Generalized Orienteering Problem with Resource Dependent Rewards

Jesse Pietz, Johannes O. Royset

Operations Research Department, Naval Postgraduate School, Monterey, California

February 19, 2013

Abstract

We introduce a generalized Orienteering Problem where, as usual, a vehicle is routed from a prescribed start node, through a directed network, to a prescribed destination node, collecting rewards at each node visited, in order to maximize the total reward along the path. In our generalization, transit on arcs in the network and reward collection at nodes both consume a variable amount of the same limited resource. We exploit this resource trade-off through a specialized branch-and-bound algorithm that relies upon partial path relaxation problems which often yield tight bounds and lead to substantial pruning in the enumeration tree. We present the Smuggler Search Problem as an important real-world application of our generalized Orienteering Problem. Numerical results show that our algorithm applied to the Smuggler Search Problem outperforms standard Mixed-Integer Nonlinear Programming solvers for moderate to large problem instances. We demonstrate model enhancements that allow practitioners to represent realistic search planning scenarios by accounting for multiple heterogeneous searchers and complex smuggler motion.

Keywords: orienteering problem; military operations research; search and surveillance; route planning; mixed-integer nonlinear programming

1 Introduction

We define the Generalized Orienteering Problem with Resource Dependent Rewards (GOP-RDR), which seeks to route a vehicle along a simple path through a directed network, between prescribed start and end nodes, in order to maximize the total reward along the path. Rewards are collected by the vehicle at each node visited, where the reward level depends on the amount of scarce resources expended. Arcs in the network are traversed while consuming the same limited resources used for reward collection. The path is constructed so that the total resource expenditure is within given limits.

The GOP-RDR is a generalization of the well-known Orienteering Problem (OP) [33]. The OP and its multi-vehicle extension, the Team OP (TOP), have wide-ranging applicability and have been used to solve many practical problems in tourism [24, 25, 26, 36], sports [3, 11, 32], military operations [16, 20], commercial service and vehicle routing [11, 27, 30], and production [14, 18]. In these problems, node visitation rewards and arc traversal resource expenditures (arc lengths) are fixed quantities.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **19 FEB 2013**	2. REPORT TYPE	3. DATES COVERED **00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE **Generalized Orienteering Problem with Resource Dependent Rewards**	5a. CONTRACT NUMBER
	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
Naval Research Logistics, to appear

14. ABSTRACT
We introduce a generalized Orienteering Problem where, as usual, a vehicle is routed from a prescribed start node, through a directed network, to a prescribed destination node, collecting rewards at each node visited, in order to maximize the total reward along the path. In our generalization, transit on arcs in the network and reward collection at nodes both consume a variable amount of the same limited resource. We exploit this resource trade-o through a spe- cialized branch-and-bound algorithm that relies upon partial path relaxation problems which often yield tight bounds and lead to substantial pruning in the enumeration tree. We present the Smuggler Search Problem as an important real-world application of our generalized Orien- teering Problem. Numerical results show that our algorithm applied to the Smuggler Search Problem outperforms standard Mixed-Integer Nonlinear Programming solvers for moderate to large problem instances. We demonstrate model enhancements that allow practitioners to rep- resent realistic search planning scenarios by accounting for multiple heterogeneous searchers and complex smuggler motion.

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)**	**35**	

The GOP-RDR generalizes the OP by allowing node rewards and arc length to vary based on the amount of the resources expended at each node. GOP-RDRs arise in military, search and rescue applications, and law enforcement operations where the objective is to route a searcher to find moving targets in an area of interest (AOI) so that the total reward garnered by the search is maximized. In these *optimal search problems*, targets can be thought of as carrying some type of illicit material. Thus, the reward garnered by the search is related to the amount of illicit material detected. Limited resources (e.g., time, fuel, etc.) are expended by the searcher while performing search actions in regions of interest and while in transit between these regions as targets move in the AOI. We introduce the Smuggler Search Problem (SSP), a path-constrained optimal search problem in continuous space and time as an important example of a GOP-RDR. The SSP deals with the high level decision of routing search vehicles through subsets of the AOI called *search regions* in the presence of uncertain information about target whereabouts.

GOP-RDRs may also arise in commercial applications where a vehicle is routed to a number of locations in order to perform a service. The distance between locations can be represented by a travel time, possibly changing with time-of-day or environmental effects, and the reward garnered by performing the service at each location may be an increasing function of time spent at the location. For example, the Red Cross blood collection problem [38] is similar to the GOP-RDR in that it is more beneficial to visit pickup locations later in the path because the visitation reward increases with time. This problem differs from the GOP-RDR in that, while rewards at each node depend on time (time being a resource consumed in transit between nodes), the activity of collecting the reward does not require resource consumption. Moreover, the arc lengths between nodes do not vary in this problem.

Other generalizations of the OP have been considered in multi-objective problems where rewards may be functions of a number of attribute scores [24, 25, 35, 36], and where the arc length between nodes are determined by general cost functions [18]. The later reference introduces a generalized OP that arises when transit resources are not fixed values but are determined by general resource expenditure functions. The GOP-RDR further generalizes the OP by allowing reward collection to vary as a function of resources expended at nodes. We are not aware of any references in the literature on OPs and related problems which consider generalizations of the node rewards and arc lengths at the same time, nor have we encountered any problems where the effort of collecting rewards at nodes is in direct competition with that of transiting between nodes. The GOP-RDR appears to be the first to consider these issues.

Several exact algorithms for solving OPs and TOPs have been proposed in the literature: see [2, 4] for branch-and-price references or [15, 19] for branch-and-bound references. Laporte and Martello [15] describe a branch-and-bound algorithm where fathoming is accomplished by computing inexpensive upper bounds based on a binary knapsack problem. This is possible because the arc lengths and rewards are fixed values, conditions which do not necessarily hold in the GOP-RDR. Lagrangian relaxation is used within a branch-and-bound procedure by Ramesh at al. [19]. They relax the budget constraint and solve the resulting relaxation for fixed Lagrange multipliers using a polynomial time degree-constrained spanning tree algorithm, a technique that is not possible for the nonlinear GOP-RDR. We present a branch-and-bound algorithm for the GOP-RDR that capitalizes on the trade-off between transit and reward collection resource usage by solving partial path relaxation problems to compute upper bounds. Several heuristics for solving OPs and TOPs have also been proposed in the literature [1, 6]. We use the heuristic presented in [6] as a point of departure in developing a heuristic for the SSP, where a simple node deletion step is used to find an improving path.

The remainder of this article is organized as follows. We formulate the GOP-RDR and provide a branch-and-bound algorithm for obtaining solutions in the next section. In Section 3 we formulate the SSP and describe a heuristic that is used to provide initial solutions to the branch-and-bound (B&B) algorithm. In Section 4 we provide numerical results, comparing branch-and-bound solutions to solutions obtained by MINLP solvers. Section 5 highlights SSP enhancements that allow practitioners to model realistic search planning problems. We conclude with final remarks in Section 6.

2 Formulation and Branch-and-Bound Framework

Before formulating the GOP-RDR we begin with a standard OP formulation [33], which we will use as a stepping stone for generalization. Consider a standard OP, where a vehicle is routed through a transportation network, collecting rewards at each node. Let $G = (N, A)$ be the directed graph that models this transportation network, where $N = \{0, 1, \ldots, n+1\}$ is the node set and A is the arc set. Nodes 0 and $n+1$ are the vehicle's home station and recovery location respectively; not necessarily the same physical location. We assume that all arcs incident to node 0 are outbound arcs, and that all arcs incident to node $n+1$ are inbound arcs. For notational convenience, we define $\hat{N} = N \backslash \{0, n+1\}$ as the set of nodes excluding the home station, node 0, and the recovery location, node $n+1$. At each node $i \in \hat{N}$ it is possible to collect a reward q_i. Traversing any arc $(i,j) \in A$ consumes a fixed resource $\bar{t}_{i,j}$. Total resource expenditure is limited by T. We model the vehicle path on G using the binary variables $x_{i,j}$, where $x_{i,j}$ takes on value 1 when arc (i,j) is in the path, and 0 otherwise. We have the following OP formulation.

Problem O:

$$\max_{\boldsymbol{x}} \quad \sum_{j \in \hat{N}} q_j \left(\sum_{i:(i,j) \in A} x_{i,j} \right) \tag{1a}$$

$$\text{s.t.} \quad \sum_{(i,j) \in A} \bar{t}_{i,j} x_{i,j} \leq T \tag{1b}$$

$$\sum_{i:(i,j) \in A} x_{i,j} - \sum_{i:(j,i) \in A} x_{j,i} = \begin{cases} -1, & j = 0 \\ 0, & \forall j \in \hat{N} \\ 1, & j = n+1 \end{cases} \tag{1c}$$

$$\sum_{i:(i,j) \in A} x_{i,j} \leq 1, \qquad \forall j \in N \tag{1d}$$

$$\sum_{\substack{(i,j) \in A: \\ i,j \in N'}} x_{i,j} \leq |N'| - 1, \qquad \forall N' \subseteq \hat{N}, N' \neq \emptyset \tag{1e}$$

$$x_{i,j} \in \{0, 1\}, \qquad \forall (i,j) \in A \tag{1f}$$

The objective (1a) accumulates rewards along the path. Constraint (1b) ensures the resources $\bar{t}_{i,j}$ expended along the path do not exceed the resource limit T. Constraints (1c) maintain a balanced network flow that starts at the home station and ends at the recovery location. Constraints (1d) ensure nodes are visited at most once. Constraints (1e) are the subtour elimination constraints proposed in [8], which are known to yield relatively tight linear programming relaxations [29, chap. 1].

