Route Optimization for Multiple Searchers

J.O. Royset and H. Sato
Operations Research Department, Naval Postgraduate School
Monterey, California, USA

September 4, 2009

Abstract. We consider a discrete time-and-space route-optimization problem, across a finite time horizon, in which multiple searchers seek to detect one or more probabilistically moving targets. The paper formulates a novel convex mixed-integer nonlinear program for this problem that generalizes earlier models to situations with multiple targets, searcher deconfliction, and target- and location-dependent search effectiveness. We present two solution approaches, one based on the cutting-plane method and the other on linearization. These approaches result in the first practical, exact algorithms for solving this important problem, which arises broadly in military, rescue, law enforcement, and border patrol operations. The cutting-plane approach solves many realistically sized problem instances in few minutes, while existing branch-and-bound algorithms fail. A specialized cut improves solution times by 50% in difficult problem instances. The approach based on linearization, which is applicable in important special cases, may further reduce solution times with one or two orders of magnitude. The solution times for the cutting-plane approach tend to remain constant as the number of searchers grows. In part, then, we overcome the difficulty that earlier solution methods have with many searchers.

Subject Classifications: Military operations research, search and surveillance, route planning, mixed-integer nonlinear programming.

Area of review: Military and Homeland Security.

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
04 SEP 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Route Optimization for Multiple Searchers	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Operations Research Department,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
in review

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**42**	

1 Introduction

We consider a discrete-time route-optimization problem, denoted SP, in which multiple, fully cooperating searchers and one or more probabilistically moving targets operate in an environment consisting of a finite set of cells. In the case of a single target, the searchers seek to minimize the probability of not detecting the target across a time horizon. In the presence of multiple targets, the searchers adopt a worst-case approach and aim to minimize the probability of not detecting the target with the largest nondetection probability. The searchers are subject to constraints on route continuity, may require some travel time to move from one cell to another, and may need to maintain a minimum distance between each other to avoid interference. The searchers are imperfect and may not detect a target present in a cell subject to search. However, we ignore the possibility of false positive reports of targets.

SP arises broadly in military, rescue, law enforcement, and border patrol operations. In military operations, the searchers may be manned or unmanned aircraft looking for suspected individuals or downed pilots in an area of interest. Park rangers may search for lost hikers. In a damaged or burning building, fire fighters and ground robots may search for trapped individuals. Law enforcement officers may act as searchers when looking for criminals. Near national borders, the searchers are border patrols seeking illegal immigrants. The searchers may also be Coast Guard cutters and helicopters scanning the ocean for ship wrecks and smugglers. We adopt the traditional terminology (see, e.g., [33, 31]) and refer to the objects being looked for as "targets," even though the searchers may not wish those objects any harm. Often in these applications, the targets are unaware of the searchers' routes or are unable to take advantage of such information. Hence, it is realistic to assume that the targets move between cells according to some known probability distribution that is independent of the searchers' routes; see Section 2 for details. This assumption is also found in, e.g., [29, 9, 33, 17] and we adopt it throughout this paper. The possibility of a target intelligently adapting to the searchers' routes is beyond the scope of the paper.

The situations described above may involve just a few searchers. However, with the technological improvement of autonomous systems and sensors, planners now frequently need to optimally route many searchers. (We refer to [25] for an overview of recent advances in the area of autonomous aerial systems and [20] for an introduction to the U.S. Department of Defense's effort to utilize a large number of autonomous systems for reconnaissance, information gathering, and search and rescue operations.) Effective manual planning for a few human-controlled searchers is difficulty (see for example p.7-7 in [33]) and, of course, even more difficult for a large number of autonomous systems. In fact, [20], p. 164, states that

> "Many recognize the potential of robotics/unmanned systems in military operations; yet few understand how to employ them effectively. Computer generated modeling and simulation tools are extremely valuable in an attempt to generate both operational concepts for tactics, techniques and procedures (TTP) as well as technical requirements to enable those procedures."

Solutions of SP recommend effective routes for the searchers that help planners to better utilize their assets, provide input to control units of autonomous systems, and possibly save lives, for example, in search and rescue operations. The U.S Coast Guard's tactical decision aid CASP and the U.S. Navy's NODESTAR both utilize solutions of special instances of SP obtained by heuristics (see Chapter 7 in [33]).

Previous studies have focused on single-searcher and single-target instances of SP; see [26] for a comprehensive review of papers up to 1991; [17, 28], and references therein, cover more recent work. The most efficient exact algorithms for such instances of SP appear to be specialized branch-and-bound algorithms. These algorithms obtain bounds by replacing, in effect, the probability of detection with the expected number of detections [7, 32, 17, 28].

In principle, such specialized branch-and-bound algorithms are also applicable to instances of SP with multiple searchers [7, 27]. However, when applied to realistically sized instances with more than two searchers, they tend to require excess computer memory and prohibitive solution times [27]. This has motivated the development of heuristics for SP,

and related problems, such as local search and genetic algorithms [7] as well as the cross-entropy method [27]. Other heuristic algorithms rely on approximations of probabilities by expectations [7], myopic optimization with a receding-horizon [7, 12, 35, 23], sequential optimization of each searcher [29, 35, 13], and decentralized optimization by each searcher [3, 36, 35]. Of these studies, only [36, 35] deal with multiple targets, and then, only in the context of heuristics. Hence, there is a need to model multi-searcher instances of SP and to provide exact algorithms for their solution.

This paper formulates a novel convex mixed-integer nonlinear program (MINLP) for SP. The program generalizes the formulation in [29] to multiple targets. It also appears to be the first to prevent interference between searchers (i.e., to "deconflict" searchers) in this context and to account for the fact that a searcher's effectiveness may depend on the target, the time of search, and the searcher's previous location. The latter generalization provides an important practical advantage as it reduces the need for fine time discretizations when modeling real-world search missions.

We present two solution approaches to MINLP resulting in the first practical, exact algorithms for SP with multiple searchers. One approach based on the cutting-plane method (see, e.g., [15, 8, 34]), leads to fast solutions for many realistically sized problem instances for which existing branch-and-bound algorithms fail. For difficult instances of SP, we improve solution times further with a specialized cut. The other approach is based on novel, equivalent linearizations of MINLP available in important special cases. Optimal solutions of the linearizations are often easily obtained by standard mixed-integer linear programming solvers leading to further reduction in solution times by one to two orders of magnitude.

Poor scalability of algorithms as the number of searchers grows is of major concern when solving SP and related problems and has led to the development of heuristic algorithms (see, e.g., [35, 13, 27]). The cutting-plane approach scales well as the solution times tend to remain constant as the number of searchers grows. We therefore overcome, in part, the difficulty of solving problem instances with many searchers.

The remainder of the paper is outlined as follows. The next section defines SP precisely.

Section 3 derives mathematical programming formulations of SP and its linearizations. In Section 4, we present algorithms and numerical results, with particular focus on an important special case involving a single target.

2 Problem Statement

2.1 Searchers, Targets, and their Environment

We let the area of interest (AOI) be discretized into a finite set of cells $\mathcal{C} = \{1, \ldots, C\}$, and let the time horizon be discretized into a finite set of time periods $\mathcal{T}_0 = \{0\} \cup \mathcal{T}$, where $\mathcal{T} = \{1, 2, ..., T\}$. Search for targets takes place during time periods $t \in \mathcal{T}$, with $t = 0$ representing the time period prior to start of the search. There are K independent targets present in the AOI with each target $k \in \mathcal{K} = \{1, 2, ..., K\}$ occupying one cell in each time period. The quantity $\omega_{k,t} \in \mathcal{C}$ denotes the (random) cell that target k occupies during time period $t \in \mathcal{T}$. The vector of cells $\omega_k = (\omega_{k,1}, \omega_{k,2}, \ldots, \omega_{k,T})$ denotes a possible path for target k, and $q_k(\omega_k)$ denotes the given probability that target k takes that path. The set Ω_k denotes the collection of all possible paths for target k with positive probability $q_k(\omega_k)$. In practice, the data Ω_k and $q_k(\omega_k)$ are generated using Monte Carlo sampling from (complex) target motion models (as in U.S. Coast Guard's decision aid CASP) or defined implicitly by Markov transition matrices (as in the case of U.S. Navy's decision aid NODESTAR); see Chapter 7 in [33]. This paper considers both ways of specifying target movement and refers to the former way as a *conditional target model* and to the latter way as a *Markovian target model*. In Sections 3 and 4, we see that it is not necessary to explicitly enumerate all possible target paths in the case of a Markovian target model.

It is trivial to extend the current framework to situations with targets that may not be present in the AOI and targets that enter and leave the AOI during the time horizon. However, we do not examine that situation further.

There are L classes of searchers, with each class $l \in \mathcal{L} = \{1, 2, ..., L\}$ containing J_l identical searchers. During each time period $t \in \mathcal{T}_0$, each searcher occupies a cell or is in transit between cells. When occupying a cell c, a searcher of class l may select to move to

any cell "adjacent" to c as defined by the forward star $\mathcal{F}_l(c) \subset \mathcal{C}$. We also let $\mathcal{R}_l(c) \subset \mathcal{C}$ denote the reverse star of cell c, which represents the set of cells from which a searcher of class l can reach cell c in one move. By convention, $c \in \mathcal{F}_l(c)$ and $c \in \mathcal{R}_l(c)$. A searcher of class l requires $d_{l,c,c'}$ time periods to move from cell c to cell $c' \in \mathcal{F}_l(c)$ and to search cell c' for one time period. Since the time to search the "destination" cell c' is included in $d_{l,c,c'}$, $d_{l,c,c'} \geq 1$ for all l, c, c' and $d_{l,c,c'} = 1$ only if the time to move from c to c' is zero.

Searchers may interfere with each other and could be required to maintain a minimum internal distance. We let n_c be the maximum number of searchers allowed to occupy cell c during any one time period $t \in \mathcal{T}$. Moreover, for each possible move between to cells for a searcher, we define a corresponding set of incompatible moves between cells that would cause interference if carried out by another searcher. Specifically, if a searcher of class l moves from cell c to cell c' starting in time period t, then the set $\mathcal{D}(l, c, c', t)$ gives all quadruples of searcher classes, cell pairs, and time periods that are incompatible with that searcher's move. We refer to these restrictions as deconfliction constraints.

We let $X_{l,c,c',t}$ denote the number of searchers of class l that occupy cell c in time period $t \in \mathcal{T}_0$ and that move to cell c' next, and let X denote the vector with components $X_{l,c,c',t}$, $l \in \mathcal{L}$, $c, c' \in \mathcal{C}$, and $t \in \mathcal{T}_0$. We refer to X as a search plan.

2.2 Sensor Model

We assume that each searcher is equipped with one imperfect sensor. Each time period $t \in \mathcal{T}$ in which a searcher occupies a cell, the searcher's sensor takes one "look" in the cell for each target. When a searcher is in transit between cells, the sensor is inactive. The probability that one look for a target in a cell detects the target, given that the target currently occupies the cell, may depend on the searcher class (is it a high- or low-quality searcher?), the target's characteristic (is it shiny or camouflaged?), the cell (is it forested or open?) and time of day (is it bright mid-day or dark midnight?). Specifically, if target k and a searcher of class l occupy cell c in time period t and c' is the searcher's previous cell, then the probability that the searcher's look during time period t detects the target is $g_{l,c',c,t,k} \in (0, 1)$. We refer to

this probability as the *glimpse-detection probability*.

