\delta \geq 0$ and $y^{(0)} = 0$. Solve the initial problem L3 ($t' = 1$ for the problem instance with 3 searchers) and set the solution as $y^{(1)}$.

Step 1. Calculate $f(y^{(0)})$, $f(y^{(1)})$, $f(y^{(0)} + \triangle_{c,t}) - f(y^{(0)})$ and $f(y^{(1)} + \triangle_{c,t}) - f(y^{(1)})$ for $\forall(c,t)$ (see (IV.4) and (IV.8)). Set lower bound $\underline{q} = 0$, upper bound $\bar{q} = f(y^{(1)})$, $\bar{y} = y^{(1)}$, and $k = 2$.

Step 2. If the gap $(\bar{q} - \underline{q})/\underline{q} \leq \delta$, stop. Else, solve the master problem MP3(k) (described above Algorithm 10), and obtain its optimal value $z^{(k)}$ and optimal solution $\hat{y} = y^{(k)}$. If $z^{(k)} > \underline{q}$, then $\underline{q} = z^{(k)}$.

Step 3. Calculate $f(y^{(k)})$ and $f(y^{(k)} + \triangle_{c,t}) - f(y^{(k)})$ for $\forall(c,t)$ (see (IV.4) and (IV.8)).

Step 4. Obtain a neighbor point $\tilde{y}$ around the best solution by comparing $f(\hat{y})$ with $\bar{q} = f(\bar{y})$. Replace k by $k+1$. Set $y^{(k)} = \tilde{y}$.

Step 5. Calculate $f(y^{(k)})$ and $f(y^{(k)} + \triangle_{c,t}) - f(y^{(k)})$ for $\forall(c,t)$.

Step 6. If $f(\hat{y}) < \bar{q}$, then $\bar{q} = f(\hat{y})$ and $\bar{y} = \hat{y}$. Replace k by $k+1$, and go to Step 2.

Algorithm 13 (OA Algorithm with multi/strong/relative-cut).

Step 0. Set a relative optimality tolerance $\delta \geq 0$ and $y^{(0)} = 0$. Solve the initial problem L4 ($t' = 1$ for the problem instance with 3 searchers) and set the solution as $y^{(1)}$.

Step 1. Calculate $f_u(y^{(0)})$, $f_u(y^{(1)})$, $f_u(y^{(0)} + \triangle_{c,t}) - f_u(y^{(0)})$ and $f_u(y^{(1)} + \triangle_{c,t}) - f_u(y^{(1)})$, for $u = 1, 2, ..., U$, $\forall(c,t)$ (see (IV.6) and (IV.9)). Set lower bound $\underline{q} = 0$, upper bound $\bar{q} = \sum_{u=1}^{U} \tilde{p}_u f_u(y^{(1)})$, $\bar{y} = y^{(1)}$, and $k = 2$.

Step 2. If the gap $(\bar{q} - \underline{q})/\underline{q} \leq \delta$, stop. Else, solve the master problem MP4(k) (described above Algorithm 11), and obtain its optimal value $z^{(k)}$ and optimal solution $\hat{y} = y^{(k)}$. If $z^{(k)} > \underline{q}$, then $\underline{q} = z^{(k)}$.

Step 3. Calculate $f_u(y^{(k)})$ and $f_u(y^{(k)}+\triangle_{c,t})-f_u(y^{(k)})$, $u=1,2,...,U,\ \forall(c,t)$ (see (IV.6) and (IV.9)). Calculate $f(\hat{y})=\sum_{u=1}^{U}\tilde{p}_u f_u(\hat{y})$

Step 4. Obtain a neighbor point $\tilde{y}$ around the best solution by comparing $f(\hat{y})$ with $\bar{q}=f(\bar{y})$. Replace k by $k+1$. Set $y^{(k)}=\tilde{y}$.

Step 5. Calculate $f_u(y^{(k)})$ and $f_u(y^{(k)}+\triangle_{c,t})-f_u(y^{(k)})$, $u=1,2,...,U,\ \forall(c,t)$.

Step 6. If $f(\hat{y})<\bar{q}$, then $\bar{q}=f(\hat{y})$ and $\bar{y}=\hat{y}$. Replace k by $k+1$, and go to Step 2.