For notational convenience we will also use the auxiliary binary variable y_j, which is uniquely determined by variables $x_{i,j}$. Variables y_j take on value 1 when node j is in the path, and 0 otherwise; i.e.,

$$y_0 = 1, \qquad y_j = \sum_{i:(i,j)\in A} x_{i,j}, \forall j \in N\backslash\{0\} \tag{2}$$

We denote by $\boldsymbol{x}$ the vector of *path* variables $\{x_{i,j} : (i,j) \in A\}$. We denote by $\boldsymbol{y}$ the vector of *node visitation* variables $\{y_j : j \in N\}$ and represent (2) with the expression $\boldsymbol{y} = \boldsymbol{\Gamma x}$ for an appropriately selected matrix $\boldsymbol{\Gamma}$. We define $\mathbb{X}$ as the set of paths that satisfy (1c), (1d), (1e), and (1f).

We relax $\mathbf{O}$ to construct a GOP-RDR as follows. A visit to any node $i \in \{1, \ldots, n\}$ is rewarded at the expense of consuming r *dwell* resources $\boldsymbol{d}_i \in \Re^r$. Similarly, *transit* resources $\boldsymbol{t}_{i,j} \in \Re^r$ are consumed when traveling directly from node i to node j. Resources may represent, for example, various consumables such as time, fuel, and/or money. For notational convenience, we also include the auxiliary resource status variable $\boldsymbol{a}_i \in \Re^r$. This variable is used to track the accumulation of resources expended along the path. Let $\boldsymbol{a}, \boldsymbol{d} \in \Re^{r(n+2)}$, and $\boldsymbol{t} \in \Re^{r|A|}$ denote vectors of resource variables; for example $\boldsymbol{a} = (\boldsymbol{a}_0^T, \boldsymbol{a}_1^T, \ldots, \boldsymbol{a}_{n+1}^T)^T$. A vehicle may collect rewards according the utility function $f(\boldsymbol{d}) : \Re^{r(n+2)} \mapsto \Re$. We assume without loss of generality that no reward is possible at nodes 0 and $n+1$. We assume that f is a concave utility function, where $f(\mathbf{0}) = 0$. The vehicle path through G must obey η resource expenditure laws on each arc $(i,j) \in A$ denoted by the functions $h_{i,j}(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{d}_i, \boldsymbol{d}_j, \boldsymbol{t}_{i,j}) : \Re^{5r} \mapsto \Re^{\eta}$. The resource expenditure laws account for applications where arc lengths are allowed to vary as resources. In the static network considered in $\mathbf{O}$, $h_{i,j} = ||\bar{t}_{i,j} - t_{i,j}||$ for fixed arc lengths $\bar{t}_{i,j}$, but in Section 3 arc lengths are not fixed because the nodes of the network are in motion. The vehicle path must also obey resource expenditure laws at each node $i \in N$ denoted by the functions $g_i(\boldsymbol{a}_i, \boldsymbol{d}_i) : \Re^{2r} \mapsto \Re^{\gamma}$ and $m_i(\boldsymbol{a}_i, \boldsymbol{d}_i) : \Re^{2r} \mapsto \Re^{\mu}$. We assume that functions $h_{i,j}$ and g_j are convex, and functions m_j are affine. The vehicle path must be such that total resource expenditure stays within the resource limits defined by $\boldsymbol{T} \in \Re^r$. Let the matrix $\boldsymbol{Y} \in \Re^{r(n+2)\times r(n+2)}$ be the diagonal matrix $\text{diag}(y_0\mathbf{0}, y_1\mathbf{1}, y_2\mathbf{1}, \ldots, y_n\mathbf{1}, y_{n+1}\mathbf{0})$, where $\mathbf{0}, \mathbf{1} \in \Re^r$ are a vectors of 0s and 1s respectively. We note that the expression $\boldsymbol{Yd}$ simply returns the dwell resource vector associated with reward collection nodes in the path $\boldsymbol{x}$. We now state the GOP-RDR.

Problem P:

Sets

N	nodes: $i, j \in \{0, 1, \ldots, n+1\}$
A	arcs
$\mathbb{X}$	paths that satisfy (1c), (1d), (1e), and (1f)

Parameters

$\boldsymbol{T} \in \Re^r$	resource expenditure limits
$\boldsymbol{\Gamma} \in \Re^{(n+2)\times\|A\|}$	path-to-node visitation mapping matrix representing (2)

Functions

$f : \Re^{r(n+2)} \mapsto \Re$	concave reward collection objective function
$h_{i,j} : \Re^{5r} \mapsto \Re^{\eta}$	convex resource expenditure law functions
$g_{i,j} : \Re^{2r} \mapsto \Re^{\gamma}$	convex node resource expenditure law functions
$m_{i,j} : \Re^{2r} \mapsto \Re^{\mu}$	affine node resource expenditure law functions

Variables

$\boldsymbol{a}_i$	node i auxiliary resource variable
$\boldsymbol{d}_i$	node i dwell resource variable
$\boldsymbol{t}_{i,j}$	arc (i,j) transit resource variable
$x_{i,j}$	arc (i,j) binary path variable
y_i	node i binary visitation variable; $\boldsymbol{Y}$ is a diagonal matrix of these variables

Formulation

$$\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y}} \quad f(\boldsymbol{Y}\boldsymbol{d}) \tag{3a}$$

$$\text{s.t.} \quad h_{i,j}(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{d}_i, \boldsymbol{d}_j, \boldsymbol{t}_{i,j})x_{i,j} \le 0, \quad \forall (i,j) \in A \tag{3b}$$

$$\sum_{j\in N} \boldsymbol{d}_j + \sum_{(i,j)\in A} \boldsymbol{t}_{i,j} \le \boldsymbol{T} \tag{3c}$$

$$g_j(\boldsymbol{a}_j, \boldsymbol{d}_j) \le 0, \quad \forall j \in N \tag{3d}$$

$$m_j(\boldsymbol{a}_j, \boldsymbol{d}_j) = 0, \quad \forall j \in N \tag{3e}$$

$$\boldsymbol{a}_j, \boldsymbol{d}_j \ge 0, \quad \forall j \in N \tag{3f}$$

$$\boldsymbol{t}_{i,j} \ge 0, \quad \forall (i,j) \in A \tag{3g}$$

$$\boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x} \tag{3h}$$

$$\boldsymbol{x} \in \mathbb{X} \tag{3i}$$

The objective (3a) maximizes the reward collected along the path. Constraints (3b) enforce resource expenditure laws on each arc. Constraint (3c) ensures that total resource expenditure is within the prescribed limits. Constraints (3d) and (3e) enforce resource expenditure laws at each node. Concavity of f makes it desirable consume resources $\boldsymbol{d}$. Constraints (3c) make it undesirable consume resources $\boldsymbol{t}$. However, $\boldsymbol{d}$ and $\boldsymbol{t}$, along with $\boldsymbol{a}$, are related through constraints (3b) so it may not be possible to consume $\boldsymbol{t} = \boldsymbol{0}$ transit resources. Observe that when node j is not in the path $\boldsymbol{x}$, (3d) and (3e) are vacuous because $\boldsymbol{a}_j$ and $\boldsymbol{d}_j$ can be chosen arbitrarily to satisfy these constraints provided total resource expenditure (3c) is not exceeded. Constraint (3c) makes $\boldsymbol{d}_j = \boldsymbol{0}$ desirable in this situation because a higher reward is obtained by consuming more dwell resources at visited nodes. We assume that $\boldsymbol{d}_j = \boldsymbol{0}$ is always feasible. Similarly, when arc (i,j) is not in the path $\boldsymbol{x}$, constraint (3b) is inactive and the resource constraint (3c) forces $\boldsymbol{t}_{i,j} = \boldsymbol{0}$. Constraints (3f) and (3g) require nonnegative resource expenditure. Binary visitation and path variables are set by (3h) and (3i) respectively.

This MINLP as posed has a non-convex continuous relaxation. When this is the case most MINLP solvers such as DICOPT [12] or BONMIN [34] provide no guarantees of finding globally optimal solutions. The problem can be convexified with a Big-M reformulation. For example, (3b) can be reformulated as

$$h_{i,j}(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{d}_i, \boldsymbol{d}_j, \boldsymbol{t}_{i,j}) \leq M(1 - x_{i,j}), \forall (i,j) \in A,$$

for M sufficiently large. We discuss Big-M reformulation and show numerical results in the context of the SSP in the sequel. Another approach is to use a B&B-based MINLP solver such as BARON that uses convexifying techniques at each node of a B&B enumeration tree to obtain globally optimal solutions [22, 28]. This approach is not pursued here, rather we use the underlying structure of $\mathbf{P}$ as a basis for computing solutions.

With these matters in mind, we now proceed to describe a B&B approach that utilizes convex relaxation problems, avoids Big-M reformulations, and capitalizes on the underlying structure of $\mathbf{P}$ as a basis for branching and pruning. We introduce the notation $G(\boldsymbol{x}) = (N(\boldsymbol{x}), A(\boldsymbol{x}))$, where $N(\boldsymbol{x}) = \{j \in N : y_j = 1; y_0, = 1, \boldsymbol{y} = \boldsymbol{\Gamma x}\}$ and $A(\boldsymbol{x}) = \{(i,j) \in A : x_{i,j} = 1\}$. For any path $\boldsymbol{x} \in \mathbb{X}$, $\mathbf{P}$ can be expressed as the following convex NLP.