We note that the glimpse-detection probability depends on the searcher's previous location. This dependence may arise if adjusting search pattern and/or altitude, refocusing a sensor, and becoming familiar with a new cell have a significant detrimental effect on the searcher's capability to detect a target. In addition, this dependance allows us to account indirectly for small transit times (much less than the length of a time period) between cells by reducing the glimpse-detection probability from its nominal value if the searcher just moved into a cell. For example, suppose that the real-world travel time from cell c' to c is one minute. To model this situation (approximately), we would normally require a time period of (approximately) one-minute duration. However, this may result in a large number of time periods and long computing times. Alternatively, we can define a longer time period, say 10 minutes, and let the glimpse-detection probability in cell c be somewhat reduced if a searcher's previous cell were c' as compared to if it were c. This will reflect the fact that a searcher coming from cell c' has only nine minutes to search c compared to 10 minutes if the searcher had already been present in c. Hence, we avoid adopting a fine time discretization with resulting high computational cost.

We assume that all the searchers' looks are independent. Hence, given search plan X, the probability that no searcher detects target k in cell c in time period t, given that target k occupies cell c at that time, equals

$$\prod_{l\in\mathcal{L}}\prod_{c'\in\mathcal{R}_l(c)}\left(1-g_{l,c',c,t,k}\right)^{X_{l,c',c,t-d_{l,c',c}}} \tag{1}$$

$$= \exp\left(-\sum_{l\in\mathcal{L}}\sum_{c'\in\mathcal{R}_l(c)}\alpha_{l,c',c,t,k}X_{l,c',c,t-d_{l,c',c}}\right), \tag{2}$$

where for all $l\in\mathcal{L}$, $c\in\mathcal{C}$, $c'\in\mathcal{R}_l(c)$, $t\in\mathcal{T}$, and $k\in\mathcal{K}$,

$$\alpha_{l,c',c,t,k} = -\ln(1-g_{l,c',c,t,k}) \tag{3}$$

is the *detection rate*. While the glimpse-detection probability may be difficult to estimate directly without extensive field testing, the detection rate can often be related to a searcher's speed and sensor range, the size of the cell, and the length of the time period; for example

see p. 2-1 in [33].

SP seeks to minimize, by choice of a search plan X, the probability of not detecting the target with the largest nondetection probability during the time horizon. The choice of search plan is subject to the route constraints induced by the forward and reverse stars $\mathcal{F}_l(c)$ and $\mathcal{R}_l(c)$, deconfliction constraints given by n_c and $\mathcal{D}(l, c, c', t)$, and the given initial condition that $x_{l,c,0}$ searchers of class l occupy cell c in time period 0.

3 Models of Search Problem

In this section, we formulate SP, in complete generality, as a convex MINLP. Since the model is nonlinear, we anticipate relatively long solution times for standard MINLP solvers (e.g., Bonmin [6], DICOPT [11]). Hence, a main focus of this paper is to develop solution approaches that utilize structure present in important classes of problem instances. Consequently, we go on to construct two *linear* models for classes of problem instances involving homogenous searchers and a single target.

This section first states the nonlinear model of SP and second deals with the linearizations for special classes. The section ends with a discussion of linearization of the full nonlinear model.

3.1 Nonlinear Model of SP

We state SP as a convex MINLP, generalizing the formulation in [29] to account for multiple targets and deconfliction constraints as well as glimpse-detection probabilities that may depend on the target, the time of search, and a searcher's previous location. The resulting program takes the following form.

Model SPX:

Indices

c, c', c'', c'''	cells ($c, c', c'', c''' \in \mathcal{C} = \{1, \ldots, C\}$).
t, t'	time periods ($t, t' \in \mathcal{T}_0 = \{0\} \cup \mathcal{T}$, $\mathcal{T} = \{1, ..., T\}$).
l, l'	searcher class ($l, l' \in \mathcal{L} = \{1, ..., L\}$).
k	target ($k \in \mathcal{K} = \{1, ..., K\}$).
ω_k	path of target k ($\omega_k \in \Omega_k$).

Sets

$\mathcal{F}_l(c) \subseteq \mathcal{C}$	forward star of cell c for searcher of class l.
$\mathcal{R}_l(c) \subseteq \mathcal{C}$	reverse star of cell c for searcher of class l.
$\mathcal{D}(l, c, c', t)$	set of quadruples (l', c'', c''', t') incompatible with a searcher of class l that moves from c to c' starting in time period t.

Parameters

$\alpha_{l,c',c,t,k}$	detection rate in cell c in time period t against target k for a searcher of class l when the searcher previously occupied c'.
$\zeta_{c,t}(\omega_k)$	1 if cell c is on target path ω_k in time period t, otherwise 0.
$x_{l,c,0}$	number of searchers of class l that occupy cell c in time period 0.
J_l	number of searchers of class l.
$q_k(\omega_k)$	probability that target k takes path ω_k.
$d_{l,c,c'}$	number of time periods needed for a searcher of class l to move directly from cell c to cell c' and search c'.
n_c	maximum number of searchers that occupy cell c in a time period.

Decision Variables

$X_{l,c,c',t}$	number of searchers of class l that occupy cell c in time period t and that move to cell c' next. (X denotes the vector with components $X_{l,c,c',t}$, $l \in \mathcal{L}, c, c' \in \mathcal{C}, t \in \mathcal{T}_0$.)
$Y_{c,t,k}$	auxiliary variable representing "search effort" in cell c against target k in time period t. (Y_k denotes the vector with components $Y_{c,t,k}$, $c \in \mathcal{C}, t \in \mathcal{T}$.)

Functions

$f_k(Y_k)$	nondetection probability of target k

$$= \sum_{\omega_k \in \Omega_k} q_k(\omega_k) \exp\left(- \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega_k) Y_{c,t,k} \right). \tag{4}$$

Formulation

$$\min_{X, Y_k, k \in \mathcal{K}} \max_k f_k(Y_k) \tag{5}$$

$$\text{s.t.} \sum_{c' \in \mathcal{R}_l(c)} X_{l,c',c,t-d_{l,c',c}} = \sum_{c' \in \mathcal{F}_l(c)} X_{l,c,c',t} \qquad \forall\ l, c, t \in \mathcal{T} \tag{6}$$

$$\sum_{c' \in \mathcal{F}_l(c)} X_{l,c,c',0} = x_{l,c,0} \qquad \forall\ l, c \tag{7}$$

$$\sum_{l \in \mathcal{L}} \sum_{c' \in \mathcal{R}_l(c)} \alpha_{l,c',c,t,k} X_{l,c',c,t-d_{l,c',c}} = Y_{c,t,k} \qquad \forall\ c, t \in \mathcal{T}, k \tag{8}$$

$$\sum_{l \in \mathcal{L}} \sum_{c' \in \mathcal{R}_l(c)} X_{l,c',c,t-d_{l,c',c}} \leq n_c \qquad \forall\ c, t \in \mathcal{T} \tag{9}$$

$$X_{l,c,c',t} + X_{l',c'',c''',t'} \leq 1 \qquad \forall\ l, c, c' \in \mathcal{F}_l(c), t, \tag{10}$$
$$\forall\ (l', c'', c''', t') \in \mathcal{D}(l, c, c', t)$$

$$X_{l,c,c',t} \in \{0, 1, 2, ..., J_l\} \qquad \forall\ l, c, c', t \tag{11}$$

$$Y_{c,t,k} \geq 0 \qquad \forall\ c, t \in \mathcal{T}, k. \tag{12}$$

The decision variable $Y_{c,t,k}$ could be eliminated by substitution using (8), but is included for notational simplicity. We obtain the nondetection probability for target k in (4) from (2) by application of the total probability theorem and the fact that detection in cell c in time period t can occur only if the target occupies that cell at that time. The objective function (5) aims to minimize the largest nondetection probability. The objective function of **SPX** is convex and the nondetection probabilities $f_k(Y_k), k \in \mathcal{K}$, are convex and continuously differentiable. In problem instances with a large number of possible target paths, calculation of $f_k(Y_k)$ using (4) is expensive. However, a Markovian target model permits a more efficient way of computing $f_k(Y_k)$, which we present and exploit in Sections 4.2 and 4.3. Consequently, **SPX** is applicable for both conditional and Markovian target models.

Constraints (6) and (7) ensure route continuity and define initial conditions for the searchers, respectively. Deconfliction constraints (9) and (10) limit the number of searchers that can occupy cell c to at most n_c in any time period $t \in \mathcal{T}$ and exclude moves in conflict with each other, respectively. We observe that **SPX** prescribes the "best" search plan prior to detection of the first target. It is beyond the scope of the paper to plan for events after the first detection. We discuss the solution of **SPX** in Sections 3.3 and 4.3, but first deal with important special cases.

3.2 Linearizations of SPX for Homogeneous Searchers and Single Target

We now consider special classes of instances of **SPX** involving homogeneous searchers and a single target, which allow us to construct two equivalent *linear* models. For simplicity we also exclude deconfliction constraints. We discuss the merits of extending these linearizations to the general **SPX** in Section 3.3. Specifically, **SPX** with one searcher class (i.e., $L = 1$), one target (i.e., $K = 1$), a constant detection rate over all cells and time periods, and no

deconfliction constraints takes the following form. In this case, we drop the subscripts k and l.

Model SP1:

Indices

c, c'	cells ($c, c' \in \mathcal{C} = \{1, \ldots, C\}$).
t	time periods ($t \in \mathcal{T}_0 = \{0\} \cup \mathcal{T}$, $\mathcal{T} = \{1, ..., T\}$).
ω	path of target ($\omega \in \Omega$).

Sets

$\mathcal{F}(c) \subseteq \mathcal{C}$	forward star of cell c for a searcher.
$\mathcal{R}(c) \subseteq \mathcal{C}$	reverse star of cell c for a searcher.

Parameters

α	detection rate for a searcher.
$\zeta_{c,t}(\omega)$	1 if cell c is on target path ω in time period t, otherwise 0.
$x_{c,0}$	number of searchers that occupy cell c in time period 0.
J	number of searchers.
$q(\omega)$	probability that target takes path ω.
$d_{c,c'}$	number of time periods needed for a searcher to move directly from cell c to cell c' and search c'.

Decision Variables

$X_{c,c',t}$	number of searchers that occupy cell c in time period t and that move to cell c' next. (X denotes the vector with components $X_{c,c',t}, c, c' \in \mathcal{C}, t \in \mathcal{T}_0$.)
$Z_{c,t}$	number of searchers that occupy cell c in time period t. (Z denotes the vector with components $Z_{c,t}, c \in \mathcal{C}, t \in \mathcal{T}$.)

Functions

$f(Z)$ nondetection probability of target

$$= \sum_{\omega \in \Omega} q(\omega) \exp\left(- \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega) \alpha Z_{c,t} \right). \tag{13}$$

Formulation

$$\min_{X,Z} f(Z) \tag{14}$$

$$\text{s.t.} \sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c' \in \mathcal{F}(c)} X_{c,c',t} \qquad \forall\ c, t \in \mathcal{T} \tag{15}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',0} = x_{c,0} \qquad \forall\ c \tag{16}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \qquad \forall\ c, t \in \mathcal{T} \tag{17}$$

$$X_{c,c',t} \geq 0 \qquad \forall\ c, c', t \tag{18}$$

$$Z_{c,t} \in \{0, 1, 2, ..., J\} \qquad \forall\ c, t \in \mathcal{T} \tag{19}$$

The objective function and constraints in **SP1** are specializations of those in **SPX** with minor changes caused by the new auxiliary integer variable $Z_{c,t}$ which effectively equals $Y_{c,t,k}$ divided by α. Hence, the objective function now involves $\alpha Z_{c,t}$ instead of $Y_{c,t,k}$. We utilize the integrality of $Z_{c,t}$ in the following linearization as well as in construction of cutting planes in Section 4.1. We refer to Z as a search plan.