We test the effect of the relative-cut using the same problem instance with the same assumptions and parameter settings as above. Table 22 describes the computational results of Algorithms 12 and 13. During one hour of calculations, we do not identify significant effect of the relative-cut by comparing Table 21 and Table 22. However, the relative-cut strategy seems not to be detrimental and it allows for a faster accumulation of cuts which may be beneficial so we implement the relative-cut in our OA algorithms.

Time	Algo. 12: single-cut			Algo. 13: multi-cut		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.510523	0.571749	0.120	0.512066	0.577300	0.127
20	0.512079	0.569864	0.113	0.514172	0.570003	0.109
30	0.513185	0.569864	0.110	0.514721	0.566784	0.101
40	0.514212	0.569864	0.108	0.514806	0.566784	0.101
50	0.514538	0.569864	0.108	0.515639	0.566784	0.099
60	0.514538	0.569864	0.108	0.515690	0.566784	0.099

Table 22. Numerical results for Algorithms 12 and 13 using relative-cut for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17.

(e) Symmetric-cut

The final cut we present is quite specific and not applicable for general cases. However it is somewhat effective in specific situation. Suppose that the shape of the area of interest (AOI) is symmetric and the searchers and the targets are initially in the cells which are located on a line that divides the AOI symmetrically (see Figure 14). Suppose also that the detection rate is identical for all cells and time periods.

And finally, suppose that the target moves to an adjacent cells equally likely. We refer to this specific situation as "symmetric." In the symmetric situation, for a search plan $\overline{\mathcal{P}}$, there exist a symmetric (or mirror) plan $\overline{\mathcal{P}}'$ with identical probability of nondetection, i.e., $q(\overline{\mathcal{P}}) = q(\overline{\mathcal{P}}')$, see Figure 14.

Thus, the basic idea of the symmetric-cut is that, in each iteration, cuts are generated not only at the current solution (plan) $\overline{\mathcal{P}}$ but also at its symmetric solution (plan) $\overline{\mathcal{P}}'$ since the symmetric solution can be evaluated from the current solution. Algorithm 14 describe the OA algorithm using the symmetric-cut, which is an extension of Algorithm 12. In Algorithm 12, at each iteration, two cuts are generated at the current solution $\hat{y}$ and a neighbor point $\tilde{y}$ around the best solution (by comparing $f(\hat{y})$ with $f(\bar{y})$). In Algorithm 14, two cuts are additionally generated at the symmetric solution y' and a neighbor point $\tilde{y}'$ around the best solution (by comparing $f(y')$ with $f(\bar{y})$). Thus, in Algorithm 14, four cuts are added at each iteration.

The problem instance examined in numerical test so far is symmetric. Thus we attempt to examine the effect of the symmetric cut in this instance. Since the multi-cut utilizes the conditioning of the target movement, it violates the symmetry property. Thus the symmetric cut is available in the algorithm using the single-cut only.

Algorithm 14 (OA Algorithm with single/strong/relative/symmetric-cut).

Step 0. Set a relative optimality tolerance $\delta \geq 0$ and $y^{(0)} = 0$. Solve the initial problem L3 (described below Algorithm 12, and $t' = 1$ for the problem instance with 3 searchers) and set the solution as $y^{(1)}$.

Step 1. Calculate $f(y^{(0)})$, $f(y^{(1)})$, $f(y^{(0)} + \triangle_{c,t}) - f(y^{(0)})$ and $f(y^{(1)} + \triangle_{c,t}) - f(y^{(1)})$ for $\forall(c,t)$ (see (IV.4) and (IV.8)). Set lower bound $\underline{q} = 0$, upper bound $\bar{q} = f(y^{(1)})$, $\bar{y} = y^{(1)}$, and $k = 2$.

Step 2. If the gap $(\bar{q} - \underline{q})/\underline{q} \leq \delta$, stop. Else, solve the master problem MP3(k) (described above Algorithm 10), and obtain its optimal value $z^{(k)}$ and optimal solution $\hat{y} = y^{(k)}$. If $z^{(k)} > \underline{q}$, then $\underline{q} = z^{(k)}$.