Problem P($\boldsymbol{x}$):

$$\begin{aligned}
\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t}} \quad & f(\boldsymbol{Y d}) && && \text{(4a)}\\
\text{s.t.} \quad & h_{i,j}(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{d}_i, \boldsymbol{d}_j, \boldsymbol{t}_{i,j}) \leq 0, && \forall (i,j) \in A(\boldsymbol{x}) && \text{(4b)}\\
& \sum_{j \in N(\boldsymbol{x})} \boldsymbol{d}_j + \sum_{(i,j) \in A(\boldsymbol{x})} \boldsymbol{t}_{i,j} \leq \boldsymbol{T} && && \text{(4c)}\\
& g_j(\boldsymbol{a}_j, \boldsymbol{d}_j) \leq 0, && \forall j \in N(\boldsymbol{x}) && \text{(4d)}\\
& m_j(\boldsymbol{a}_j, \boldsymbol{d}_j) = 0, && \forall j \in N(\boldsymbol{x}) && \text{(4e)}\\
& \boldsymbol{a}_j, \boldsymbol{d}_j \geq 0, && \forall j \in N(\boldsymbol{x}) && \text{(4f)}\\
& \boldsymbol{t}_{i,j} \geq 0, && \forall (i,j) \in A(\boldsymbol{x}) && \text{(4g)}
\end{aligned}$$

When $\boldsymbol{x}$ is fixed, $\boldsymbol{y}$ can be computed by (2). Variables $\boldsymbol{a}_j$ and $\boldsymbol{d}_j$ corresponding to unvisited nodes are removed from the problem. Similarly, variables $\boldsymbol{t}_{i,j}$ corresponding to arcs not traversed are eliminated. The resulting convex NLP in the remaining variables $\boldsymbol{a}, \boldsymbol{d}$, and $\boldsymbol{t}$ is efficiently solved by standard NLP solvers such as MINOS [17]. If we are able to enumerate all possible paths $\boldsymbol{x} \in \mathbb{X}$ and solve the associated fixed path NLP, we are assured to find the optimal solution to $\mathbf{P}$.

Observing that since $\boldsymbol{d}$ contributes to reward collection and $\boldsymbol{t}$ consumes resources without reward, nonzero values of $\boldsymbol{d}$ and small values of $\boldsymbol{t}$ are always desired. We construct the matrix $\tilde{\boldsymbol{I}}$ by taking an $r(n+2) \times r(n+2)$ identity matrix, and setting the first r diagonal entries and the last r diagonal entries to zero. The expression $\tilde{\boldsymbol{I}}\boldsymbol{d}$ returns the vector of dwell resources, setting home and recovery dwell resources to zero. We define $\boldsymbol{\delta}_{i,j} \in \Re^r$ as the smallest possible resource expenditure between node i and node j. If we consider $\mathbf{P}(\boldsymbol{x})$ and allow reward to be collected at every node with no transit resource expenditure on any arc, we obtain the following relaxed NLP.

Problem RP(0):

$$\begin{aligned}
\max_{\boldsymbol{a},\boldsymbol{d}} \quad & f(\tilde{\boldsymbol{I}}\boldsymbol{d}) && \text{(5a)}\\
\text{s.t.} \quad & \sum_{j \in N} \boldsymbol{d}_j \leq \boldsymbol{T} - \min_{\substack{j \in N: \\ (0,j) \in A}} \{\boldsymbol{\delta}_{0,j}\} - \min_{\substack{j \in N: \\ (j,n+1) \in A}} \{\boldsymbol{\delta}_{j,n+1}\} && \text{(5b)}\\
& \text{(3d), (3e), and (3f)}
\end{aligned}$$

Observe that the resource limit decrement on the right hand side of (5b)

$$\min_{\substack{j\in N:\\(0,j)\in A}} \{\boldsymbol{\delta}_{0,j}\} + \min_{\substack{j\in N:\\(j,n+1)\in A}} \{\boldsymbol{\delta}_{j,n+1}\} \tag{6}$$

is a lower bound on $\boldsymbol{t}_{0,j} + \boldsymbol{t}_{j,n+1}, \forall j \in \hat{N}$. A path that visits any nonempty subset of nodes $N' \subseteq \hat{N} : N' \neq \emptyset$ consumes at least (6) transit resources. $\mathbf{RP(0)}$ is clearly a relaxation of $\mathbf{P}$. $\mathbf{P}$ is obtained by adding constraints (3b), (3g), (3h), and (3i) to $\mathbf{RP(0)}$, while restricting reward collection (3a) to nodes in the path, and incurring a transit resource expenditure in (3c) that is no less than (6). Denoting the optimal objective function values of $\mathbf{P}$, $\mathbf{P}(\boldsymbol{x})$, and $\mathbf{RP(0)}$ by Z^*, $Z(\boldsymbol{x})^*$, and $Z(\mathbf{0})^*$, respectively, we state the following result.

Proposition 1. $Z(\mathbf{0})^* \geq Z^* \geq Z(\boldsymbol{x})^*, \forall \boldsymbol{x} \in \mathbb{X}$

Proof. The result follows from the fact that $\mathbf{RP(0)}$ is a relaxation of $\mathbf{P}$ and that $Z^* = \max_{\boldsymbol{x}\in\mathbb{X}} Z(\boldsymbol{x})^*$. □

In order to obtain useful bounds on $\mathbf{P}$, we introduce the notion of a partial path. We define a partial path $\hat{\boldsymbol{x}}_\ell$ to be the binary vector satisfying constraints (1d), (1e), and (1f), while constraints (1c) are satisfied for all nodes except the recovery location $n+1$ and the last node ℓ visited. We define the indicator parameter I_ℓ that takes on value 1 when $\ell = n+1$, and 0 otherwise.

For any partial path $\hat{\boldsymbol{x}}_\ell$ we have the following convex partial path relaxation NLP.

Problem RP($\hat{\boldsymbol{x}}_\ell$):

$$\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t}} \quad f(\tilde{\boldsymbol{I}}\boldsymbol{d}) \tag{7a}$$

$$\text{s.t.} \quad h_{i,j}(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{d}_i, \boldsymbol{d}_j, \boldsymbol{t}_{i,j}) \leq 0, \qquad \forall (i,j) \in A(\hat{\boldsymbol{x}}_\ell) \tag{7b}$$

$$\sum_{j\in N} \boldsymbol{d}_j + \sum_{(i,j)\in A(\hat{\boldsymbol{x}}_\ell)} \boldsymbol{t}_{i,j} \leq \boldsymbol{T} - (1 - I_\ell)\boldsymbol{\delta}_{\ell,n+1} \tag{7c}$$

$$I_\ell \boldsymbol{d}_j = 0, \qquad \forall j \in N \backslash N(\hat{\boldsymbol{x}}_\ell) \tag{7d}$$

$$\boldsymbol{t}_{i,j} \geq 0, \qquad \forall (i,j) \in A(\hat{\boldsymbol{x}}_\ell) \tag{7e}$$

$$\text{(3d), (3e), and (3f)}$$

We note that when $I_\ell = 1$, the path is complete, and $\mathbf{P}(\boldsymbol{x})$ and $\mathbf{RP}(\hat{\boldsymbol{x}}_\ell)$ are equivalent. Conversely, when $I_\ell = 0$, the path is a partial path, (7d) is inactive and resources associated with unvisited nodes are allowed to take on nonzero values, and the right hand side of (7c) is decremented by $\boldsymbol{\delta}_{\ell,n+1}$.

Consider extending the partial path $\hat{\boldsymbol{x}}_\ell$ by adding any arc $(\ell, k) \in A$ to the path as shown in Figure 1. In this depiction the minimum resource expenditure $\boldsymbol{\delta}_{\ell,n+1}$ in the partial path $\hat{\boldsymbol{x}}_\ell$ is no larger than the transit resource expenditure $\boldsymbol{t}_{\ell,k} + \boldsymbol{\delta}_{\ell,n+1}$ in the partial path $\hat{\boldsymbol{x}}_k$. Next we show a result that supports building successive restrictions of $\mathbf{RP}(\hat{\boldsymbol{x}}_\ell)$ by adding to the partial path. Let $Z(\hat{\boldsymbol{x}}_\ell)^*$ be the optimal objective function value of $\mathbf{RP}(\hat{\boldsymbol{x}}_\ell)$.

Proposition 2. $Z(\hat{\boldsymbol{x}}_\ell)^* \geq Z(\hat{\boldsymbol{x}}_k)^*, \forall k : (\ell, k) \in A$

Proof. Observe that if $\ell = n+1$, then $\{k : (\ell, k) \in A\} = \emptyset$. Suppose $\ell \in N \backslash \{n+1\}$. Adding node k and arc (ℓ, k) to the partial path adds a block of constraints to (7b) and variable $\boldsymbol{t}_{\ell,k}$ in (7c) and (7e). Since the increase in resource expenditure along the new partial path is at least $\boldsymbol{\delta}_{\ell,n+1}$, $\mathbf{RP}(\hat{\boldsymbol{x}}_k)$ is a restriction of $\mathbf{RP}(\hat{\boldsymbol{x}}_\ell)$ and the result follows. □

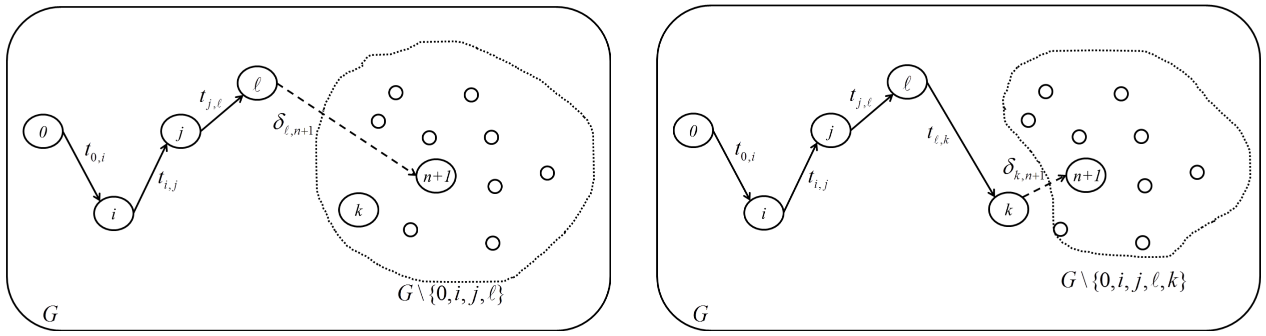

Figure 1: Partial path $\hat{\boldsymbol{x}}_\ell$ (left). Partial path $\hat{\boldsymbol{x}}_k$ (right). Transit resource expenditure along partial path $\hat{\boldsymbol{x}}_\ell$ is no larger than transit resource expenditure along path $\hat{\boldsymbol{x}}_k$.