We next derive two linearizations of **SP1**. The first linearization is applicable in the case of a conditional target model with a moderate number of possible target paths. The second linearization is limited to the situation with a Markovian target model. We deal with the two linearizations in turn.

The objective function in **SP1** is a finite sum of exponential functions over all possible target paths; see (13). Each exponential function has as argument an integer multiple of α between 0 and JT, where J is the number of searchers. Hence, the objective function in **SP1** can equivalently be represented by piecewise linear functions. This observation leads to the first linearization of **SP1**.

Model SP1-L:

Indices
As in **SP1**.
i number of looks on a target path ($i = 0, 1, ..., JT$).

Sets and Parameters
As in **SP1**.

Decision Variables
As in **SP1**.
U_ω auxiliary variable representing nondetection probability given target path ω.

Formulation

$$\min \sum_{\omega \in \Omega} q(\omega) U_\omega \tag{20}$$

$$\text{s.t. } e^{-i\alpha}(1 + i - ie^{-\alpha}) + \frac{1}{\alpha} e^{-i\alpha}(e^{-\alpha} - 1) \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega) \alpha Z_{c,t} \leq U_\omega \qquad \forall\ \omega, i \tag{21}$$

$$(15) - (19)$$

SP1-L is a mixed-integer linear program. Constraints (21) ensure that the optimal solution results in a value of U_ω that is exactly the conditional nondetection probability given that the target follows path ω. The other constraints are identical to those in **SP1**. The number of constraints and variables in **SP1-L** grows linearly in the number of possible target paths and, hence, the formulation becomes difficult to solve for large number of such paths. This motivates a second linearization of **SP1**.

The second linearization of **SP1**, denoted **SP1-LM**, assumes a Markovian target model where the target at time $t \in \mathcal{T}$ moves according to a transition probability matrix Γ_t with elements $\gamma_{c,c',t}$, $c, c' \in \mathcal{C}$. Specifically, $\gamma_{c,c',t}$ is the probability that a target occupying cell c in time period t occupies cell c' in time period $t+1$. As we see below, it is not necessary to enumerate all possible target paths in the case of a Markovian target model.

We derive **SP1-LM** from **SP1** by introducing an "information state" $P_{c,t}$ which equals the probability that the target occupies cell c in time period t and that the target has not been detected prior to t. Given this information state and a search plan with j searchers occupying cell c in time period t, the probability of detection in cell c in time period t and no prior detection, is simply

$$P_{c,t}(1 - e^{-j\alpha_{c,t}}), \tag{22}$$

where $\alpha_{c,t}$ is the detection rate of each searcher in cell c in time period t.

Suppose that a search plan is described by the binary variables $V_{c,t,j}$, which is 1 if j searchers occupy cell c in time period t and is 0 otherwise. Then, the probability of detection over the full time horizon becomes

$$\sum_{t\in\mathcal{T}} \sum_{c} P_{c,t}(1 - e^{-\alpha_{c,t}\sum_{j=1}^{J} jV_{c,t,j}}). \tag{23}$$

The information state $P_{c,t}$ depends on the search plan as follows. Clearly, $P_{c,1} = p_c$, the probability that the target occupies cell c initially. Moreover, it follows from the definition of $P_{c,t}$ and the assumption of a Markovian target model that

$$P_{c,t+1} = \sum_{c'} \gamma_{c',c,t} P_{c',t} e^{-\alpha_{c',t}\sum_{j=1}^{J} jV_{c',t,j}} \tag{24}$$

for all $c \in \mathcal{C}$ and $t = 1, 2, ..., T-1$. While the expressions (23) and (24) are nonlinear, they can be linearized as shown in the following formulation.

Model SP1-LM:

Indices

c, c'	cells ($c, c' \in \mathcal{C} = \{1, \ldots, C\}$).
t	time periods ($t \in \mathcal{T}_0 = \{0\} \cup \mathcal{T}$, $\mathcal{T} = \{1, ..., T\}$).
j	number of searchers in a cell ($j \in \mathcal{J} = \{1, \ldots, J\}$).

Sets

As in **SP1**.

Parameters

$\alpha_{c,t}$	detection rate in cell c in time period t for any searcher.
$x_{c,0}$	number of searchers that occupy cell c in time period 0.
$d_{c,c'}$	number of time periods needed for a searcher to move directly from cell c to cell c' and search c'.
$\gamma_{c,c',t}$	probability that a target that occupies cell c in time period t occupies cell c' in time period $t+1$.
p_c	probability that the target occupies cell c in time period 1.
$q_{c,t}$	probability that the target occupies cell c in time period t, i.e., $q_{c,t} = \sum_{c'} q_{c',t-1}\gamma_{c',c,t-1}$, $t = 2, 3, ..., T$, $q_{c,1} = p_c$.

Decision Variables

$P_{c,t}$	probability that the target occupies cell c in time period t and target not detected prior to t.
$Q_{c,t,j}$	auxiliary variable that equals $P_{c,t}(1 - e^{-j\alpha_{c,t}})$ if $V_{c,t,j} = 1$ and otherwise 0.
$V_{c,t,j}$	1 if there are j searchers that occupy cell c in time period t and otherwise 0.
$W_{c,t}$	auxiliary variable that equals $P_{c,t}e^{-j\alpha_{c,t}}$ if $V_{c,t,j} = 1$ and otherwise $P_{c,t}$.
$X_{c,c',t}$	number of searchers that occupy cell c in time period t and that move to cell c' next.

Formulation

$$\min \; 1 - \sum_{t \in \mathcal{T}} \sum_{c} \sum_{j} Q_{c,t,j} \tag{25}$$

$$\text{s.t. } Q_{c,t,j} \leq q_{c,t}(1 - e^{-j\alpha_{c,t}})V_{c,t,j} \qquad \forall\, c, t \in \mathcal{T}, j \tag{26}$$

$$Q_{c,t,j} \leq (1 - e^{-j\alpha_{c,t}})P_{c,t} \qquad \forall\, c, t \in \mathcal{T}, j \tag{27}$$

$$P_{c,t+1} = \sum_{c'} \gamma_{c',c,t} W_{c',t} \qquad \forall\, c, t = 1, 2, ..., T-1 \tag{28}$$

$$W_{c,t} \leq P_{c,t} \qquad \forall\, c, t \in \mathcal{T} \tag{29}$$

$$W_{c,t} \leq e^{-j\alpha_{c,t}} P_{c,t} + q_{c,t}(1 - e^{-j\alpha_{c,t}})(1 - V_{c,t,j}) \quad \forall\, c,t \in \mathcal{T}, j \tag{30}$$

$$P_{c,1} = p_c \quad \forall\, c \tag{31}$$

$$P_{c,t} \leq q_{c,t} \quad \forall\, c,t \tag{32}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_j j V_{c,t,j} \quad \forall\, c,t \in \mathcal{T} \tag{33}$$

$$\sum_j V_{c,t,j} \leq 1 \quad \forall\, c,t \in \mathcal{T} \tag{34}$$

$$(15), (16)$$

$$P_{c,t} \geq 0 \quad \forall\, c,t \in \mathcal{T} \tag{35}$$

$$Q_{c,t,j} \geq 0 \quad \forall\, c,t \in \mathcal{T}, j \tag{36}$$

$$V_{c,t,j} \in \{0,1\} \quad \forall\, c,t \in \mathcal{T}, j \tag{37}$$

$$W_{c,t} \geq 0 \quad \forall\, c,t \in \mathcal{T} \tag{38}$$

$$X_{c,c',t} \geq 0 \quad \forall\, c,c',t \tag{39}$$

The objective function (25) in **SP1-LM** gives the probability of nondetection; its correctness follows from (23). However, since (23) is nonlinear, we linearize it using the auxiliary variable $Q_{c,t,j}$, which equals $P_{c,t}(1-e^{-j\alpha_{c,t}})$ if $V_{c,t,j} = 1$ and equals 0 otherwise. This linearization is accomplished using constraints (26) and (27). This is a "big-M" type of formulation (see, for example, [22], pp. 642-643) where, in theory, any constant at least as large as $P_{c,t}$ would suffice in front of $(1-e^{-j\alpha_{c,t}})$ in (26). Recall that $P_{c,t}$ is the probability that the target occupies cell c in time period t *and* target not detected prior to t. Moreover, recall that $q_{c,t}$ is the probability that the target occupies cell c in time period t. Hence, $q_{c,t} \geq P_{c,t}$ for all c,t. Consequently, we set the "big-M" in (26) to $q_{c,t}$. We also use $q_{c,t}$ to bound the range of $P_{c,t}$ in (32).

The evolution of the information state is also nonlinear; see (24). In **SP1-LM**, we linearize that expression by means of the auxiliary variable $W_{c,t}$ and constraints (28)-(30). Note that $W_{c,t}$ equals $P_{c,t}e^{-j\alpha_{c,t}}$ if $V_{c,t,j} = 1$ and equals $P_{c,t}$ otherwise. The initial target location is accounted for in (31). Constraints (33) and (34) relate $X_{c,c',t}$ to the binary variable $V_{c,t,j}$.

3.3 Linearizations of the General SPX

SP1-L, which linearizes **SP1** in the case of a conditional target model, generalizes easily to a linear model equivalent to **SPX** if all detection rates $\alpha_{l,c',c,t,k}$ in **SPX** are rational numbers. In that case, all detection rates **SPX** can be expressed as an integer multiple of a number, say, α. Hence, $Y_{c,t,k}$ could be expressed as α times an auxiliary integer variable. Similar to the approach leading to **SP1-L**, the exponential terms in $f_k(Y_k)$ could then be expressed by piecewise-linear functions. Standard techniques for linearizing piecewise-linear functions would then lead to a mixed-integer linear program. If $\alpha_{l,c',c,t,k}$ differs substantially across different elements of $\mathcal{L}$, $\mathcal{C}$, $\mathcal{T}$, and $\mathcal{K}$, α would need to be relatively small. Hence, the piecewise-linear functions may involve a large number of pieces and the resulting mixed-integer program may be large.

Under the same assumption on the detection rates and given a Markovian target model, **SP1-LM** generalizes to a linear model equivalent to **SPX** through a redefinition of j. While j gives the number of searchers occupying a cell during one time period in **SP1-LM**, the new linear model would require j to represent α^{total}/α, where α^{total} denotes the sum of the detection rates of all searchers occupying a cell in a time period. This sum may be larger than the number of searchers occupying the cell as each searcher would have a detection rate of $\xi\alpha$, where ξ is a positive integer. Since this linearization effectively assigns a binary variable to each possible value of the "search effort" in a cell, the resulting mixed-integer linear program may become large.

In view of the above discussion, we see that linearizations of **SPX** tend to be of reasonable size and practical value when all detection rates $\alpha_{l,c',c,t,k}$ can be expressed as small integer multiples of α. For example, this is the case when all detection rates equals $1 \cdot \alpha$ for some $\alpha > 0$ as in **SP1**. In the next section, we focus on **SP1-L** and **SP1-LM** for this important class of instances, but also consider the solution of the full **SPX** by means of a cutting-plane algorithm.

4 Algorithms and Numerical Results

This section discusses the solution of **SPX**, with particular focus on **SP1** as linearizations appear especially attractive in that case. In addition to linearizations, we also consider the cutting-plane method (see, e.g., [15, 8, 34]) as applied to **SPX** and **SP1**, and present a specialized, strengthened cutting plane for **SP1**. First, we consider **SP1** in the case of a conditional target model with a moderate number of possible target paths. In practice, this situation occurs when the paths are generated by Monte Carlo sampling from a (complex) motion model. The U.S. Coast Guard's decision aid CASP generates target paths in this manner. Second, we examine **SP1** in the case of a Markovian target model. The Markovian assumption is adopted in practice, for example by the U.S. Navy's decision aid NODESTAR, as well as in the literature [32, 17]. We refer to Chapter 7 in [33] for further details about target motion models. Third, we solve **SPX** using a cutting-plane algorithm for the case with a Markovian target model. The algorithm is also applicable in the case of a conditional target model, but we do not examine that situation in detail.