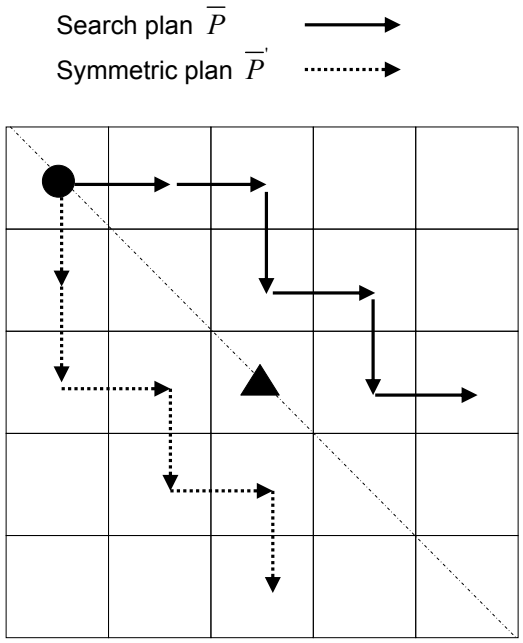

Figure 14. Symmetric environment where the symmetric-cut is available.

Step 3. Calculate $f(y^{(k)})$ and $f(y^{(k)} + \triangle_{c,t}) - f(y^{(k)})$ for $\forall(c,t)$ (see (IV.4) and (IV.8)).

Step 4. Obtain a neighbor point $\tilde{y}$ around the best solution by comparing $f(\hat{y})$ with $\bar{q} = f(\bar{y})$. Replace k by $k+1$. Set $y^{(k)} = \tilde{y}$.

Step 5. Calculate $f(y^{(k)})$ and $f(y^{(k)} + \triangle_{c,t}) - f(y^{(k)})$ for $\forall(c,t)$.

Step 6. For the solution $\hat{y}$, obtain the symmetric solution $\hat{y}'$. Replace k by $k+1$. Set $y^{(k)} = \hat{y}'$.

Step 7. Calculate $f(y^{(k)})$ and $f(y^{(k)} + \triangle_{c,t}) - f(y^{(k)})$ for $\forall(c,t)$.

Step 8. Obtain a neighbor point $\tilde{y}'$ around the best solution by comparing $f(\tilde{y}')$ with $\bar{q} = f(\bar{y})$. Replace k by $k+1$. Set $y^{(k)} = \tilde{y}'$.

Step 9. Calculate $f(y^{(k)})$ and $f(y^{(k)} + \triangle_{c,t}) - f(y^{(k)})$ for $\forall(c,t)$.

Step 10. If $f(\hat{y}) < \bar{q}$, then $\bar{q} = f(\hat{y})$ and $\bar{y} = \hat{y}$. Replace k by $k+1$ and go to Step 2.

Table 23 reports the effect of the symmetric-cut as applied to the same problem instance as in Table 22. Comparing with Algorithm 12 in Table 22, the relative gap after one hour of calculations is reduced from 0.097 to 0.092. Both the lower and upper bound are improved. Thus, among the OA algorithms using the single-cut, Algorithm 14 is the best for this problem instance.

Time	Algo. 14: Symmetric-cut		
(min)	LB	UB	Gap
10	0.512580	0.574371	0.121
20	0.514023	0.572793	0.114
30	0.515390	0.568628	0.103
40	0.515390	0.568628	0.103
50	0.515812	0.568628	0.102
60	0.516447	0.568628	0.101

Table 23. Numerical results for Algorithms 14 using symmetric-cut for every 10 minutes of calculations. The best known non-detection probability is 0.563472 same as in Table 17.

We examined several OA algorithms in which the new cuts (multi-, strong-, relative- and symmetric-cuts) are cumulatively applied. Specifically, Algorithm 13 (multi-cut) and Algorithm 14 (single-cut) perform best among the algorithms we examined. For the problem instance (15 by 15 cells, time horizon 18 and 3 searchers) with specific assumptions and parameters settings (e.g., identical detection rate and symmetric situation, etc.), Algorithms 13 and 14 provide an approximate solution with relative optimality gap about 10% after one hour of calculations. In Chapter III, for the same problem instance (see Tables 15 and 16), the Cross-Entropy (CE) heuristic provides a better solution 0.563472 (=1-0.436528) after 674.85 seconds. Thus the CE heuristic performs better for this one-hour computational test, but, of course the CE heuristic does not provide solution quality guarantees. Thus a hybrid method using the CE heuristic and the OA algorithm is expected to be more effective: the CE heuristic is initially applied to obtain a good initial feasible solution followed by iterations of the OA algorithm. This dissertation does not pursue this hybrid scheme.