We present a GOP-RDR B&B framework that begins at the home station and forms partial paths by adding nodes to a path sequentially, solving restrictions of $\mathbf{RP}(\hat{\boldsymbol{x}}_\ell)$ along the way. We denote the set $\hat{\boldsymbol{X}}$ as the set of all possible partial paths $\hat{\boldsymbol{x}}$. Let $l \in \{0, 1, \ldots, n+1\}$ denote the level of the B&B enumeration tree, and let $L_l \subseteq N$ be the set of nodes yet to be considered at level l. We define the set $\Omega \subseteq \hat{\boldsymbol{X}} \times N$ to be a subset of partial path and B&B enumeration tree level pairs. Let $\epsilon \geq 0$ be the absolute optimality gap stopping tolerance.

Algorithm B&B:

1. Initialize $\ell = 0$; $\boldsymbol{x}^* = \hat{\boldsymbol{x}}_\ell = \mathbf{0}$; lower bound $LB = 0$; $l = 0$; $L_0 = \emptyset$; $L_k = N, \forall k = 1, \ldots, n+1$; and $\Omega = \{(\mathbf{0}, 0)\}$. Solve $\mathbf{RP}(\mathbf{0})$. If $\mathbf{RP}(\mathbf{0})$ is infeasible, then stop; $\mathbf{P}$ is infeasible. Otherwise, initialize upper bound $UB = Z(\mathbf{0})^*$.
2. If $UB - LB < \epsilon$, then stop and return $\boldsymbol{x}^*$. Otherwise, choose $(\hat{\boldsymbol{x}}_\ell, l) \in \Omega$. Add node j to partial path $\hat{\boldsymbol{x}}_\ell$ to form the extended partial path $\hat{\boldsymbol{x}}_j \in \hat{\boldsymbol{X}}$ that contains arc (ℓ, j). Add $\{(\hat{\boldsymbol{x}}_j, l+1)\}$ to Ω. Remove j from L_{l+1}. Solve $\mathbf{RP}(\hat{\boldsymbol{x}}_j)$.
3. If $j = n+1$ and $LB < Z(\hat{\boldsymbol{x}}_j)^*$ and $\mathbf{RP}(\hat{\boldsymbol{x}}_j)$ is feasible, then set $LB = Z(\hat{\boldsymbol{x}}_j)^*$ and $\boldsymbol{x}^* = \hat{\boldsymbol{x}}_j$ as the best complete path found thus far.
4. If $L_{l+1} = \emptyset$ then set $UB = \max\{Z^*_{n+1}, Z^*_{l+1}\}$, where Z^*_{n+1} is the largest value $Z(\boldsymbol{x})^*$ of all complete paths explored thusfar and Z^*_{l+1} is the largest value $Z(\hat{\boldsymbol{x}}_j)^*$ on level $l+1$ of the B&B enumeration tree.
5. If $Z(\hat{\boldsymbol{x}}_j)^* < LB$ or if $\mathbf{RP}(\hat{\boldsymbol{x}}_j)$ is infeasible, then fathom partial path $\hat{\boldsymbol{x}}_j$ by removing from Ω all elements $(\hat{\boldsymbol{x}}_{\tilde{j}}, \tilde{l})$, where $\hat{\boldsymbol{x}}_j$ is a subpath of $\hat{\boldsymbol{x}}_{\tilde{j}}$ and $\tilde{l} > l$.
6. Return to step 2.

This algorithm can be accelerated by obtaining an initial feasible solution $\boldsymbol{x}$ that produces a better lower bound in step 1, thereby allowing fathoming in step 5 to occur more rapidly. To this end, we provide a specialized heuristic for the SSP in Section 3.3. We do not prescribe the nature of branching to be performed step 2. Numerical results discussed in Section 4 use depth-first-search, but other branching strategies can also be used (see, for example [19] or [29, ch. 2]).

3 The Smuggler Search Problem

The Smuggler Search Problem (SSP) is special case of a GOP-RDR that arises in challenging real-world search operations. This work is motivated by ongoing efforts to detect and interdict the flow of illicit traffickers in international waters. To accomplish this mission coalition forces strive to employ a limited number of search assets as effectively as possible, under strict resource constraints, as they respond to uncertain estimates of how illicit traffickers move in the AOI.

OPs and similar models have been used to solve path-constrained optimal search problems. Path-constrained optimal search problems are known to be NP-hard [31]. Many approaches which focus on discrete space and time models can be found in the literature [7, 10, 13, 21, 23]. We proceed to formulate the SSP, a novel path-constrained optimal search model in continuous space and time.

Consider a planning scenario where a search vehicle is to be routed throughout an AOI to detect multiple moving targets and within a D-hour long mission execution period. We assume that the targets are in linear motion with a constant speed, independent of the search effort. The search controller has, based on planning factors and intelligence estimates, the information listed in Table 1.

Maximum cruise speed of the searcher while in transit	V
Speed of the searcher while performing search actions	$\hat{V}$
Searcher sensor sweep width	W
Searcher endurance time limit	T
Scenario time limit	D
Number of targets	n
Speed of target j	U_j

Table 1: Known data available to the search controller

Suppose that there is uncertainty with regard to where and when each target departs, as well as the value of detecting the target, but the nature of the intelligence allows the search controller to estimate these values within some range of uncertainty. These data are listed in Table 2.

Expected departure time of target j	τ_j
Time uncertainty range of target j	$\tilde{\tau}_j$
Expected departure location of target j	$\boldsymbol{\rho}_j$
Expected arrival location of target j	$\bar{\boldsymbol{\rho}}_j$
Departure/arrival location uncertainty range of target j	$\tilde{\rho}_j$
Expected value of detecting target j	q_j

Table 2: Uncertainty data derived by the search controller

Based on these data, the latest departure time for each target j can be calculated as

$$\tau_j^{min} = \tau_j + \frac{1}{2}\tilde{\tau}_j.$$

Similarly, the earliest arrival time for each target j can be calculated as

$$\tau_j^{max} = \tau_j - \frac{1}{2}\tilde{\tau}_j + \frac{1}{U_j}||\boldsymbol{\rho}_j - \bar{\boldsymbol{\rho}}_j||.$$

Lastly, we can calculate the velocity vector of target j, $\boldsymbol{u}_j$ as a function of speed, and expected departure and arrival locations.

We model search within each target's region of uncertainty (search region) using a random search law with known sensor sweep width W; for details on random search models see [37, ch. 2]. We assume that search actions cannot be conducted for more than one target at the same time. This reflects the operational setting where the searcher is seeking out a specific target looking for characteristics outlined in intelligence reports. Thus, in the event that search regions overlap in space and time, the searcher cannot receive additional reward for searching for more than one target at a time. Given all the available information the search controller wishes to route the search vehicle through the AOI in order to maximize the expected value of the search effort. We model this as routing a vehicle across a transportation network $G = (N, A)$, where nodes are defined by the search regions and arcs are defined by the searchers transit between each pair of moving search regions.

3.1 Formulation

Since targets are in linear motion, the distance required to travel directly between each (i, j) pair of search regions can be computed as a function of time using Euclidean distance calculations. We proceed under the assumption that the path of the searcher is through the center of each search region. We also assume that the error between Euclidean distance and great circle distance is small relative to the size of the search regions. Suppose a searcher is searching for target in some predefined order and that the searcher has just completed searching region i. If a_i represents the time the searcher began searching region i and d_i represents the duration of the search in region i, then we can compute the current position $\boldsymbol{p}_i$ of the searcher as $\boldsymbol{p}_i = \boldsymbol{\rho}_i + (a_i + d_i - \tau_i)\boldsymbol{u}_i$. We assume that $a_i \geq \tau_i, \forall i$; the searcher will never arrive to search a target that has not departed. Suppose the searcher is next routed to region j, and that the transit time from region i to region j is denoted as $t_{i,j}$. The position of region j at the moment the searcher arrives is $\boldsymbol{p}_j = \boldsymbol{\rho}_j + (a_i + d_i + t_{i,j} - \tau_j)\boldsymbol{u}_j$. We can now relate the distance between region i and region j to the distance the searcher can travel in the same amount of time $||\boldsymbol{p}_i - \boldsymbol{p}_j|| = Vt_{i,j}$. We can relax this relationship by recognizing that the searcher does not always have to travel at maximum cruise speed. The searcher could choose to travel slower. Thus, $||\boldsymbol{p}_i - \boldsymbol{p}_j|| \leq Vt_{i,j}$, which is second-order cone constraint in the time resources $\boldsymbol{a}, \boldsymbol{d}$, and $\boldsymbol{t}$ of the form (3b).

Drawing from search theory [37, ch. 2] we define the detection rate in search region j, α_j, as

$$\alpha_j = \frac{W\hat{V}_j}{\tilde{\tau}_j \tilde{\rho}_j U_j}. \tag{8}$$

We assume that the searcher speed $\hat{V}$ is much greater than the target speeds $U_j, \forall j \in \hat{N}$. From the searcher's perspective, within each search region, the target is essentially stationary. It is possible to model the problem where this does not hold [37, sec. 6-1]. However, this assumption approximately holds in our SSP model, where we consider search aircraft and surface (e.g., boats) smugglers.

This problem can be formulated as the following MINLP, which is a special case of $\mathbf{P}$.