4.1 Solution of SP1 and SP1-L for a Conditional Target Model

This section develops a specialized cutting plane for **SP1** and compares the performance of the resulting cutting-plane algorithms with those corresponding to the solution of **SP1** and **SP1-L** by standard solvers.

The standard cutting-plane algorithm for convex (mixed-integer) programs sequentially builds and minimizes successively better piecewise-linear approximations of a convex function; see, e.g., [15, 8, 34] and the recent review [2]. For completeness and ease of reference, we next state our implementation of the standard cutting-plane algorithm. Below, we need the partial derivatives

$$\frac{\partial f(Z)}{\partial Z_{c,t}} = -\alpha \sum_{\omega \in \Omega} q(\omega) \zeta_{c,t}(\omega) \exp\left(- \sum_{c',t' \in \mathcal{T}} \zeta_{c',t'}(\omega) \alpha Z_{c',t'} \right). \tag{40}$$

Algorithm 1 (Obtains near-optimal solutions of SP1)

Data. Relative optimality tolerances $\delta, \delta_i \geq 0, i = 0, 1, 2, ...$

Step 0. Set the lower bound, $\underline{\xi}$, on the optimal value of **SP1** to 0; set the upper bound, $\overline{\xi}$, on the optimal value of **SP1** to 1; and set $i = 1$ and $Z^1 = 0$.

Step 1. Calculate $f(Z^i)$ and $\nabla f(Z^i)$. If $f(Z^i) < \overline{\xi}$, then set $\overline{\xi} = f(Z^i)$.

Step 2. If $\overline{\xi} - \underline{\xi} \leq \delta\underline{\xi}$, then stop.

Step 3. Solve

$$\mathbf{P}_i : \qquad \min \xi \tag{41}$$

$$\text{s.t. } f(Z^j) + \nabla f(Z^j)'(Z - Z^j) \leq \xi \qquad \forall\ j = 1, 2, ..., i \tag{42}$$

$$(15) - (19)$$

to near optimality. That is, determine a lower bound $\underline{\xi}^{i+1}$ and a feasible solution $(\overline{\xi}^{i+1}, Z^{i+1}, X^{i+1})$ of $\mathbf{P}_i$ such that $\overline{\xi}^{i+1} - \underline{\xi}^{i+1} \leq \delta_i \underline{\xi}^{i+1}$.

Step 4. If $\underline{\xi}^{i+1} > \underline{\xi}$, then set $\underline{\xi} = \underline{\xi}^{i+1}$.

Step 5. If $\overline{\xi} - \underline{\xi} \leq \delta\underline{\xi}$, then stop. Else, replace i by $i + 1$, and go to Step 1.

We note that Algorithm 1 is guaranteed to solve **SP1** to optimality if $\delta = 0$ and $\delta_i = 0$ for all i. Fixing $\delta_i > 0$ (i.e., accepting near-optimal solutions of $\mathbf{P}_i$) does not guarantee convergence, but does improve computational speed. To balance convergence and speed, we allow for small positive δ_i and adopt the simple safeguard of replacing δ_i by $\delta_i/2$ if a previous solution is repeated. This approach prevents Algorithm 1 from jamming at a nonoptimal solution.

The cutting plane (42) can be strengthened by taking advantage of the special structure of $f(Z)$ and the fact that $Z_{c,t}$ is integer. The strengthened cut uses finite differences of the objective function $f(Z)$ by considering the perturbation from $Z_{c,t}$ to $Z_{c,t} + 1$ while keeping all other variables fixed. Theorem 1 formalizes this discussion, using $\Delta_{c,t}$ to denote a CT-dimensional binary vector in which component in position (c, t) is 1 and the other components are all 0.

Theorem 1 *For any CT-dimensional nonnegative integer vectors Z and $\hat{Z}$,*

$$f(\hat{Z}) + \sum_{c,t\in\mathcal{T}} (f(\hat{Z} + \Delta_{c,t}) - f(\hat{Z}))(Z_{c,t} - \hat{Z}_{c,t}) \leq f(Z). \tag{43}$$

Proof. Let a^ω be a CT-dimensional vector defined by components $\zeta_{c,t}(\omega)\alpha$ and $b^\omega = -\ln q(\omega)$. Then, $a^\omega \geq 0$ and $b^\omega \geq 0$. Hence, $f(Z) = \sum_\omega f_\omega(Z)$, where $f_\omega(Z) = \exp(-a^\omega Z - b^\omega)$, and the result holds if $f_\omega(\hat{Z}) + \sum_{c,t\in\mathcal{T}}(f_\omega(\hat{Z} + \triangle_{c,t}) - f_\omega(\hat{Z}))(Z_{c,t} - \hat{Z}_{c,t}) \leq f_\omega(Z)$ for all ω. Consequently, we need to show that $f_\omega(\hat{Z})[1 + \sum_{c,t\in\mathcal{T}}(\exp(-\alpha^\omega_{c,t}) - 1)(Z_{c,t} - \hat{Z}_{c,t}) - \exp(-a^\omega(Z - \hat{Z}))] \leq 0$ for an arbitrary target path $\omega \in \Omega$. Let $\beta = \exp(-\alpha)$, and let $\mathcal{N}$ denote the set of the cell-time pairs $(c,t) \in \mathcal{C}\times\mathcal{T}$ such that $\zeta_{c,t}(\omega) = 1$ (i.e., such that cell c is on path ω in time period t). Now, we only need to show that $\phi(\beta) = (1-\beta)k + \beta^k \geq 1$, where $k = \sum_{(c,t)\in\mathcal{N}}(Z_{c,t} - \hat{Z}_{c,t})$. We find that $d\phi(\beta)/d\beta = 0$ for $\beta = 1$. Hence, it follows from convexity of $\phi(\cdot)$ on $(0,\infty)$ that $\phi(\cdot)$ has a minimum value of 1 for any k. This completes the proof.

We observe that the assumption of constant detection rate α in **SP1** is critical for the validity of Theorem 1 as the following counterexample illustrates.

Consider a two-cell problem instance with $C = 2$, $T = 2$, and a single target path $w = (1,2)$, i.e., the target occupies cells one and two in time periods one and two, respectively. Suppose that $\hat{Z} = (\hat{Z}_{11}, \hat{Z}_{12}, \hat{Z}_{21}, \hat{Z}_{22}) = (0,0,1,1)$ and $Z = (Z_{11}, Z_{12}, Z_{21}, Z_{22}) = (1,1,0,0)$. If the detection rate is the same in cells 1 and 2, then it is easily shown that equality holds in (43). Hence, (43) is satisfied as expected. However, if the detection rate is 2 in cell one and 1 in cell two, then the right-hand side of (43) equals $\exp(-2) \approx 0.13$ and the left-hand side equals $\exp(-3) - \exp(-2) + \exp(-1) \approx 0.28$. Hence, (43) does not hold in this case.

We refer to the cut of Theorem 1 as a secant cut, and to Algorithm 1 with (42) replaced by

$$f(Z^j) + \sum_{c,t\in\mathcal{T}} (f(Z^j + \Delta_{c,t}) - f(Z^j))(Z_{c,t} - Z^j_{c,t}) \leq \xi \quad \forall\ j = 1,2,...,i \tag{44}$$

as Algorithm 2.

We have also examined the used of submodular cuts ([19], p. 710), but find them weak.

Indeed, they lead to a cutting-plane algorithm that is 10 to 100 times slower than Algorithm 2. (New results in [1] strengthen those cuts and may result in a faster algorithm. We do not pursue that topic here, however.)

We consider a third variation of the cutting-plane algorithm where the continuous relaxation of $\mathbf{P}_i$ is solved for a set of initial iterations after which $\mathbf{P}_i$ is solved. This version is motivated by the fact that a large number of cuts can be obtained quickly by solving the continuous relaxation of $\mathbf{P}_i$. (See [30] for a similar idea.) In this version of the cutting-plane algorithm, called Algorithm 3, (44) is used when Z^{i+1} is integer valued and (42) is used otherwise.

We implement Algorithms 1-3 as well as the models **SPX**, **SP1**, **SP1-L**, and **SP1-LM** in the General Algebraic Modeling System (GAMS) Distribution 22.9 [10] on a laptop computer with 1.0 GB of RAM and 2.16 GHz processor running Windows XP. The mixed-integer linear programs $\mathbf{P}_i$ and **SP1-L** are solved (approximately) using CPLEX 11.2 [14] with default options.

We find that solution times of Algorithms 1-3 are reduced significantly when we only require a near-optimal solution of $\mathbf{P}_i$ rather than an optimal solution. Hence, we normally run Algorithms 1 and 2 with $\delta_1 = 0$ and $\delta_i = \min\{0.03, g_i/3\}$ for $i \geq 2$, where $g_i = (\overline{\xi} - \underline{\xi})/\underline{\xi}$ is computed after Step 1 of iteration i. However, we use $\delta_i = \min\{0.03, g_i/3, \delta_{i-1}/2\}$ if X^i is a repetition of a previous solution. For Algorithm 3, we set $\delta_i = 0$ if the continuous relaxation of $\mathbf{P}_i$ is solved and otherwise follow Algorithms 1 and 2. The continuous relaxation is solved until either $g_i \leq 10^{-3}$ is achieved or a user-defined maximum time is consumed. At that point, Algorithm 3 starts solving $\mathbf{P}_i$. We use 10 minutes as the maximum time, which works well in our tests.

We compare Algorithms 1-3 to the mixed-integer nonlinear programming solvers Bonmin [6] and DICOPT [11] as implemented in GAMS. DICOPT is essentially identical to Algorithm 1 with $\delta_i = 0$ for all i and an initial solution of the continuous relaxation of **SP1**. DICOPT uses CPLEX 11.2 to solve mixed-integer linear programs and MINOS 5.51 [18] for nonlinear programs. We use default options in DICOPT with exception of "stop 1," "maxcycles 1e4,"

and "epsmip 1e-5," which prevent premature termination.

Bonmin is implemented with open-source solvers Ipopt and Cbc for nonlinear programs and mixed-integer linear programs, respectively, see [5]. We allow multiple solve attempts for Ipopt (option "num_retry_unsolved_random_point" is set to 100), but otherwise use default options for Bonmin. We examine all available algorithms in Bonmin: BB, OA, QG, Hyb, and ECP. Option BB is a branch-and-bound algorithm based on continuous relaxation of the MINLP. Hence, at each node in the branch-and-bound tree a nonlinear program is solved. (This approach was examined in [9] in the case of a single searcher.) Option OA and Ecp are essentially identical to DICOPT when applied to **SP1**, but with other solvers for nonlinear programs and mixed-integer linear programs. Option QG is an implementation of the branch-and-cut algorithm in [21]. Option Hyb combines QG and OA. DICOPT, Algorithm 1, and Bonmin, with OA, QG, Hyb, and ECP, use a tangent cut of the form (42), while Algorithm 2 uses the secant cut (44). The excessive memory requirement of specialized branch-and-bound algorithms using bounds based on expected number of detections [7, 27] prohibits their testing on problem instance of interest in this paper [27].