3. Numerical Study of Three Searchers

In the previous subsection, we developed specific OA algorithms (i.e., Algorithm 13 and Algorithm 14 among others) to solve the MHSP. For the moderately sized problem instance (15 by 15 cells, time horizon 18 and 3 searchers) examined, the OA algorithms provided a solution about 10% of the optimal solution in one hour. We now examine the run time of Algorithm 13 on a more extensive set of problem instances of somewhat smaller size. Results for Algorithm 14 is not presented, but they are almost identical.

As problem instances, we consider the following five cases with three searchers in which we apply the same assumptions and parameter settings as the problem instance in the previous subsection, except for changing the area size and the time horizon.

- case 1: 15 by 15 cells and time horizon 16
- case 2: 13 by 13 cells and time horizon 14
- case 3: 11 by 11 cells and time horizon 12
- case 4: 9 by 9 cells and time horizon 10
- case 5: 7 by 7 cells and time horizon 8

For each case, we run Algorithm 13 for two hours. Since the problem instances are small, the master problems MP4(k) in Algorithm 13 are expected to be solved quickly. Thus, we allow a smaller relative optimality tolerance $\epsilon^{(k)}$ for the solver of MP4(k). Specifically, we find empirically that the following simple rule works well: set $\epsilon^{(k)}$= min$\{0.03,(\bar{q}-\underline{q})/(3\underline{q})\}$. In short, at each iteration, we use the optimality tolerance 0.03, but apply one third of the latest gap $(\bar{q}-\underline{q})/\underline{q}$ after the gap has dropped below 9%.

Table 24 reports the gaps at every ten minute during two-hours numerical tests using Algorithm 13 on these five cases. Essentially all cases obtain solution within 5% of optimality within 20 minutes. The less the size of the problem instance is,

the smaller gaps are achieved. Case 5 results in an essentially optimal solution after two hours. After two hours, the lower and upper bounds in that case are 0.454141 and 0.455054, respectively. Thus Algorithm 13 is quite effective for these problem instances and provides near-optimal solutions in reasonable computing times.

Time	Gap				
(min)	case1	case2	case3	case4	case5
10	0.056	0.050	0.039	0.025	0.022
20	0.051	0.047	0.034	0.020	0.016
30	0.049	0.044	0.031	0.018	0.013
40	0.048	0.041	0.029	0.016	0.011
50	0.048	0.040	0.028	0.015	0.009
60	0.047	0.038	0.027	0.014	0.007
70	0.046	0.038	0.026	0.013	0.006
80	0.044	0.037	0.026	0.012	0.005
90	0.041	0.037	0.026	0.012	0.004
100	0.041	0.036	0.025	0.011	0.004
110	0.041	0.036	0.025	0.010	0.003
120	0.041	0.036	0.024	0.009	0.002

Table 24. Relative optimality gaps for Algorithm 13 applied three searcher problem instances during two hours of calculations.

4. Application with Large Number of Searchers

This subsection considers the MHSP with more searchers, especially 5, 10, 15, and 30 searchers. Conceptually, the OA algorithms can treat the HMSP with a large number of searchers more easily than the Cross-Entropy (CE) heuristics since they do not require the generation of specific network structures. Moreover, discussed and demonstrated below, the OA algorithms actually may benefit from more searchers as the continuous relaxation of the master problems may becomes stronger and thus reduce the master problem solution times. In this subsection, we examine the effect on the OA algorithms with respect to the number of searchers. For the computational tests, we use the same problem instance (15 by 15 cells and time horizon 18) as in the previous subsection 2, except we change the number of searchers. The other assumptions and the parameter settings are identical except that: (i) The identical

detection rate α is adjusted according to the number of searchers. For example, for the case with 30 searchers, detection rate 0.1α is used to make the search effect equivalent to the 3 searcher case (i.e., $3\alpha = 30 \cdot 0.1\alpha$). (ii) The optimality tolerance $\epsilon^{(k)}$ for the master problem solver is adjusted adaptively by setting $\epsilon^{(k)}= \min\{0.03,(\bar{q} - \underline{q})/(3\underline{q})\}$. For the cases with a large number of searchers, the master problem (mixed-integer program: MIP) becomes almost equivalent to the continuous relaxation of the MIP. Thus, the master problem with small tolerance (e.g., 0.01) can be solved quickly thus we do not need a large optimality tolerance (0.03). On the other hand, the MIP with much smaller optimality tolerance (e.g., 0.001) is quite time-consuming to be solved. (3) In the initial problem to find a good initial solution, the non-revisit constraint is ignored since that constraint may be detrimental in situations with a large numbers of searchers in a small and moderately sized areas.