Problem SSP:

$$
\begin{aligned}
\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y}} \quad & \sum_{j\in\hat{N}} q_j\left(1-\exp\left\{-\alpha_j d_j y_j\right\}\right) && && \text{(9a)}\\
\text{s.t.} \quad & (||\boldsymbol{\rho}_i + (a_i+d_i-\tau_i)\boldsymbol{u}_i - \boldsymbol{\rho}_j - (a_i+d_i+t_{i,j}-\tau_j)\boldsymbol{u}_j|| \\
& \quad \ldots - Vt_{i,j})x_{i,j} \le 0, && \forall (i,j)\in A && \text{(9b)}\\
& (a_i+d_i+t_{i,j}-a_j)\,x_{i,j} \le 0, && \forall (i,j)\in A && \text{(9c)}\\
& \sum_{j\in\hat{N}} d_j + \sum_{(i,j)\in A} t_{i,j} \le T && && \text{(9d)}\\
& \sum_{j\in N} d_j + \sum_{(i,j)\in A} t_{i,j} \le D && && \text{(9e)}\\
& a_j \ge \tau_j^{min}, && \forall j\in N && \text{(9f)}\\
& a_j + d_j \le \tau_j^{max}, && \forall j\in N && \text{(9g)}\\
& a_0 = 0 && && \text{(9h)}\\
& d_{n+1} = 0 && && \text{(9i)}\\
& a_j, d_j \ge 0, && \forall j\in N && \text{(9j)}\\
& t_{i,j} \ge 0, && \forall (i,j)\in A && \text{(9k)}\\
& \boldsymbol{y} = \boldsymbol{\Gamma}\boldsymbol{x} && && \text{(9l)}\\
& \boldsymbol{x}\in\mathbb{X} && && \text{(9m)}
\end{aligned}
$$

The objective (9a) is to maximize the expected value of the search effort. (9b) ensures that the distance between search region i and search region j obeys Pythagorean's Theorem. Constraints (9c) propagate arrival times $\boldsymbol{a}$ forward in time as arcs are traversed. (9b) and (9c) correspond to (3b) in **P**. Constraints (9d) and (9e), corresponding to (3c) in **P**, ensure that the plan does not exceed resource limits T and D respectively. Note that the left summation in (9d) is over the set of nodes not including the home station and recovery location $\hat{N}$. This may appear to be inconsistent with (3c), however in **SSP** we could equivalently model two d_j terms for each node. One retains the correct dwell resources at all nodes, and the other is nearly a copy but consumes zero dwell resources at nodes 0 and $n+1$. We choose the more compact formulation here. Constraints (9f) require that the vehicle be routed to search regions only after the target has departed. Similarly, constraints (9g) preclude searching in a region after the target has arrived. (9h) and (9i) are included for completeness. They require respectively that time resources start at 0 and that the mission ends upon recovery. (9f), (9g), (9h) and (9i) correspond to (3d) in **P**. Constraints (9j), (9k), (9l), and (9m) are as discussed previously.

The dwell-to-transit resource trade-off that underlies **SSP** can be observed as follows. Consider the objective (9a), and constraints (9b) and (9d). Since the objective (9a) is monotonically increasing in $\boldsymbol{d}$, large dwell time is desirable. At the same time, since constraint (9d) limits the searcher's flying time, small values of $\boldsymbol{t}$ are desirable. However, due to constraints (9b), going from one search region to the next consumes transit time. This is the trade-off. Dwell time in a search region cannot be consumed without also consuming transit time in order to get to said search region.

We define an NLP analogous to $\mathbf{P}(\boldsymbol{x})$ which, for any path $\boldsymbol{x}\in\mathbb{X}$, provides the optimal time resource expenditure. We arrive at this problem by fixing $\boldsymbol{x}$ and $\boldsymbol{y}$, and retaining from **SSP** only the interesting constraints and objective function terms. Recall that $N(\boldsymbol{x}) = \{j\in N : y_j = 1; y_0, =$

$1, \boldsymbol{y} = \boldsymbol{\Gamma x}\}$ and $A(\boldsymbol{x}) = \{(i,j) \in A : x_{i,j} = 1\}$. Additionally, we define the set of search regions in the path $\hat{N}(\boldsymbol{x}) = N(\boldsymbol{x}) \backslash \{0, n+1\}$.

Problem SSP($\boldsymbol{x}$):

$$\begin{aligned}
\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t}} \quad & \sum_{j \in \hat{N}(\boldsymbol{x})} q_j \left(1 - \exp\{-\alpha_j d_j\}\right) & \\
\text{s.t.} \quad & ||\boldsymbol{\rho}_i + (a_i + d_i - \tau_i)\boldsymbol{u}_i - \boldsymbol{\rho}_j - (a_i + d_i + t_{i,j} - \tau_j)\boldsymbol{u}_j|| - V t_{i,j} \leq 0, & \forall (i,j) \in A(\boldsymbol{x}) \\
& a_i + d_i + t_{i,j} - a_j \leq 0, & \forall (i,j) \in A(\boldsymbol{x}) \\
& \sum_{j \in \hat{N}(\boldsymbol{x})} d_j + \sum_{(i,j) \in A(\boldsymbol{x})} t_{i,j} \leq T & \\
& \sum_{j \in N(\boldsymbol{x})} d_j + \sum_{(i,j) \in A(\boldsymbol{x})} t_{i,j} \leq D & \\
& a_j \geq \tau_j^{min}, & \forall j \in N(\boldsymbol{x}) \\
& a_j + d_j \leq \tau_j^{max}, & \forall j \in N(\boldsymbol{x}) \\
& a_j, d_j \geq 0, & \forall j \in N(\boldsymbol{x}) \\
& t_{i,j} \geq 0, & \forall (i,j) \in A(\boldsymbol{x}) \\
& \text{(9h) and (9i)} &
\end{aligned}$$

After a Big-M reformulation, **SSP** can be equivalently stated as an MINLP with a convex continuous relaxation. Let $M_{i,j}^R$ be a number that is always greater than the distance between region i and region j. Let $M_{i,j}^T$ be a number that is always greater than the time required for the searcher to travel between region i and region j. Let M_j^D be a number that is always greater than the search time in region j.

Problem SSPM

$$\begin{aligned}
\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y}} \quad & \sum_{j \in \hat{N}} q_j \left(1 - \exp\{-\alpha_j d_j\}\right) & & \text{(11a)} \\
\text{s.t.} \quad & ||\boldsymbol{\rho}_i + (a_i + d_i - \tau_i)\boldsymbol{u}_i - \boldsymbol{\rho}_j - (a_i + d_i + t_{i,j} - \tau_j)\boldsymbol{u}_j|| & & \\
& \qquad \ldots - V t_{i,j} \leq M_{i,j}^R (1 - x_{i,j}), & \forall (i,j) \in A & \quad \text{(11b)} \\
& a_i + d_i + t_{i,j} - a_j \leq M_{i,j}^T (1 - x_{i,j}), & \forall (i,j) \in A & \quad \text{(11c)} \\
& d_j \leq M_j^D y_j, & \forall j \in N & \quad \text{(11d)} \\
& \sum_{j \in \hat{N}} d_j + \sum_{(i,j) \in A} t_{i,j} \leq T & & \quad \text{(11e)} \\
& \sum_{j \in N} d_j + \sum_{(i,j) \in A} t_{i,j} \leq D & & \quad \text{(11f)} \\
& \text{(9f) through (9m)} & &
\end{aligned}$$

The nonlinear interactions between the binary variables and the continuous variables in **SSP** are modeled with a Big-M on the right hand sides of (11b), (11c) and (11d). Constraint (11d) requires that search duration be zero when the corresponding search region is not visited, which makes it possible to remove the nonlinear interactions in the objective function (3a), yielding (11a). It is well known that unnecessarily large Big-M values lead to poor continuous relaxations and ultimately slow down computation time [5]. In the case of the SSP, since target motion is linear, we can compute these values based on the maximum distance between each pair of targets.

3.2 Partial Path Relaxations

For any two target search regions i and j (home base possibly being one of them), the following convex NLP produces the minimum travel distance between them.

Problem D

$$\begin{aligned} \delta^*_{i,j} = \min_t \quad & ||(\boldsymbol{\rho}_i + (t-\tau_i)\boldsymbol{u}_i) - (\boldsymbol{\rho}_j + (t-\tau_j)\boldsymbol{u}_j)|| \\ \text{s.t.} \quad & \max\{\tau_i^{min}, \tau_j^{min}\} \le t \le \min\{\tau_i^{max}, \tau_j^{max}\} \end{aligned}$$

We let $\delta_{j,n+1} \equiv V^{-1}\delta^*_{j,n+1}$ be the minimum travel time resource expenditure between search region j and the recovery location, as described in Section 2. We proceed under the assumption that the home station and the recovery location are the same physical location, therefore $\delta_{0,j} = \delta_{j,n+1} = V^{-1}\delta^*_{0,j} = V^{-1}\delta^*_{j,n+1}$. This is usually the case in the type search planning problems we consider and it imposes no limitations on our model or solution procedures. The SSP instance of $\mathbf{RP(0)}$ is obtained when no path $\boldsymbol{x}$ is specified. We force $\boldsymbol{a}$ to take on lower bound values in order to allow $\boldsymbol{d}$ to take on highest possible values. When this is done $\boldsymbol{a}$ and $\boldsymbol{t}$ can be eliminated from the problem, resulting in the following NLP in the search time $\boldsymbol{d}$.

Problem RSSP(0)

$$\begin{aligned} \max_{\boldsymbol{d}} \quad & \sum_{j\in\hat{N}} q_j \left(1 - \exp\{-\alpha_j d_j\}\right) \\ \text{s.t.} \quad & d_j \le \tau_j^{max} - \tau_j^{min}, & \forall j \in N \\ & \sum_{j\in\hat{N}} d_j \le T - 2\min_{j\in N}\{\delta_{j,n+1}\} \\ & \sum_{j\in N} d_j \le D - 2\min_{j\in N}\{\delta_{j,n+1}\} \\ & d_{n+1} = 0 \\ & d_j \ge 0 & \forall j \in N \end{aligned}$$

For any partial path $\hat{\boldsymbol{x}}_\ell$, we have the following relaxed NLP as a special case of $\mathbf{RP}(\hat{\boldsymbol{x}}_\ell)$.