The calculation times reported below are the CPU times required by the solvers ("resource usage" in GAMS) for DICOPT, Bonmin, and CPLEX (for **SP1-L**). For Algorithms 1-3, calculation time is the total time required including cut-generation (Step 1), model generation (Step 3), and mixed-integer linear program solver time (Step 3). Since GAMS handles cut generation in Step 1 and repeated model generation (Step 3) rather inefficiently, the fraction of the total calculation time used by CPLEX on $\mathbf{P}_i$ is typically in the range 0.65-0.95, with the lower-end values dominant for larger problem instances. Hence, the calculation times reported for Algorithms 1-3 can be improved, possibly substantially, with a more efficient implementation.

We test algorithms and models on problem instances that are essentially multi-searcher generalizations of those used in [17, 28]. Specifically, we consider a square AOI with $C = 25$, 49, 81, 121, 169, or 225 cells. Cells are numbered from left to right and from top to bottom. Hence, cell 1 is in the upper-left corner and cell C is in the lower-right corner of the AOI.

After each time period, a searcher remains in its current cell or moves to a cell directly above, below, left, or right of the current cell, if such a cell exists; a target moves similarly. Hence, the forward and reverse stars of most cells consist of five cells, except on the boundary of the AOI. For all allowable cells $d_{c,c'} = 1$. The time horizon $T = 7, 8, ..., 15$. These time horizons allow the searchers to have at least a moderate chance to detect the targets. Typically, the optimal nondetection probabilities are in the range 0.4 to 0.8.

We adopt a conditional target model and randomly generate $|\Omega|$ possible target paths using a Markov chain with a transition probability matrix defined as follows. The probability of the target remaining in a cell from one time period to the next, denoted ρ, is 0.6, with the probability of moving to any of the other allowable cells being equal. We use $|\Omega| = 1000$ unless stated otherwise. The detection rate of all searchers is $\alpha = -3(\ln 0.4)/J$, where J is the number of searchers in the problem instance. This choice of detection rate makes one searcher in a one-searcher instance as capable as ten searchers occupying the same cell in a ten-searcher instance. Moreover, if $J = 3$, then this detection rate gives a glimpse-detection probability of 0.6. All searchers occupy cell 1 in time period 0 and, hence, can search cell $1, 2$, and $\sqrt{C} + 1$ in time period 1. The target is initially located in the center of the AOI, i.e., $p_c = 1$ if $c = \sqrt{C}(\sqrt{C} - 1)/2 + (\sqrt{C} - 1)/2 + 1$ and $p_c = 0$ otherwise. These problem parameters are similar to the ones in [17, 28]. Testing not reported here shows that solution times are comparable for values of ρ and glimpse detection probability in the range $[0.1, 0.9]$.

We define the relative optimality gap to be $(\overline{\xi} - \underline{\xi})/\underline{\xi}$ for Algorithms 1-3 and comparable expressions for the other algorithms. Table 1 shows such gaps after 900 seconds of calculation time for instances of **SP1** and **SP1-L** as described above with $J = 3$, $C = 81$, and a varying number of time periods. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds. When an algorithm fails to return a feasible solution or a nonzero lower bound, we label it "failed."

Bonmin with option BB fails in all tests because no feasible solution is found; see Column 3 of Table 1. These results indicate that simple branch-and-bound algorithms are not particular efficient for **SP1**. In contrast, cutting-plane algorithms (i.e., all algorithms in

	Algorithms									
T	DICOPT	BB	OA	QG	Hyb	Ecp	CPLEX	Algo. 1	Algo. 2	Algo. 3
7	0.0034	failed	0.0315	0.0315	0 [49]	0.0315	0 [0]	0 [51]	0 [26]	0 [77]
8	0.0161	failed	0.0806	0.0806	0 [368]	0.0806	0 [1]	0 [253]	0 [86]	0 [216]
9	0.0247	failed	0.0219	0.0221	0.3002	0.3002	0 [1]	0.0147	0 [337]	0 [866]
10	0.0384	failed	0.4050	0.4050	0.4050	0.4050	0 [12]	0.0786	0.0246	0.0335
11	0.0592	failed	0.0525	0.6838	0.1315	0.0600	0 [35]	0.1664	0.0681	0.0628
12	0.0868	failed	0.8555	0.8555	0.8555	0.8555	0 [174]	0.2175	0.1159	0.0983
13	0.1052	failed	0.1521	1.3821	1.3821	1.3821	0.0094	0.2968	0.1497	0.1425
14	0.1465	failed	1.1606	1.1606	0.2022	1.1606	0.0411	0.4586	0.2085	0.2142
15	0.1630	failed	0.2933	0.9287	0.3463	0.9948	0.0650	0.5790	0.2711	0.2697

Table 1: Relative optimality gap $(\overline{\xi} - \underline{\xi})/\underline{\xi}$ for different algorithms after 900 seconds of calculation time on **SP1** or **SP1-L** with three searchers, 81 cells, and a varying time horizon T. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds. When an algorithm fails to return a feasible solution or a nonzero lower bound, we label it "failed."

Table 1 with exception of CPLEX and "BB") perform better. We observe that Bonmin, with cutting-plane based options OA, QG, Hyb, and Ecp, occasionally terminates prior to 900 seconds due to inconsistent performance of the open-source solvers.

	Algorithms									
T	DICOPT	BB	OA	QG	Hyb	Ecp	CPLEX	Algo. 1	Algo. 2	Algo. 3
7	0.0007	failed	0.0347	0.0347	0.0347	0.0347	0 [4]	0 [120]	0 [143]	0 [137]
8	0.0008	failed	0.0654	0.0654	0 [782]	0 [200]	0 [7]	0.0003	0 [345]	0 [361]
9	0.0010	failed	0.0218	0.3002	0.0034	0.1811	0 [12]	0.0024	0.0020	0.0008
10	0.0037	failed	0.2243	0.2243	0.0379	0.1053	0 [20]	0.0043	0.0041	0.0040
11	0.0036	failed	0.1028	0.0858	0.8304	0.8304	0 [531]	0.0151	0.0124	0.0155
12	0.0049	failed	0.8026	0.8026	0.8026	0.8026	0.0007	0.0356	0.0321	0.0343
13	0.0101	failed	1.0302	1.0303	0.7304	0.7304	0.0008	0.0488	0.0425	0.0475
14	0.0327	failed	0.3311	0.9813	0.9813	0.9813	failed	0.0920	0.0719	0.0755
15	0.0193	failed	0.2846	1.1157	0.3039	1.0446	failed	0.1385	0.1178	0.1179

Table 2: Relative optimality gaps. Same problem instances and parameters as in Table 1 but with 15 searchers.

The solution of **SP1-L** by CPLEX is by far the most efficient approach; see Column 8 of Table 1. Table 1 also illustrates the benefit of the secant cut (44) in Algorithm 2 over the tangent cut (42) in Algorithm 1. Typically, compared to Algorithm 1, Algorithm 2 reduces the solution time, or the optimality gap remaining after a fixed amount of calculation time, by a factor of two. Algorithm 3 appears to be comparable to Algorithm 2, but sometimes a little faster on harder problem instances. We also note that DICOPT is substantially slower than Algorithm 2 on smaller instances, but gains the advantage over Algorithm 2 as problem

size increases. The main reason for this effect is the increase in overhead associated with our implementation of Algorithm 2 as problem size increases. In the largest example (see the last row of Table 1), DICOPT carries out 101 major iterations in 900 seconds while Algorithm 2 manages only 43 iterations. However, the work per iteration should be less for Algorithm 2 (which obtains near-optimal solution of $\mathbf{P}_i$) than for DICOPT (which obtains optimal solutions of its master problems). These observations illustrates the substantial overhead associated with our implementation of Algorithm 2 as well as the strength of the secant cut in Algorithm 2 over the tangent cut used in DICOPT.

	Algorithms		
$\|\Omega\|$	DICOPT	CPLEX	Algo. 2
1000	0.0161	0 [1]	0 [86]
2000	0.0189	0 [10]	0 [270]
4000	0.0251	0 [48]	0 [888]
8000	0.0268	0 [177]	0.0080
16000	0.0246	0 [806]	0.0254
32000	0.0269	failed	0.0402

Table 3: Relative optimality gap for different algorithms after 900 seconds of calculation time on **SP1** (columns 2 and 4) or **SP1-L** (column 3) with three searchers, 81 cells, time horizon 8, and varying number of possible target paths $|\Omega|$. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds. When an algorithm fails to return a feasible solution or a nonzero lower bound, we label it "failed."

Table 2 reports relative optimality gaps for the same cases as in Table 1 but with $J = 15$. The advantage of the secant cut (44) of Algorithm 2 as compared to the tangent cut (42) of Algorithm 1 is still typically present. The effect, however, is much reduced as $Z_{c,t}$ may now be moderately large integers and finite differences are close to the corresponding partial derivatives. We find that the solution of **SP1-L** by CPLEX is faster than the alternatives as long as **SP1-L** is not too large. **SP1-L** has $(JT + 1)|\Omega|$ linearization constraints, which, after reductions, result in a reduced mixed-integer linear program with 145,039 rows, 3,777 columns, and 1,142,886 nonzero elements for $T = 14$. CPLEX fails to find a feasible solution within the time limit of 900 seconds for this problem instance and the one with $T = 15$.

We examine further the effect of large instances of **SP1-L** in Table 3. That table reports relative optimality gaps as the number of possible target paths $|\Omega|$ grows. We see again that

	Algorithms			
C	DICOPT	CPLEX	Algo. 2	Algo. 3
25	0.2152	0.0327	0.2412	0.1595
49	0.0927	0 [210]	0.1215	0.1026
81	0.0396	0 [13]	0.0246	0.0372
121	0.0138	0 [3]	0 [451]	0.0001
169	0.0056	0 [1]	0 [167]	0 [336]
225	0 [6]	0 [0]	0 [106]	0 [157]

Table 4: Relative optimality gap for different algorithms after 900 seconds of calculation time on **SP1** or **SP1-L** with three searchers, varying number of cells, and time horizon 10. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds.

	Algorithms			
C	DICOPT	CPLEX	Algo. 2	Algo. 3
25	0.5271	0.2069	2.8603	0.6393
49	0.2505	0.1015	0.3954	0.4191
81	0.1626	0.0650	0.2711	0.3067
121	0.1212	0.0191	0.1704	0.1820
169	0.0676	0 [104]	0.0984	0.1292
225	0.0207	0 [8]	0.0312	0.0456

Table 5: Relative optimality gap for problem instances as in Table 4, but with time horizon 15.

the solution of **SP1-L** by CPLEX is clearly the fastest approach as long as the number of possible target paths is moderate. In the instances examined, 16000 target paths define the break point. We also observe that Algorithm 2 significantly outperforms DICOPT unless the number of possible target paths is large. In such cases, the overhead associated with our implementation of Algorithm 2 becomes substantial. For $|\Omega| = 32000$, DICOPT manages to carry out 87 major iterations in 900 seconds and reaches an optimality gap of 0.0269, while Algorithm 2 completes only 10 iterations but still reaches a gap of 0.0402. This indicates again the strength of the secant cut used in Algorithm 2 as compared to the tangent cut used in DICOPT.

Table 4 displays the optimality gaps for the same problem instances and parameters as in Table 1, but for a varying number of cells. We again use $|\Omega| = 1000$ possible target paths. Interestingly, problem instances with more cells are easier to solve than those with fewer cells. This effect results from the fact that a large number of cells makes the search more difficult as each possible target path is searched only a relatively small number of times. Hence, the nondetection probability tends to be large and the corresponding cuts (42) and

(44) have relatively large negative coefficients. This implies strong cuts in DICOPT and Algorithms 1-3, as well as relatively few active constraints in (21). This effect was also eluded to in [9] for the case of a single searcher. As a result, problem instances with 225 cells solve in just few seconds for short and moderate time horizons. Table 4 shows that CPLEX is less influenced by the effect than the algorithms based on cutting planes. Moreover, in cases with weak cuts (see the smaller problem instances in Table 4) solving the continuous relaxation of $\mathbf{P}_i$, as done in Algorithm 3, appears beneficial. Table 5 show similar results to those of Table 4, but with a longer time horizon.