We examine the following four cases are examined on the problem instance with 15 by 15 cells and time horizon 18:

- case 1: 5 searchers
- case 2: 10 searchers
- case 3: 15 searchers
- case 4: 30 searchers

Since the number of searchers is fairly large, a continuous relaxation of the MHSP becomes almost equivalent to the original integer MHSP problem. A continuous relaxation means that the integrality restriction in MHSP is relaxed, which implies that the searchers can be "partitioned" arbitrarily. We denote the instances corresponding to cases 1-4 with the continuous relaxation for cases 1r-4r.

For the computational tests, we use Algorithm 13 (multi-cut version) but Algorithm 14 generates almost identical results. We refer to Algorithm 13 without integer restrictions in the master problem for Algorithm 15. Clearly, Algorithm 15 may obtain non-integer solutions and thus a search plan that is not implementable.

One way of obtaining an (integer) search plan from the non-integer solution is by applying some rounding heuristic. However, we have not examined that approach to obtain integer solutions from the continuous relaxation. Algorithm 15 does provide a lower bound on the optimal value of MHSP as it solves a relaxation. We note that gradient-descent methods may also solve the relaxed MHSP. However, we have not pursued that avenue.

Tables 25 to 28 report the results of one-hour computational test for the case with 5, 10, 15 and 30 searchers, respectively. For the cases with more than 5 searchers, after one hour, Algorithm 13 provides quite good solutions with gaps at about 2%. Even for 5 searchers, the gap is moderate about 5%. Algorithm 15 (continuous relaxation) gives non-detection probability (NDP) values close to those from Algorithm 13. Especially for the cases with large number of searchers (15, 30), the obtained NDP values from Algorithms 13 and 15 are almost same.

Comparing with the lower bounds of Algorithms 13 and 15, we find that Algorithm 15 provides larger lower bounds than Algorithms 13. Of course, the upper bound from Algorithm 15 is not valid for MHSP. Algorithm 15 solves each iteration more quickly than Algorithms 13 since it solves continuous relaxation problems. Thus, Algorithm 15 generates many more cuts and brings up the lower bound quickly. Thus, if the allowed calculation time is small, a hybrid method using Algorithms 13 and 15 may be better. For example, Algorithm 15 (continuous relaxation) is initially implemented for some period to push up the lower bound quickly by generating many cuts, and later Algorithm 13 is implemented to provide a good (integer) search plan. We note the Algorithm 13 with many cuts takes much run time. Thus, this hybrid approach may be extremely inefficient when Algorithm 15 has generated too many "weak" cuts before Algorithm 13 starts. We examined this hybrid approach for several cases, but found it generally to be an inferior approach for this reason.

Time	Algo. 13: case 1			Algo. 15: case 1r		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.505386	0.554756	0.098	0.506761	0.547174	0.080
20	0.514396	0.546065	0.062	0.518175	0.538696	0.040
30	0.516137	0.546065	0.058	0.521317	0.535791	0.028
40	0.518325	0.546065	0.054	0.523267	0.534307	0.021
50	0.518325	0.546065	0.054	0.524436	0.533078	0.016
60	0.518810	0.546065	0.053	0.525279	0.532586	0.014

Table 25. Numerical results for Algorithms 13 and 15 for the case with 5 searchers for every 10 minutes of calculations.

Time	Algo. 13: case2			Algo. 15: case2r		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.501388	0.549834	0.097	0.506429	0.546805	0.080
20	0.511533	0.541721	0.059	0.518083	0.538787	0.040
30	0.518910	0.535177	0.031	0.521340	0.535598	0.027
40	0.520400	0.535177	0.028	0.523113	0.534203	0.021
50	0.521077	0.535081	0.027	0.524351	0.533240	0.017
60	0.522048	0.533785	0.022	0.525165	0.532440	0.014

Table 26. Numerical results for Algorithms 13 and 15 for the case with 10 searchers for every 10 minutes of calculations.