Problem RSSP($\hat{\boldsymbol{x}}_\ell$)

$$\begin{aligned}
\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t}} \quad & \sum_{j\in\hat{N}} q_j\left(1-\exp\{-\alpha_j d_j\}\right) & \\
\text{s.t.} \quad & (||\boldsymbol{\rho}_i + (a_i+d_i-\tau_i)\boldsymbol{u}_i - \boldsymbol{\rho}_j - (a_i+d_i+t_{i,j}-\tau_j)\boldsymbol{u}_j|| - Vt_{i,j}) \le 0, & \forall (i,j)\in A(\hat{\boldsymbol{x}}_\ell) \\
& (a_i+d_i+t_{i,j}-a_j) \le 0, & \forall (i,j)\in A(\hat{\boldsymbol{x}}_\ell) \\
& \sum_{j\in\hat{N}} d_j + \sum_{(i,j)\in A(\hat{\boldsymbol{x}}_\ell)} t_{i,j} \le T - (1-I_\ell)\delta_{\ell,n+1} & \\
& \sum_{j\in N} d_j + \sum_{(i,j)\in A(\hat{\boldsymbol{x}}_\ell)} t_{i,j} \le D - (1-I_\ell)\delta_{\ell,n+1} & \\
& I_\ell d_j = 0, & \forall j \in N\backslash N(\hat{\boldsymbol{x}}_\ell) \\
& t_{i,j} \ge 0, & \forall (i,j)\in A(\hat{\boldsymbol{x}}_\ell) \\
& \text{(9f) through (9j)} &
\end{aligned}$$

SSP can be solved by Algorithm B&B using **RSSP**($\hat{\boldsymbol{x}}_\ell$) relaxations and the lower bound initialization heuristic described next.

3.3 SSP Heuristic

Denote the optimal solution to **SSP** as Z^*, we observe that if $\epsilon = 0$ and the initial guess $\boldsymbol{x}$ is provided to Algorithm B&B where $Z(\boldsymbol{x})^* = Z^*$, then the number of NLP solutions required to prove $\boldsymbol{x} = \boldsymbol{x}^*$ is constant regardless of how branching is done in step 2. This is a direct consequence of the fact that fathoming only depends on the lower bound. Of course this observation is not unique to our problem setting. In fact, it is true of any branch-and-bound algorithm provided the algorithm does not include more sophisticated fathoming rules. This observation is the impetus to develop a reliable way of providing initial solutions to Algorithm B&B, possibly eliminating the need for complex branching strategies.

In order to provide a good initial solution to Algorithm B&B, we propose a five-phase heuristic that relies on the knowledge that solving a TOP-RDR entails finding an acceptable balance between dwell and transit resource expenditure. Ramesh and Brown [18] outline a four-phase heuristic for the GOP using a *bang-for-buck* ratio that relates the reward at each node to the bounds on transit time. We use a similar idea here, however since rewards and transit times are generally not known quantities, we consider a bang-for-buck ratio that relates expected search value to the area of the search region. We also add considerations for transit time by *clustering* targets based on temporal and spatial proximity. Throughout, we use **SSP**($\boldsymbol{x}$) to quickly determine the value of search plan at each iteration in the heuristic.

Phase I of the SSP heuristic initializes parameters and solves **RSSP**(0) in order to rule out targets that have low search value. In Phase II, remaining targets are grouped into spatial clusters based on geographical region, then, using a clustering parameter Δ, further grouped into clusters based on their earliest arrival times τ_j^{max}. In our SSP application, the searcher's home station is generally closer to target arrival locations than departure locations. Therefore search is more likely to take place at the end of the target's movement track than it is at the beginning. Earlier target arrivals represent search opportunities that are eliminated earliest in the scenario. Thus, earlier arriving targets would likely be searched first if they are searched at all. Targets are then ordered within each cluster and entire clusters are ordered with respected to one another. Phase III performs a feasibility check and, if necessary, systematically removes the lowest value targets (of those remaining) from consideration in order to find an initial feasible solution. In Phase IV, pairs of consecutively ordered targets that belong to different clusters are examined in a cluster seam refinement process. Finally, all remaining targets are considered for removal in Phase V. Further details of the SSP heuristic are provided in Appendix A.

In the worst case the SSP heuristic requires solving $2n+2$ NLPs. This occurs when a feasible path is found in Phase II, and all targets are considered for removal in Phases IV and V. In this situation each target occupies its own cluster. This can be prevented in well posed problem instances where the clustering parameter Δ is chosen appropriately with respect to the arrival times τ_j^{max}. Since the NLP subproblems can be solved quickly, approximately 1/10 of a second for large problems, the heuristic is quite fast even in the worst case.

4 Numerical Results

We consider **SSP** where smugglers move in a northwesterly direction as they attempt to transport illicit material from the south. In this scenario, we assume smuggler movement occurs through corridors defined by coastal strips of likely departure and arrival locations as depicted by the abstract map shown in Figure 2. Observe that, even though the dotted lines intersect on the spatial map, target search regions rarely overlap in space and time.

We assume searcher and target performance data that is consistent with known planning factors for P-3 aircraft and GO FAST smuggler boats. We assume a 24-hour mission execution period (D). Departure time uncertainty data is randomly generated within the mission planning period with uncertainty ranging up to four hours. Location uncertainty data is randomly generated where smugglers are equally likely to depart and arrive anywhere on the aforementioned coastal strips. Expected value of detecting each target is randomly generated within the range $[500, 5000]$ lbs, corresponding to estimated payload capacity of GO FAST boats.

We solve 100 randomly generated problem instances with 3, 5, 7, 8, and 10 smugglers and compare model performance using Algorithm B&B with heuristic initialization applied to **SSP** versus solving **SSPM** directly using two MINLP solvers. For each set of problem instances we deem the *best solver* to be the one that identifies the optimal solution in the shortest amount of average computing time. For the purposes of this numerical experiment, Algorithm B&B is implemented with a depth-first-search strategy and the optimality tolerance is set to zero, $\epsilon = 0$. The SSP heuristic is implemented with temporal clustering parameter $\Delta = 6$ hours and spatial clustering corresponding to the southwest (bottom-left) and northeast (top-right) regions shown in Figure 2. All computations are done on a 64-bit Windows 7 desktop computer (2x Intel Xeon 3.46GHz; RAM 24GB) using GAMS 23.8. We use MINOS to solve all NLP subproblems. In initial testing, DICOPT and BONMIN with ECP solver option appeared to be the most effective GAMS-based solvers for directly solving **SSPM**. Accordingly we limit our MINLP numerical results to these two solvers. For brevity, we refer to Algorithm B&B with heuristic initialization as B&B and BONMIN with ECP solver option as BONMIN(ECP). For the remainder of this paper, unless otherwise stated, searcher and target data is presented in nautical miles, nautical miles per hour, hours, and pounds; computation runtimes are given in seconds; and optimality gaps are reported as a percent difference from the optimal objective function value.

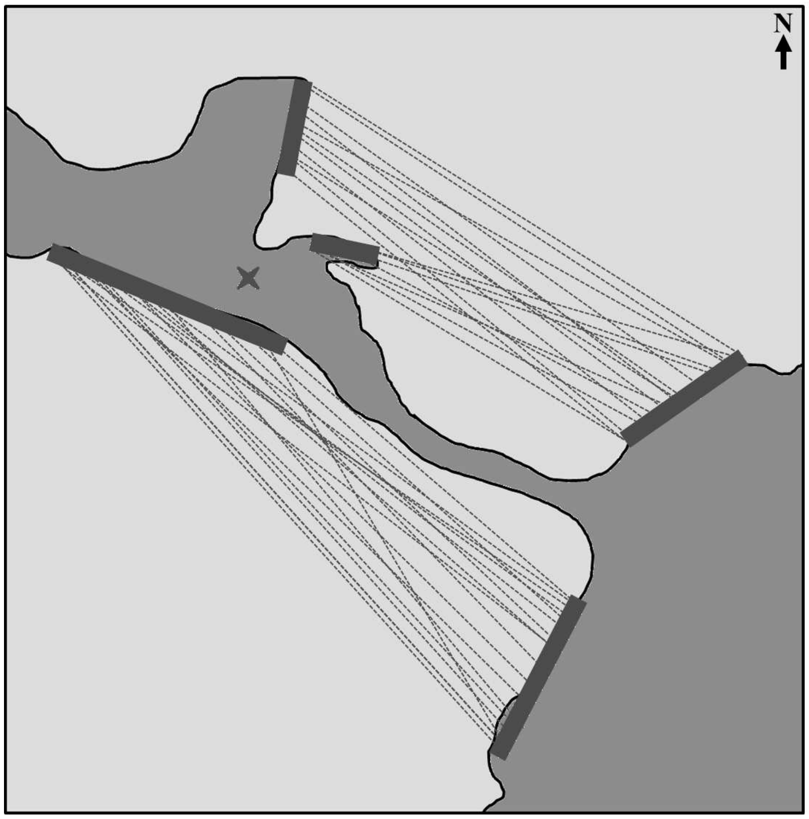

Figure 2: Target movement tracks are randomly generated with origin and destination points chosen within coastal strips marked by thick solid bars. Given an origin and a destination, the target track goes along a straight line within corridors indicated by dotted lines in a northwesterly direction. The searcher's home station is identified by "$\times$". Dotted lines shown illustrate possible target movement tracks. Randomly generated target movement tracks are not limited to those depicted here, but stay within the envelope boundaries.

As indicated in Table 3, all three solvers are able to solve all 100 of the 5-target SSPs to optimality within 13 seconds computing time. For these problem instances BONMIN(ECP) is the best solver, while B&B yields the slowest average runtime.

We use *performance profiles* [9] as a method for comparing solver runtimes. Performance profiles require two components: *performance ratios* and *performance metrics*. A performance ratio is a ratio of the runtime for solver s on problem p to the fastest runtime for all solvers tested on problem p. A performance metric is the empirical probability, across all problems p, that the runtime for solver s is within a factor of k of the fastest solver runtime. A performance profile is a distribution function of the performance metric over factors k.