	Algorithms			
J	DICOPT	CPLEX	Algo. 2	Algo. 3
1	0.1367	0 [5]	0 [177]	0 [550]
2	0.1041	0 [13]	0.0288	0.0743
3	0.0384	0 [12]	0.0246	0.0335
4	0.0271	0 [19]	0.0253	0.0277
5	0.0160	0 [24]	0.0231	0.0203
6	0.0115	0 [34]	0.0157	0.0149
8	0.0079	0 [67]	0.0108	0.0105
10	0.0087	0 [63]	0.0086	0.0084
15	0.0037	0 [20]	0.0041	0.0040
20	0.0017	0 [32]	0.0031	0.0040
30	0.0016	0 [580]	0.0032	0.0042
50	0.0015	0.0000	0.0037	0.0039

Table 6: Relative optimality gap for different algorithms after 900 seconds of calculation time on **SP1** or **SP1-L** with varying number of searchers, 81 cells, and time horizon 10. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds.

We next examine the effect of increasing number of searchers. Tables 6 and 7 report optimality gaps for 1-50 searchers for $T = 10$ and $T = 15$, respectively. Again, CPLEX outperforms the other algorithms, except when $J > 8$ and $T = 15$. Moreover, optimality gaps tend to decrease as the number of searchers increases. This results from the fact that the continuous relaxations of $\mathbf{P}_i$ and **SP1-L** are stronger for more searchers because the number of searchers in a cell may be a moderately larger integer in those cases. (Note that with more searchers there is a tendency of multiple searchers occupying the same cell at the same time.) From Tables 6 and 7 and the fact that there are fast specialized branch-and-bound algorithms available for single searcher problems (see [28]), we conclude that problem

	Algorithms			
J	DICOPT	CPLEX	Algo. 2	Algo. 3
1	0.7047	0 [304]	0.2283	0.3554
2	0.2801	0.0850	0.3935	0.4659
3	0.1630	0.0650	0.2711	0.2697
4	0.0908	0.0385	0.1741	0.1968
5	0.0712	0.0258	0.1396	0.1709
6	0.0663	0 0129	0.1240	0.1591
8	0.0403	0.0092	0.1112	0.1408
10	0.0499	failed	0.1015	0.1246
15	0.0193	failed	0.1178	0.1179
20	0.1693	failed	0.1257	0.1344
30	0.0093	failed	0.1259	0.1251
50	0.7720	failed	0.1383	0.1346

Table 7: Relative optimality gap as in Table 6 but with time horizon 15. When an algorithm fails to return a feasible solution or a nonzero lower bound, we label it "failed."

instances with 2-5 searchers tend to be the most difficult to solve.

While solving **SP1-L** using CPLEX may be the preferred solution approach in most problem instances above, the approach fails in cases with a large number of possible target paths (see Table 3). Hence, we next consider the situation with an extremely large number of possible target paths, but add the assumption of a Markovian target model.

4.2 Solution of SP1 and SP1-LM for a Markovian Target Model

This section empirically compares the efficiency of solving **SP1** for a Markovian target model by means of Algorithm 2, with that of solving the equivalent linearized model **SP1-LM** by means of CPLEX. We have examined several cutting-plane algorithms for solving **SP1**, but here only report the results of Algorithm 2, as they are typically the best. We note, however, that the optimality gaps obtained by Algorithm 2 are typically half of those of Algorithm 1 in the case of few searchers. Hence, the secant cut (44) remains superior to the tangent cut (42).

A good cutting-plane algorithm requires efficient means for evaluating $f(Z)$, $\nabla f(Z)$, as well as the finite difference in (44) even for large $|\Omega|$. Since the expressions (13) and (40) use all possible target paths $\omega \in \Omega$, they are not useful in practice when $|\Omega|$ is large. Brown [4] introduces alternative, but equivalent expressions for $f(Z)$ and $\nabla f(Z)$ that utilize the fact

that the target movement follows a Markovian target model. We repeat those expressions here with a slight generalization to the case of a time-dependent Markov transition matrix and argue how they lead to a simple expression for the finite differences used in (44).

Given a search plan Z, let $r_{c,t}(Z)$ be the probability that the target occupies cell c in time period t and that it is not detected in time periods 1, 2, ..., $t-1$, and let $s_{c,t}(Z)$ denote the probability that the target is not detected in time periods $t+1$, $t+2$, ..., T given that it occupies cell c in time period t. Let $r_t(Z) = [r_{1,t}(Z), r_{2,t}(Z), \ldots, r_{C,t}(Z)]$, and let $s_t(Z) = [s_{1,t}(Z), s_{2,t}(Z), \ldots, s_{C,t}(Z)]$. We define $r_{c,1}(Z) = p_c$ and $s_{c,T}(Z) = 1$ for any cell $c \in \mathcal{C}$. Thus, $r_t(Z)$ and $s_t(Z)$ may be calculated recursively by

$$r_t(Z) = [r_{1,t-1}(Z)\exp(-\alpha Z_{1,t-1}), \ldots, r_{C,t-1}(Z)\exp(-\alpha Z_{C,t-1})]\Gamma_{t-1}, \tag{45}$$

and

$$s_t(Z) = [s_{1,t+1}(Z)\exp(-\alpha Z_{1,t+1}), \ldots, s_{C,t+1}(Z)\exp(-\alpha Z_{C,t+1})]\Gamma_t', \tag{46}$$

where Γ_t' denotes the transpose matrix of Γ_t. In this notation, for any $t \in \mathcal{T}$, we find that

$$f(Z) = \sum_{c\in\mathcal{C}} r_{c,t}(Z)\exp(-\alpha Z_{c,t})s_{c,t}(Z), \tag{47}$$

and components of $\nabla f(Z)$ are

$$\frac{\partial f(Z)}{\partial Z_{c,t}} = -\alpha r_{c,t}(Z)\exp(-\alpha Z_{c,t})s_{c,t}(Z). \tag{48}$$

The calculation of finite differences $f(Z+\Delta_{c,t}) - f(Z)$ follows similarly:

$$\begin{aligned} & f(Z+\Delta_{c,t}) - f(Z) \\ = & \sum_{c'\in\mathcal{C}} r_{c',t}(Z)[\exp(-\alpha Z_{c',t} - \alpha\Delta_{c,t}) - \exp(-\alpha Z_{c',t})]s_{c',t}(Z) \\ = & \; r_{c,t}(Z)[\exp(-\alpha(Z_{c,t}+1)) - \exp(-\alpha Z_{c,t})]s_{c,t}(Z). \end{aligned} \tag{49}$$

Thus, $f(Z)$, its gradient, and finite difference can be evaluated with moderate computational effort. We utilize (47) and (49) in Algorithm 2 in the following computational tests on the same problems instances as in Section 4.1 and with the same computational environment.

However, we now consider all possible target paths induced by the Markov chain defined in Section 4.1.

We solve **SP1-LM** using CPLEX, with default options except for branching direction (up first, i.e., brdir 1) and the use of disjunctive cuts (disjcuts 3), which was found to generally perform slightly better then the default options. Columns 2 and 3 of Table 8 show relative optimality gaps for CPLEX and Algorithm 2 after 900 seconds on problem instances with three searchers. CPLEX performs significantly better when $T \leq 10$. For larger T, the big-M formulation of **SP1-LM** leads to weak relaxations and long run times. Columns 4 and 5 of Table 8 display analogous results for $J = 15$. These instances of **SP1-LM** become large, and CPLEX competes poorly with Algorithm 2.

	$J = 3$ Searchers		$J = 15$ Searchers	
T	CPLEX	Algo. 2	CPLEX	Algo. 2
7	0 [0]	0 [7]	0.0000	0 [13]
8	0 [1]	0 [80]	0.2055	0.0002
9	0 [7]	0 [203]	9.4739	0.0021
10	0 [172]	0.0287	failed	0.0043
11	0.1650	0.0729	failed	0.0075
12	0.4964	0.1158	failed	0.0106
13	1.1479	0.1528	failed	0.0274
14	2.7033	0.1737	failed	0.0599
15	21.4129	0.2356	failed	0.0866

Table 8: Relative optimality gap for different algorithms after 900 seconds of calculation time on **SP1** or **SP1-LM** with three and 15 searchers, 81 cells, and a varying time horizon T. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds. When an algorithm fails to return a feasible solution or a nonzero lower bound, we label it "failed."

The effect of number of searchers, J, is further examined in Table 9. As when $|\Omega|$ is moderate (see Tables 6 and 7), Algorithm 2 tends to obtain near-optimal solutions faster as J grows. In contrast, **SP1-LM** becomes increasingly harder to solve as J increases and the resulting model becomes larger. Hence, the two approaches are complimentary: four searchers represents a "cross-over point" above which Algorithm 2 has a clear advantage and below which the solution of **SP1-LM** prevails.

Table 10 examines the effect of number of cells, C. As in Tables 4 and 5, larger number of cells improves the strength of the cuts in Algorithm 2 and reduces the optimality gaps. A

	Algorithms	
J	CPLEX	Algo. 2
1	0 [1]	0 [97]
2	0 [12]	0.0287
3	0 [172]	0.0287
4	0.1189	0.0320
5	0.3246	0.0227
10	failed	0.0074
15	failed	0.0043

Table 9: Relative optimality gap for different algorithms after 900 seconds of calculation time on **SP1** or **SP1-L** with varying number of searchers, 81 cells, and time horizon 10. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds. When an algorithm fails to return a feasible solution or a nonzero lower bound, we label it "failed."

comparable strengthening of the continuous relaxation of **SP1-LM** occurs as the number of cells increases. This allows the solution of large problem instances with a Markovian target model in seconds for small and moderate T.

	Algorithms	
C	CPLEX	Algo. 2
25	failed	0.2630
49	0.6450	0.1244
81	0 [172]	0.0287
121	0 [10]	0 [15]
169	0 [1]	0 [31]
225	0 [1]	0 [20]

Table 10: Relative optimality gap for different algorithms after 900 seconds of calculation time on **SP1** or **SP1-LM** with three searchers, varying cells, and time horizon 10. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 900 seconds. When an algorithm fails to return a feasible solution or a nonzero lower bound, we label it "failed."