Time	Algo. 13: case3			Algo. 15: case3r		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.495162	0.553566	0.118	0.507577	0.546699	0.077
20	0.511244	0.540650	0.058	0.517370	0.538297	0.040
30	0.517330	0.537061	0.038	0.520760	0.536154	0.030
40	0.519869	0.535204	0.029	0.522931	0.534244	0.022
50	0.521624	0.534385	0.024	0.524083	0.533416	0.018
60	0.523123	0.533276	0.019	0.524873	0.532910	0.015

Table 27. Numerical results for Algorithms 13 and 15 for the case with 15 searchers for every 10 minutes of calculations.

Time	Algo. 13: case4			Algo. 15: case4r		
(min)	LB	UB	Gap	LB	UB	Gap
10	0.487626	0.555553	0.139	0.508092	0.542999	0.069
20	0.505637	0.542030	0.072	0.518134	0.538112	0.039
30	0.513595	0.538593	0.049	0.521319	0.535204	0.027
40	0.516808	0.537237	0.040	0.523258	0.533536	0.020
50	0.520377	0.534938	0.028	0.524341	0.533536	0.018
60	0.522040	0.533946	0.023	0.524980	0.532239	0.014

Table 28. Numerical results for Algorithms 13 and 15 for the case with 30 searchers for every 10 minutes of calculations.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE RESEARCH

A. CONCLUSIONS

This dissertation develops models and solution methodologies to solve the discrete-time path-optimization problems for single and multiple searchers looking for a non-evading moving target in a finite set of cells. We especially focus on the following problems:

- Single searcher problem (SSP) with additional resource constraints related to risk exposure to threats and fuel consumption (RSSP)
- SSP for multiple searchers (MSP)
- MSP for multiple homogeneous searchers (MHSP)

The dissertation starts by formulating RSSP, which generalizes existing models of the SSP by considering (i) history-dependent glimpse detection probability, (ii) multiple altitudes for the searcher, and (iii) multiple constraints on "consumption" of resources such as time, fuel, and risk.

We develop a specialized branch-and-bound (B/B) algorithm for solving RSSP and propose a new bound (Lagrangian directional static bound) on the optimal detection probability using network expansion to account for a portion of the history of the current path and using a Lagrangian relaxation to eliminate resource constraints. We also derive a series of network reduction procedures that tighten the Lagrangian relaxation and reduce the amount of path enumeration required by the B/B algorithm.

In direct comparison with a state-of-the-art algorithm for SSP, the proposed bound and network reduction procedures reduce the run times by at least one order of magnitude. For RSSP with time, fuel, and risk constraints, as well as two altitudes, our B/B algorithm solves problem instances with 10 by 10 cells and a time horizon of 40 typically within 20 minutes.

We develop a B/B algorithm and two heuristics (the static bound heuristic and the cross-entropy (CE) heuristic) to solve MSP. Among these algorithms, the CE heuristic performs better for a broad range of problem instances.

We also focus on MHSP and develop an exact outer approximation (OA) algorithms using several novel cutting planes (multi-cut, strong-cut, relative-cut, and symmetric-cut). In empirical studies, this algorithm provides search plans that are guaranteed to be within 5% of an optimal plan in less than about 20 minutes for problem instances involving 15 by 15 cells, 3 searchers, and a time horizon of 16. Instances with 5 searchers are solved even faster. In addition, we prove that under certain assumptions the nonlinear multiple homogenous searcher problem is equivalent to a large-scale linear mixed-integer program.

B. FUTURE RESEARCH

The CE heuristic appears to quickly generate a good feasible solution for MSP. However, it does not guarantee solution quality. A possible future area of research would be to develop a hybrid algorithm that uses the CE heuristic to generate an initial feasible solution for an OA algorithm. Currently we implement the CE heuristic using C++ and execute the OA algorithms using GAMS with the CPLEX solver. The development of an integrated computational program is worthy to effectively solve the MSP/MHSP. Furthermore, an improved implementation of the OA algorithms outside GAMS may reduce cut generation time and prove worthwhile.

This dissertation does not consider the multiple searcher problem with resource constraints (MRSP). In practical applications, the mission planner would like to find an optimal search plan for multiple "resource-constrained" searchers. A combined approach of the RSSP algorithm (Lagrangian directional static bound) with the MSP algorithm (CE heuristic) could be developed for this purpose. For example, such an approach is applicable for the case where the multiple searchers start the search operation all at the same time and finish their mission at the same time.