We see in Figure 3 that BONMIN(ECP) preforms well on 5-target SSPs, with the fastest runtime for nearly 90% of these problems. DICOPT runtimes stay within a factor of 3 of the fastest runtime for 90% of problems. B&B lags behind the MINLP solvers, with runtimes within a factor of 7 of the fastest runtimes for approximately 80% of problems. All of the problem instances being examined here are solved in 13 seconds or less. On a relative (performance profile) scale BONMIN(ECP) seems to be the clear winner, but all of these solvers would be acceptable to planners in a practical sense.

	B&B	BONMIN(ECP)	DICOPT
Num Solved	100	100	100
Runtime (sec)			
Average	5.77	1.25	2.03
St Dev	2.52	0.30	0.89
St Error	0.25	0.03	0.09
Median	5.44	1.22	1.92
90th Percentile	9.21	1.70	3.08
Min	1.54	0.61	0.61
Max	13.00	1.98	4.99

Table 3: Runtime summary for 5-target SSPs. Num Solved refers to the number of problems out of 100 that were solved to optimality within 30 minutes. BONMIN(ECP) dominates in all metric categories and B&B yields the slowest average runtime.

In the 7 to 8-target range the relative performance of these solvers changes dramatically. Table 4 and the performance profile plot, Figure 4, highlight that runtimes of BONMIN(ECP) and B&B on 7-target SSP instances are nearly identical. DICOPT yields the slowest runtimes of the three solvers tested.

	B&B	BONMIN(ECP)	DICOPT
Num Solved	100	100	100
Runtime (sec)			
Average	32.84	30.47	115.39
St Dev	20.48	16.31	97.76
St Error	2.05	1.63	9.78
Median	27.82	26.19	86.20
90th Percentile	53.06	52.99	240.25
Min	8.77	8.02	15.90
Max	141.07	81.96	618.70

Table 4: Runtime summary for 7-target SSPs. BONMIN(ECP) and B&B have nearly identical runtimes. DICOPT yields the slowest runtimes of the three solvers tested.

Table 5 shows that B&B is the best solver for the larger 8-target problem instances. B&B runtimes are, on average, 90 seconds (1.5 minutes), while BONMIN(ECP) runtimes are much larger at 267 seconds (4.5 minutes). Observe that since the limiting distribution of the sample mean is normal, and considering the standard error of the average runtimes, we can say with high ($> 99\%$) confidence that the true mean runtimes for B&B on all 8-target problems in this sample space are faster than that of the other two solvers.

The performance profile plot (Figure 5) demonstrates that runtimes for BONMIN(ECP) and DICOPT in nearly all problem instances are several times larger than that of B&B, with over half of the probability mass for BONMIN(ECP) being in the $k = 3$ to $k = 9$ range.

When considering larger, 10-target, SSPs it is clear that B&B is the only viable algorithm among

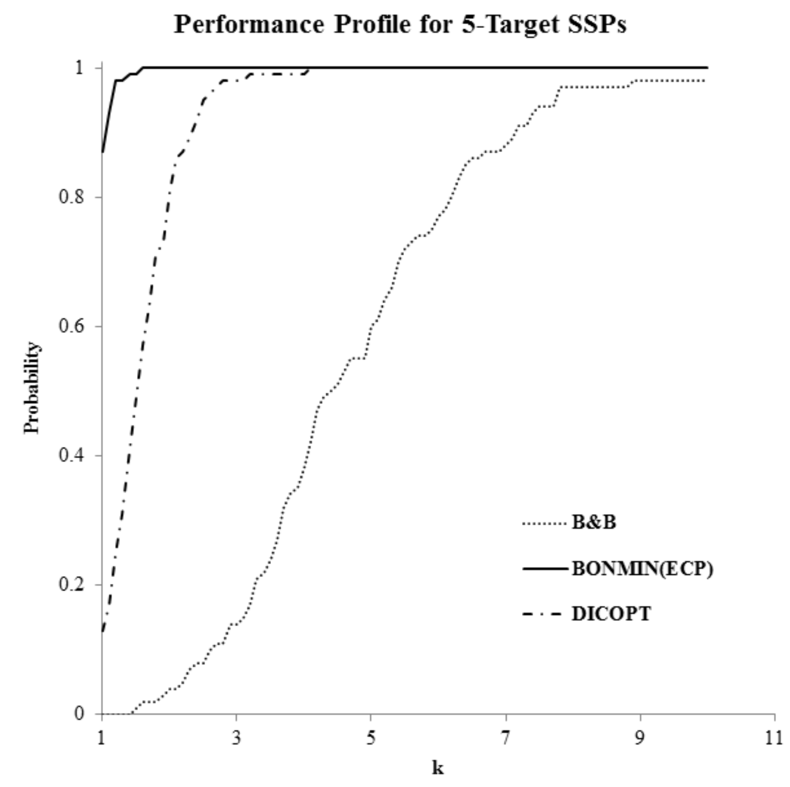

Figure 3: Performance profile for 5-target SSPs. BONMIN(ECP) preforms well on 5-target SSPs, with the fastest runtime for nearly 90% of these problems. B&B lags behind the MINLP solvers, with runtimes within a factor of 7 of the fastest runtimes for approximately 80% of problems.

the three solvers tested. Table 6 highlights that B&B is able to solve 97 out of 100 problem instances within 30 minutes of computing time. The average runtime is 8.5 minutes and 90% of the problems are solved within 17 minutes of computing time. Meanwhile BONMIN(ECP) and DICOPT are unable to solve any of the SSP test problems within 30 minutes.

We now quantify *how far* the BONMIN(ECP) and DICOPT solutions are from the 10-target SSP optimal solutions by looking at the reported optimality gaps upon termination. Table 7 highlights that when BONMIN(ECP) and DICOPT terminate upon reaching the 30 minute time limit, the solution available is usually far from optimal. On average solutions are off by a factor of at least 3.5 (optimality gaps in excess of 350%). For over half of the problems tested, DICOPT is unable to provide a bound on the optimal solution because the initial MIP for the linearized subproblem is not solved within 30 minutes.

The trend continues for larger problems. On a set of 25 randomly generated 15-target problem instances B&B solves each problem instance to optimality in 4667.77 seconds (1.30 hours) on average, solving 18 out of 25 problem instances within 2 hours. BONMIN(ECP) is unable to solve any of these problem instances within 2 hours, terminating with an average optimality gap of 949%.

We are able to gain some insight into why B&B outperforms the MINLP solvers as the problem size increases by examining the branch-and-bound enumeration tree of a representative problem instance. We consider a 10-target SSP instance that is solved in 324 seconds, near the median runtime. In order to isolate the efficiency gained by the heuristic, we solve this problem with no initial solution provided as well as with heuristic initialization. We note that a 10-target SSP results

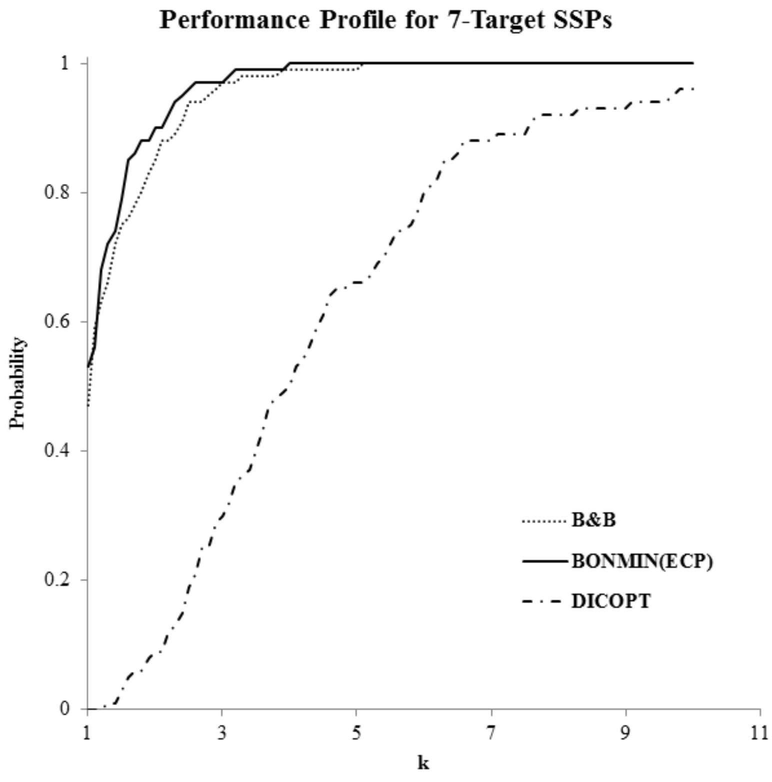

Figure 4: Performance profile for 7-target SSPs. BONMIN(ECP) and B&B each have the best runtimes in approximately half of the test problems. DICOPT yields the slowest runtimes of the three solvers tested.

in an enumeration tree of nearly 20 million nodes, spanning 12 levels deep. At each node we solve $\mathbf{RSSP}(\hat{\boldsymbol{x}}_\ell)$. Clearly, Algorithm B&B visits only a small fraction of these nodes due to pruning. Any path through the tree that has length 12 is a path that visits all target search regions. We use the term *perceived depth* to refer to the depth of visited nodes in the enumeration tree. If the average perceived depth of the tree were large, it would be tantamount to enumerating all possible paths $\boldsymbol{x} \in \boldsymbol{X}$. Thus, in order for our B&B algorithm to perform efficiently we need that the perceived depth of the tree remain relatively small for large problems. This is possible due to the dwell-to-transit resource trade-off that takes place when we consider extensions to partial paths in the enumeration tree. Table 8 shows the number of nodes at each level of the tree for a representative problem instance with and without heuristic initialization. We see that the tree is explored no more than 9 levels deep, as the partial path relaxation provided by $\mathbf{RSSP}(\hat{\boldsymbol{x}}_\ell)$ encounters the optimal solution bound fairly shallow in the tree. The majority of the nodes are visited in levels 5-7 of the tree. Considering that a 5-target SSP, solvable in only a few seconds, yields an enumeration tree that is 7 nodes deep, the perceived depth of the tree for larger 10-target SSP is shallow relative to its problem size.