4.3 Solution of SPX

We now return to the general **SPX** with inhomogeneous searchers, deconfliction constraints, and multiple targets. As discussed in Section 3.3, it is straightforward to construct linear models of **SPX**. However, such models tend to be extremely large and we therefore focus on a cutting-plane algorithm. We let "Algorithm 4" denote the extension of Algorithm 1 to **SPX**. Algorithm 4 is essentially identical to Algorithm 1, except that the following problem is solved during the i-th iteration instead of $\mathbf{P}_i$:

$$\mathbf{PX}_i : \min \xi \tag{50}$$

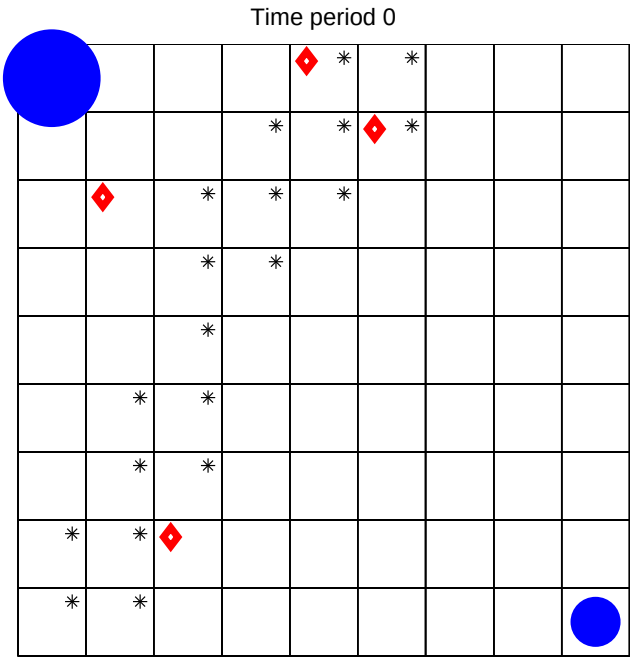

Figure 1: Area of interest with diamonds indicating initial location for moving targets, asterisks indicating difficult-to-search cells, and circles indicating initial locations of searchers. The number of searchers is proportional to the radius of a circle.

$$\text{s.t.} f_k(Y^j) + \nabla f_k(Y^j)'(Y - Y^j) \leq \xi \qquad \forall\ k, j = 1, 2, ..., i \tag{51}$$

$$(6) - (12).$$

Moreover, since each target in **SPX** moves independently, $f_k(Y_k)$ and $\nabla f_k(Y_k)$ are computable by extensions of (47) and (48). We also extended Algorithm 2 to **SPX** using an approach similar to the one described in Section 3.3. The resulting algorithm is faster than Algorithm 4 only if few searchers are present and all detection rates can be expressed as small integer multiples of a positive constant. Hence, below we focus on Algorithm 4.

We apply Algorithm 4 to problem instances with $C = 81$ similar to the instances examined in Sections 4.1-4.2; see Figure 1. Cells are numbered as before. We consider four targets that follow Markovian target models. At time period one, one target occupies each of the cells 5, 15, 20, and 66 (marked with diamonds in Figure 1). After each time period, a target remains in its current cell or move to a cell directly above, below, left, or right of the current cell if such a cell exists. The probabilities of a target remaining in a cell from

one time period to the next is 0.4, 0.3, 0.2, and 0.1, respectively, for the four targets; the probability of moving to any of the other allowable cells is equal. Hence, the target that initially occupies cell 5 moves slowly, the target that initially occupies cell 66 moves quickly, and the other two targets move at intermediate speeds.

We consider two classes of airborne searchers and set the travel time $d_{l,c,c'}$=max{1, round($\delta_{c,c'}/\nu_l$)}, where round(a) is the nearest integer to a, $\delta_{c,c'}$ is the distance between c and c' measured in the Euclidean distance between the center of the cells, and ν_l is the speed of searchers of class l; $\nu_1 = 1$, $\nu_2 = 2$ cells per time period. Moreover, we let the forward stars $\mathcal{F}_l(c) = \mathcal{F}^1(c) \cup \mathcal{F}_l^2(c), l = 1, 2$, where $\mathcal{F}^1(c)$ equals the set consisting of c and the four cells sharing a side with c, if they exist, and $\mathcal{F}_l^2(c)$ equals the set of all cells c' with $d_{l,c,c'} \in [3, 5]$, if they exist. Hence, a searcher can after a time period either proceed and search "locally" (i.e., select a cell in $\mathcal{F}^1(c)$) or transit for several time periods to a distant cell (i.e., select a cell in $\mathcal{F}_l^2(c)$). The reverse stars $\mathcal{R}_l(c), l = 1, 2$, are defined similarly.

	Number of searchers		Glimpse-detection probability							
			Searcher Class 1				Searcher Class 2			
Scenario	J_1	J_2	$c = c'$	$c \neq c'$	$c = c'*$	$c \neq c'*$	$c = c'$	$c \neq c'$	$c = c'*$	$c \neq c'*$
1	2	1	0.50	0.29	0.29	0.16	0.29	0.16	0.16	0.09
2	4	2	0.50	0.29	0.29	0.16	0.29	0.16	0.16	0.09
3	20	10	0.07	0.03	0.03	0.02	0.03	0.02	0.02	0.01

Table 11: Scenarios defining problem instances of **SPX**. Columns marked with $c = c'$ ($c \neq c'$) give glimpse-detection probability for a searcher that occupy (not occupy) the current cell previously. An asterisk indicates a column with glimpse-detection probability for difficult-to-search cells; see Figure 1.

We consider three scenarios with variable glimpse-detection probability and number of searchers as summarized in Table 11. In scenario 1, two searchers of class 1 occupy cell 1 in time period 0 and one searcher of class 2 initially occupies cell 81. The glimpse-detection probability of a searcher of the first class is 0.50 if the searcher occupied the current cell in the last time period ($c = c'$), but the searcher's detection rate is reduced with a factor 0.5 if the searcher just moved into the cell ($c \neq c'$). In view of (3), this implies a glimpse-detection probability of 0.29; see Table 11. This reduction accounts for the effect, which we have observed in field experiments with actual UAVs [16, 24], that a searcher often wastes some

search time transiting from one cell to another even if the cells are adjacent. Using the model flexibility of **SPX**, we incorporate this effect without resorting to a fine time discretization.

When a searcher occupies one of the cells marked with an asterisk in Figure 1, all detection rates are reduced by a factor of 0.5. These cells represent areas with poor search conditions and consequently low detection rates. This results in a glimpse-detection probability of 0.29 when $c = c'$ and 0.16 when $c \neq c'$. For the class-2 searcher, the detection rate is reduced with a factor of 0.5 compared to class 1 in all situations, with resulting glimpse-detection rates given in Table 11.

Scenario 2 is identical to scenario 1 except it has four class-1 searchers and two class-2 searchers. Scenario 3 is identical to scenario 1 except that there are 20 class-1 searchers and 10 class-2 searchers, and the detection rate is reduced with a factor of 0.1 in all situations. The last row of Table 11 gives the resulting glimpse-detection probabilities. We note that the collective search effort (i.e., total detection rate) of the searchers in scenario 3 is therefore identical to that of those in scenario 1. Scenario 3, however, allows more flexibility as the search effort can be spread more widely.

We consider both the situation with and without deconfliction constraints (9) and (10). In these scenarios, deconfliction amounts to ensuring that at most one searcher occupies a cell each time period and that a searcher is not allowed to move from a cell c to an adjacent cell $c' \in \mathcal{F}^1(c)$ when another searcher makes the opposite move from c' to c. We assume that transit to a distance cell (i.e., a cell $c' \in \mathcal{F}_l^2(c)$) takes place by flying at high altitude, while search is carried out at low altitude. In **SPX**, we incorporate these constraints by setting $n_c = 1$ for all c and $\mathcal{D}(l, c, c', t) = \{(l', c'', c''', t') \mid l' \in \mathcal{L}, c'' = c', c''' = c, t' = t\}$ whenever $c' \in \mathcal{F}^1(c), c' \neq c$ and otherwise $\mathcal{D}(l, c, c', t) = \emptyset$. Since searchers transiting between distance cells can be separated easily by altitude, we allow the routes of such searchers to cross each other as well as to cross over searchers occupying cells.

Table 12 shows lower and upper bounds on the optimal value of **SPX** as well as the corresponding relative optimality gaps after 15 and 60 minutes of calculation time of Algorithm 4 for these scenarios with $T = 8$, 10, and 12. Columns 4 and 5 present the results

Scenario	T	Measure	No deconfliction After 15 min	No deconfliction After 60 min	Deconfliction After 15 min	Deconfliction After 60 min
1	8	Lower bound	0.8514	0.8577	0.8519	0.8578
		Upper bound	0.8741	0.8663	0.8726	0.8663
		Relative gap	0.0267	0.0100	0.0244	0.0099
2	8	Lower bound	0.7191	0.7267	0.7214	0.7310
		Upper bound	0.7521	0.7521	0.7736	0.7598
		Relative gap	0.0458	0.0349	0.0724	0.0395
3	8	Lower bound	0.8367	0.8370	0.8875	
		Upper bound	0.8374	0.8370	0.8875	
		Relative gap	0.0008	0.0000	0 [390]	
1	10	Lower bound	0.7773	0.7958	0.7809	0.8005
		Upper bound	0.8330	0.8330	0.8423	0.8423
		Relative gap	0.0716	0.0467	0.0786	0.0523
2	10	Lower bound	0.6263	0.6345	0.6279	0.6364
		Upper bound	0.7097	0.7033	0.7107	0.7107
		Relative gap	0.1333	0.1083	0.1319	0.1168
3	10	Lower bound	0.7877	0.7879	0.8400	0.8402
		Upper bound	0.7894	0.7894	0.8408	0.8405
		Relative gap	0.0022	0.0019	0.0010	0.0004
1	12	Lower bound	0.7258	0.7419	0.7273	0.7415
		Upper bound	0.8181	0.8181	0.8149	0.8149
		Relative gap	0.1271	0.1027	0.1205	0.0991
2	12	Lower bound	0.5433	0.5537	0.5427	0.5506
		Upper bound	0.6750	0.6637	0.6544	0.6544
		Relative gap	0.2425	0.1987	0.2057	0.1886
3	12	Lower bound	0.7446	0.7450	0.7970	0.7974
		Upper bound	0.7479	0.7469	0.7994	0.7986
		Relative gap	0.0044	0.0026	0.0031	0.0015

Table 12: Lower and upper bounds on the optimal value of **SPX** as well as relative optimality gaps after 15 and 60 minutes of calculation times of Algorithm 4 for scenarios 1-3 with and without deconfliction constraints. The time in seconds to reach optimality is reported in brackets when zero gap is achieved within 60 minutes.

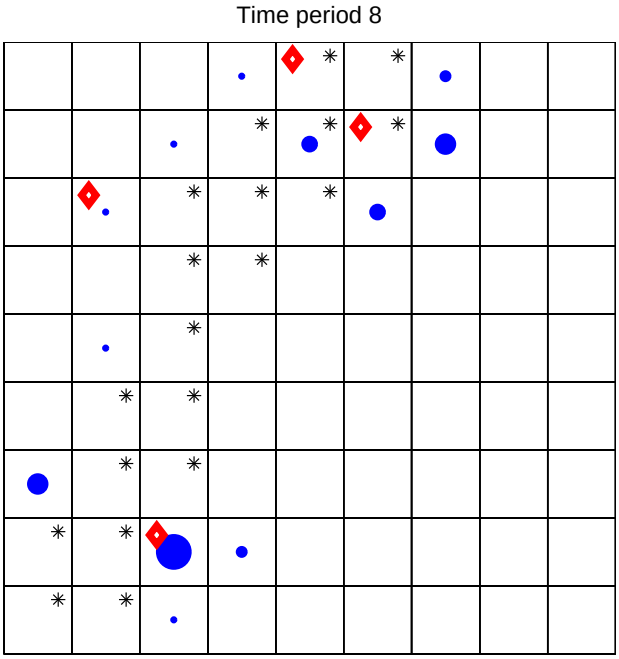

Figure 2: Optimal searcher location during time period 8 for scenario 3, $T = 8$, and no deconfliction constraints. The radius of a circle is proportional to the number of searchers occupying the corresponding cell during time period 8 (cells 4, 7, 12, 14, 16, 20, 24, 38, 55, 66, 67, and 75 contain 1, 2, 1, 3, 4, 1, 3, 1, 4, 7, 2, and 1 searchers, respectively). Diamonds indicate initial location for moving targets and asterisks indicate difficult-to-search cells.

for the case with no deconfliction constraints, while columns 6 and 7 include deconfliction constraints. As in the case of Algorithms 1-3, Algorithm 4 solves problem instances with more searchers (as in scenario 3) quicker than those with fewer searchers (as in scenario 1). We also find longer time horizons to be more difficult, again primarily due to the weaker cuts in the case of smaller nondetection probabilities.