LIST OF REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

[2] Adam Howell Raja Sengupta Allison Ryan, Marco Zennaro and J. Karl Hedrick. An overview of emerging results in cooperative uav control. In *43rd IEEE Conference on Decision and Cotrol*, 2004.

[3] Owen Dafydd Jones Andre Costa and Dirk Kroese. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35(5):573–580, 2007.

[4] Y. Aneja, V. Aggarwal, and K. Nair. Shortest chain subject to side conditions. *Networks*, 13:295–302, 1983.

[5] J. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.

[6] John R. Birge and Francois Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 2000.

[7] S. S. Brown. Optimal search for a moving target in discrete time and space. *Operations Research*, 28(6):1275–1289, 1980.

[8] T. Caldwell. On finding minimal routes in a network with turning penalties. *Communications of the ACM*, 4:107–108, 1961.

[9] W. M. Carlyle, J. O. Royset, and R. K. Wood. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks*, to appear, 2007. http://www.nps.navy.mil/orfacpag/resumePages/papers/roysetpa.htm (last accessed 22 September 2008).

[10] W. M. Carlyle, J. O. Royset, and R. K. Wood. Routing military aircraft with a constrained shortest-path algorithm. *Military Operations Research*, in review, 2007. http://www.nps.navy.mil/orfacpag/resumePages/papers/roysetpa.htm (last accessed 22 September 2008).

[11] W. M. Carlyle and R. K. Wood. Near-shortest and k-shortest simple paths. *Networks*, 46:98–109, 2005.

[12] Krishna Chepuri and Tito Homem de Mello. Solving the vehicle routing problems with stochastic demands using the cross entropy method. *Annals of Operations Research*, 2004. submitted.

[13] R.F. Dell, J.N. Eagle, G.H.A. Martins, and A.G. Santos. Using multiple searchers in constrained-path, moving-target search problems. *Naval Research Logistics*, 43:463–480, 1996.

[14] D. DeWolfe, J. Stevens, and R. K. Wood. Setting military reenlistment bonuses. *Naval Research Logistics*, 40:143–160, 1993.

[15] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithm for the weight-constrained shortest path problem. *Networks*, 42:135–153, 2003.

[16] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.

[17] J.N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations Research*, 32(5):1107–1115, 1984.

[18] J.N. Eagle and J.R. Yee. An optimal branch and bound procedure for the constrained path, moving target search problem. *Operations Research*, 38:110–114, 1990.

[19] B. L. Fox and D. M. Landi. Searching for the multiplier in one-constraint optimization problems. *Operations Research*, 18:253–262, 1970.

[20] Tomonari Furukawa Frederic Bourgult and Hugh F. Durrant-Whyte. Coordinated decentralized search for a lost target in a bayesian world. In *Proceeding of the 2003 IEEE/RSJ Intl Conference on Intelligence and Systems, Las Vegas*, 2003.

[21] GAMS. *GAMS Distribution 22.5*. GAMS Development Corporation, Washington, DC, 2007. http://www.gams.com (last accessed 22 September 2008).

[22] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, California, 1979.

[23] Don A. Grundel. Constrained search for a moving target. In *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems*, pages 327–332, 2005.

[24] G. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.

[25] Shoudong Huang Haye Lau and Gamini Dissanayake. Optimal search for multiple targets in a built environment. In *Proceeding of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton*, 2005.

[26] G. Hollinger and S. Singh. Proofs and experiments in scalable, near-optimal search by multiple pobots. In *Robotics: Science and Systems*, 2008.

[27] ILOG. *CPLEX 10.0 Documentation*. ILOG, Inc., Mountain View, California, 2006. http://www.cplex.com (last accessed 22 September 2008).

[28] Gaemus E. Collins James R. Riehl and Joao P. Hespanha. Cooperative graph-based model predictive search. In *46th IEEE Conference on Decision and Control*, pages 2998–3004, 2007.

[29] N. J. Karczcewski. *Optimal aircraft routing in a constrained path-dependent environment*. Master's thesis, Naval Postgraduate School, Monterey, California, 2007.

[30] J. E. Kelley. The cutting plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8:703–712, 1960.

[31] M. Kress and J. O. Royset. Aerial search optimization model (ASOM) for uavs in special operations. *Military Operations Research*, 13(1):23–33, 2008.