Deconfliction constraints restrict **SPX** and result in an increase in the optimal value. In scenarios 1 and 2, the change is small due to the relatively low number of searchers. Deconfliction constraints increase the optimal value with about 0.05 in scenario 3 where 30 searchers are present. Hence, deconfliction constraints effectively force the searchers to give up 0.05 in probability of detection.

Figure 2 illustrates the optimal location of searchers in time period 8 for scenario 3, $T = 8$, and no deconfliction constraints. The radius of a circle is proportional to the number

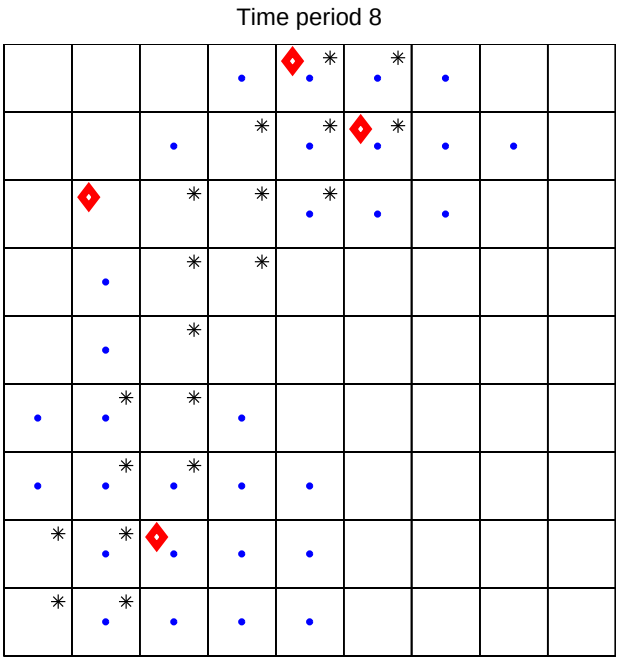

Figure 3: Optimal searcher location during time period 8 for scenario 3, $T = 8$, and deconfliction constraints. A dot indicates one searcher.

of searchers located in the corresponding cell during time period 8 (cells 4, 7, 12, 14, 16, 20, 24, 38, 55, 66, 67, and 75 contain 1, 2, 1, 3, 4, 1, 3, 1, 4, 7, 2, and 1 searchers, respectively). We observe that multiple searchers focus on a relatively small number of cells with high probability of containing targets. This tendency is also present in earlier time periods. In fact, the ten searchers initially occupying the bottom-right corner of the AOI (see Figure 1) ignore cells near their initial location, make a long-distance move to the nearest target (cell 66), and, due to their high speed, arrive there for search in time period 3. Seven of the ten searchers remain in cell 66 for the remaining time periods to avoid the lower detection rate associated with moving to another cell. Three searchers fan out to adjacent cells, but noticeably avoid the difficult-to-search cells marked with asterisks in Figure 2. The 20 searchers starting in the upper-left corner (see Figure 1) spread more widely, even tackle difficult-to-search cells, but remain somewhat clustered.

Figure 3 shows the optimal location of searchers in time period 8 as in Figure 2, but in the case with deconfliction constraints. Now, each cell allows at most one searcher and the

searchers are forced to spread out. As seen from row 10 in Table 12, spreading the effort is less effective and the optimal value increases from 0.8370 to 0.8875.

	No deconfliction		Deconfliction	
T	UB	Rel. gap	UB	Rel. gap
10	0.7894	0.0019	0.8405	0.0004
14	0.7093	0.0038	0.7600	0.0016
18	0.6433	0.0063	0.6906	0.0023
22	0.5869	0.0117	0.6292	0.0021
26	0.5393	0.0225	0.5755	0.0041
30	0.5126	0.0865	0.5266	0.0053
34	0.4755	0.1171	0.4833	0.0093
38	0.4473	0.1811	0.4452	0.0222
42	0.4102	0.2337	0.4114	0.0598

Table 13: Upper bounds (UB) on the optimal value of **SPX** as well as relative optimality gaps after 60 minutes of calculation times of Algorithm 4 for scenario 3 with varying T, and with and without deconfliction constraints.

Table 13 further examines the upper bounds (UB) on the optimal value of **SPX** and corresponding relative optimality gaps in scenario 3 as T increases. As in Table 12, we find a worsening in solution quality after 60 minutes of calculation time as T increases. However, the increase is moderate and essentially insignificant for the case with deconfliction constraints. We are able to obtain near-optimal solutions even for long time horizons. Interestingly, deconfliction constraints reduce optimality gaps in these instances even though they increase the model size.

5 Conclusions

This paper has presented models and algorithms for a discrete-time route-optimization problem, denoted SP, where multiple searchers and one or more probabilistically moving targets operate on a finite set of cells. We have formulated a novel convex mixed-integer nonlinear program (MINLP) for SP that considers searcher deconfliction, time of search, and target- and location-dependent search effectiveness. MINLP allows modeling of real-world situations not previously considered including those with many searchers.

We propose two solution approaches for the MINLP that result in the first practical, exact algorithms for SP. One approach is based on the cutting-plane method and leads to

near-optimal solutions with 5% relative optimality gap in less than 15 minutes of calculation time for problem instances with three searchers, 81 cells, and 10 time periods. We enhance the approach by developing a specialized secant cut that reduces the solution time with a factor of two in direct comparisons with a standard tangent cut on difficult instances of the MINLP. In all problem instances examined, existing branch-and-bound algorithms either fail to find a feasible solution within 15 minutes or require memory in excess of one gigabytes.

The other approach is based on two novel linearizations of the MINLP available in important special cases involving a single target. In the case of a moderate number of possible target paths, a linearization is easily solved by standard solvers when the number of searchers is less than 10. This leads to reduction in solution times with one or two orders of magnitude when compared to the cutting-plane approach and, for example, allows the solution of a problem instance with three searchers, 81 cells, and 10 time periods in 12 seconds. In the case of a target moving according to a Markov chain, the solution times for solving a linearization of MINLP is longer but an instance with three searchers, 81 cells, and 10 time periods is still solved to optimality in less than three minutes.

The cutting-plane approach, and to a less extent the linearization approach, scale well as the number of searchers grows. Empirically, we observe that the cutting-plane approach exhibits an approximately constant solution time to reach near-optimality as the number of searchers grows. This enables the computation of near-optimal solutions of problem instances with up to 50 searchers in less than 15 minutes. In a realistic scenario involving 30 searchers divided in two classes with different speed and sensor quality, four targets of variable characteristics, deconfliction constraints, 81 inhomogeneous cells, and a long time horizon of 38 periods, we obtain a 3% near-optimal solution in 30 minutes. Hence, we overcome, in part, the difficulty of solving problem instances with many searchers.

Problem instances remain challenging when they have two or more classes of searchers, each with only a few searchers, and when long time horizons lead to low nondetection probabilities. A heuristic allocation of searchers to nonoverlapping subareas may be a practical approach in the first situation and a rolling-time horizon and/or fix-and-relax optimization

may heuristically address the second situation. However, fast solution of these classes of instances of SP remains a goal for future research.

Acknowledgement

The first author acknowledges financial support from the Office of Naval Research under grant N0001409AF00002. The authors thank Distinguished Professor R. Kevin Wood, Naval Postgraduate School, for valuable advice during the course of this study and Professor Moshe Kress, Naval Postgraduate School, for commenting on a draft of the paper.

References

[1] S. Ahmed and A. Atamturk. Maximizing a class of submodular utility functions. *www.optimization-online.org*, 2009. (Last accessed 10 July 2009).

[2] P. Bonami, L.T. Biegler, A.R. Conn, G. Cornuejols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wachter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–204, 2008.

[3] F. Bourgult, T. Furukawa, and H. F. Durrant-Whyte. Coordinated decentralized search for a lost target in a bayesian world. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligence and Systems*, pages 48–53, Las Vegas, Nevada, 2003.

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

[5] COIN-OR. http://www.coin-or.org/ (last accessed 10 June 2009).

[6] COIN-OR. *Bonmin web page*. http://www.coin-or.org/Bonmin/ (last accessed 10 June 2009).

[7] 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.

[8] 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.

[9] 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.

[10] GAMS. *GAMS Distribution 22.9*. GAMS Development Corporation, Washington, DC, 2008. http://www.gams.com (last accessed 22 December 2008).

[11] I. E. Grossmann, J. Viswanathan, A. Vecchietti, R. Raman, and E. Kalvelagen. *DICOPT user manual*. GAMS Development Corporation, Washington, DC, 2008. http://www.gams.com (last accessed 22 December 2008).

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

[13] G. Hollinger and S. Singh. Proofs and experiments in scalable, near-optimal search by multiple robots. In *Proceedings of Robotics: Science and Systems Conference*, Zurich, Switzerland, 2008. Available at http://www.roboticsproceedings.org/ (last accessed 10 July 2009).

[14] ILOG. *CPLEX 11.2 Documentation*. ILOG, Inc., Mountain View, California, 2008. http://www.cplex.com (last accessed 22 December 2008).

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

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

[17] 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.

[18] B. Murtagh and M. Saunders. MINOS 5.5 User's guide. Technical Report SOL-86-20R, System Optimization Laboratory, Stanford University, Stanford, California, 1998.

[19] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley, New York, New York, 1988.

[20] U.S. Department of Defense. Fy2009-2034 unmanned systems integrated roadmap. Available at: www.acq.osd.mil/uas/docs/UMSIntegratedRoadmap2009.pdf (last accessed 1 September 2009).

[21] I. Quesada and I.E. Grossmann. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers & Chemical Engineering*, 18:563–578, 1992.

[22] R. L. Rardin. *Optimization in operations research*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[23] J. R. Riehl, G. E. Collins, and J. P. Hespanha. Cooperative graph-based model predictive search. In *Proceedings of 46th IEEE Conference on Decision and Control*, pages 2998–3004, New Orleans, Louisiana, 2007.

[24] J. O. Royset and D. N. Reber. Optimizing routing of unmanned aerial systems for the interdiction of improvised explosive devices. *In review*, 2009. Available at http://faculty.nps.edu/joroyset/roysetpa.htm (last accessed 23 July 2009).

[25] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick. An overview of emerging results in cooperative UAV control. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, pages 602–607, Atlantis, Bahamas, 2004.

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

[27] H. Sato. *Path optimization for single and multiple searchers: models and algorithms*. Phd thesis, Naval Postgraduate School, Monterey, California, 2008.

[28] H. Sato and J.O. Royset. Path optimization for the resource-constrained searcher. *In review*, 2009. Available at http://faculty.nps.edu/joroyset/roysetpa.htm (last accessed 23 July 2009).

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

[30] C. Still and T. Westerlund. Solving convex MINLP optimization problems using a sequential cutting plane algorithm. *Computational Optimization and Applications*, 34(1):63–83, 2006.

[31] L. D. Stone. *Theory of Optimal Search*. Informs, Hanover, Maryland, 2. edition, 2004.

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

[33] A. R. Washburn. *Search and Detection*. INFORMS, Linthicum, Maryland, 4. edition, 2002.

[34] T. Westerlund and F. Pettersson. An extended cutting plane method for solving convex MINLP problems. *Computers & Chemical Engineering*, 19:131–136, 1995.

[35] 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.

[36] Y. Yang, A.A. Minai, and M. M. Polycarpou. Decentralized cooperative search by networks uavs in an uncertain environment. In *Proceedings of the 2004 American Control Conference*, pages 5558–5563, Boston, Massachusetts, 2004.