[32] H. Lau, S. Huang, and G. Dissanayake. Discounted mean bound for the optimal searcher path problem with non-uniform travel times. *European Journal of Operational Research*, 190(2):383–397, 2008.

[33] L. Margolin. *Cross-entropy method for combinatorial optimization*. Master's thesis, Israel Institute of Technology, Haifa, Israel, 2002.

[34] G. A. Martin. *A New Branch-and-bound Procedure for Computing Optimal Search Path*. Master's thesis, Naval Postgraduate School, Monterey, California, 1993.

[35] R. Murphey, S. Uryasev, and M. Zabarankin. Optimal path planning in a threat environment. In P. Pardalos, editor, *Recent Developments in Cooperative Control and Optimization*, pages 349–406, Kluwer Academic, Dordrecht, Netherlands, 2003.

[36] Shie Mannor Pieter-Tjerk de Boer, Dirk P. Kroese and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[37] D.N. Reber. Major, Ops Analysis Division, U.S. Marine Corps Combat Development Command. Private communication November 26, 2007.

[38] Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112, 1997.

[39] Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continous optimization. *Methodology and Computing in Applied Probability*, 2:127–190, 1999.

[40] Reuven Y. Rubinstein. Cross-entropy and rare events for maximal cut and bipartition problems. *ACM Transactions on Modeling and Computer Simulation*, 12:27–53, 2002.

[41] M. G. Monticino S. J. Benkoski and J. R. Weisinger. A survey of the search theory literature. *Naval Research Logistics*, 38:469–494, 1991.

[42] Michael S. Scioletti. *A Heuristic Algorithm for Optimized Routing of Unmanned Aerial Systems for the Interdiction of Improvised Explosive Devices*. Master's thesis, Naval Postgraduate School, Monterey, California, 2008.

[43] Barry R. Secrest. *Traveling Salesman Problem for Surveillance mission Using Particle Swarm Optimization*. Master's thesis, Air University, Department of the Air Force, Wright-Patterson Air Force Base, Ohio, 2001.

[44] Reuven Y. Rubinstein Shie Mannor and Yohai Gat. The cross entropy method for fast policy search. In *The 20th International Conference on Machine Learning (ICML-2003), Washington, DC*, 2003.

[45] Nah-Oak Song and Demosthenis Teneketzis. Discrete search with multiple sensors. *Mathematical Methods of Operations Research*, 60(1):1–13, 2004.

[46] T. J. Stewart. Search for a moving target when searcher motion is restricted. *Computers & operations research*, 6(3):129–140, 1979.

[47] L. D. Stone. *Theory of Optimal Search*. Academic Press, New York, 1975.

[48] L. C. Thomas and J. N. Eagle. Criteria and approximate methods for path-constrained moving-target search problems. *Naval Research Logistics*, 42:27–38, 1995.

[49] K. E. Trummel and J.R. Weisinger. The complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327, 1986.

[50] A. R. Washburn. Search for a moving target: The fab algorithm. *Operations Research*, 31(4):739–751, 1983.

[51] A. R. Washburn. Branch and bound methods for search problems. Technical report, Naval Postgraduate School, Monterey, California, April 1995.

[52] A. R. Washburn. Branch and bound methods for a search problem. *Naval Research Logistics*, 45:243–257, 1998.

[53] E. Wong, F. Bourgault, and T. Furukawa. Multi-vehicle Bayesian search for multiple lost targets. In *Proceedings of the 2005 IEEE Internationl Conference on Robotics and Automation*, pages 3169–3174, Barcelona, Spain, 2005.

[54] Ali A. Minai Yanli Yang and Marios M. Polycarpou. Decentralized cooperative search by networks uavs in an uncertain environment. In *Proceeding of the 2004 American Control Conference Boston, Massachusetts*, 2004.

[55] M. Zabarankin, S. Uryasev, and R. Murphey. Aircraft routing under the risk of detection. *Naval Research Logistics*, 53:728–747, 2006.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Assistant Professor Johannes O. Royset
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

4. Professor R. Kevin Wood
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

5. Professor James N. Eagle
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

6. Professor Moshe Kress
   Department of Operations Research
   Naval Postgraduate School
   Monterey, California

7. Professor Isaac I. Kaminer
   Department of Mechanical and Astronautical Engineering
   Naval Postgraduate School
   Monterey, California