Table 8 also shows the benefit of using the SSP heuristic to determine the initial guess for B&B. Runtime increases proportionally to the number of required NLP solutions in the enumeration tree. Having a good bound on the optimal solution reduces the total number of required NLP solutions by a factor of 2.3.

We conclude this section with some remarks on the performance of the heuristic presented in

	B&B	BONMIN(ECP)	DICOPT
Num Solved	100	100	82
Runtime (sec)			
Average	90.01	267.82	876.74
St Dev	72.13	185.82	578.37
St Error	7.21	18.58	57.84
Median	67.38	191.67	737.59
90th Percentile	181.18	500.62	1800
Min	8.74	65.58	57.69
Max	415.33	946.74	1800

Table 5: Runtime summary for 8-target SSPs. B&B is the best solver for these problem instances. B&B runtimes are, on average, 90 seconds (1.5 minutes), while BONMIN(ECP) runtimes are much larger at 267 seconds (4.5 minutes)

	B&B	BONMIN(ECP)	DICOPT
Num Solved	97	0	0
Runtime (sec)			
Average	515.91	-	-
St Dev	697.87	-	-
St Error	69.79	-	-
Median	313.06	-	-
90th Percentile	1006.96	-	-
Min	33.27	-	-
Max	5048.39	-	-

Table 6: Runtime summary for 10-target SSPs. B&B is the only viable solver for these problems. BONMIN(ECP) and DICOPT are unable to solve any of these problems within 30 minutes of computing time.

Section 3.3 with respect to the 3, 5, 7, 8, and 10-target SSP test set. The heuristic correctly identifies 223 optimal solutions out of 500 total SSPs tested. Table 9 shows that the accuracy of the heuristic tends to diminish as the number of targets increases. However, the average relative optimality gap remains within the 1-3% range throughout. Therefore while the accuracy rate in finding the optimal solution decreases, the heuristic does not *miss* by too wide of a margin on average. The heuristic is able to get within 7% of optimality in at least 90% of all problems tested. In all cases, the average accuracy rate is driven by one to three poor performing problem instances. While the average accuracy diminishes, the runtime remains fairly constant. It is at or below half a second for all problems tested. This is consistent with the worst case runtime analysis presented in Section 3.3. Observe that, comparing Table 9 to Table 6, on average the heuristic's 90th percentile optimality gaps for 10-target problem instances are smaller than the optimality gaps for BONMIN(ECP) and DICOPT. A problem-by-problem comparison reveals that the heuristic solutions have a smaller optimality gap in all problem instances.

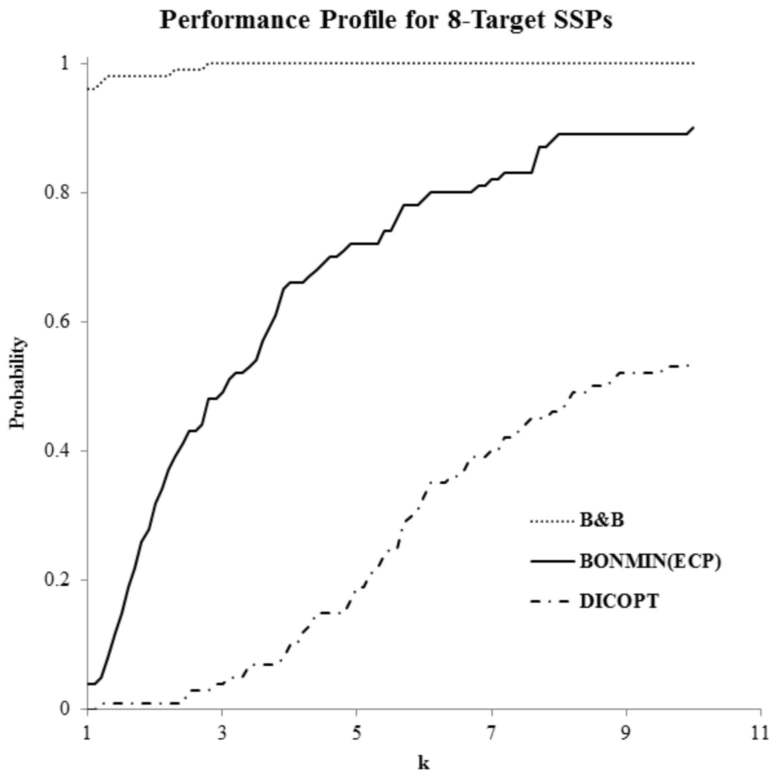

Figure 5: Performance profile for 8-target SSPs. B&B yields the fastest runtimes for nearly all problem instances. BONMIN(ECP) and DICOPT runtimes are at least three times larger than that of B&B for over 60% of problems tested.

5 Smuggler Search Problem Model Enhancements

SSP serves as a baseline model for solving many interesting problems that arise in search planning. Real-world search planning scenarios typically involve developing search plans for multiple searchers, while accounting for complicated smuggler movement tracks. We show how our model can be enhanced to capture these difficult planning problems.

5.1 Multiple Searchers

We consider a search planning operation where $s \in S$ searchers are available. We model this planning problem as a GOP-RDR on a searcher-expanded network $G_S = (N_S, A_S)$, where the nodes are searcher-target pairs $N_S = \{(s,j) : s \in S, j \in N\}$ and the arcs $A_S = \{(s,i,j) : s \in S, (i,j) \in A\}$ represent the transit of searcher s between search region i and search region j. Utilizing the vector forms of $\boldsymbol{a}_j, \boldsymbol{d}_j$ and $\boldsymbol{t}_{i,j}$ in $\mathbf{P}$, we allow each of these resource variables to have $|S|$ elements. We denote by $a_{s,j}$ and $d_{s,j}$ the respective arrival time and dwell time of searcher s in search region j, and we denote by $t_{s,i,j}$ the transit time of searcher s from search region i to search region j. The multiple searcher SSP is stated as follows.

	BONMIN(ECP)	DICOPT
Num Bounds	100	44
Gap (%)		
Average	504	358
St Dev	323	223
Median	433	316
10th Percentile	236	126
90th Percentile	903	551
Min	74	17
Max	1893	1182

Table 7: Optimality gap summary for BONMIN(ECP) and DICOPT on 10-target SSPs. Num Bounds refers to the number of problem instances out of 100 for which the respective solver provided a bound on the objective function value within 30 minutes of computing time. Reported solutions generally differ from the optimal solutions by a large margin, upwards of 350% for both solvers. DICOPT is unable to report an optimality gap within 30 minutes for 56 problem instances.

Problem SSP-MS:

$$\max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y}} \quad \sum_{j\in\hat{N}} q_j \left(1 - \exp\left\{-\sum_{s\in S}\alpha_{s,j} d_{s,j} y_{s,j}\right\}\right) \tag{15a}$$

$$\begin{aligned}
\text{s.t.} \quad & (||\boldsymbol{\rho}_{s,i} + (a_{s,i} + d_{s,i} - \tau_{s,i})\boldsymbol{u}_{s,i} - \boldsymbol{\rho}_{s,j} \\
& \quad \ldots - (a_{s,i} + d_{s,i} + t_{s,i,j} - \tau_{s,j})\boldsymbol{u}_{s,j}|| \\
& \quad\quad \ldots - V t_{s,i,j}) x_{s,i,j} \leq 0, && \forall (s,i,j) \in A_S && (15b)\\
& (a_{s,i} + d_{s,i} + t_{s,i,j} - a_{s,j})\, x_{s,i,j} \leq 0, && \forall (s,i,j) \in A_S && (15c)\\
& \sum_{j\in\hat{N}_S} d_{s,j} + \sum_{(i,j)\in A_S} t_{s,i,j} \leq T_s, && \forall s \in S && (15d)\\
& \sum_{j\in N_S} d_{s,j} + \sum_{(i,j)\in A_S} t_{s,i,j} \leq D_s, && \forall s \in S && (15e)\\
& a_{s,j} \geq \tau^{min}_{s,j}, && \forall (s,j) \in N_S && (15f)\\
& a_{s,j} + d_{s,j} \leq \tau^{max}_{s,j}, && \forall (s,j) \in N_S && (15g)\\
& a_{s,0} = 0, && \forall s \in S && (15h)\\
& d_{s,n+1} = 0, && \forall s \in S && (15i)\\
& a_{s,j}, d_{s,j} \geq 0, && \forall (s,j) \in N_S && (15j)\\
& t_{s,i,j} \geq 0, && \forall (s,i,j) \in A_S && (15k)\\
& \boldsymbol{y}_s = \boldsymbol{\Gamma}\boldsymbol{x}_s, && \forall s \in S && (15l)\\
& \boldsymbol{x}_s \in \mathbb{X}, && \forall s \in S && (15m)
\end{aligned}$$

Each expression in **SSP-MS** is a direct extension of its **SSP** counterpart where $a_{s,j}, d_{s,j}$ and $t_{s,i,j}$ are the arrival time, dwell time, and the transit time of searcher s contained in the vectors $\boldsymbol{a}_j, \boldsymbol{d}_j$ and $\boldsymbol{t}_{i,j}$ respectively. We allow resources T and D in (15d) and (15e) respectively to vary by searcher. This is a useful feature that allows the model to account for heterogeneous searchers. In

Level	With Heuristic NLP Solutions	Without Heuristic NLP Solutions
1	1	1
2	10	10
3	100	100
4	657	702
5	2008	3024
6	2051	5061
7	714	3354
8	85	720
9	0	40
Total	5626	13012
Runtime (sec)	324.78	676.69

Table 8: Number of NLP solutions on each level of the B&B tree for a representative 10-target SSP instance. The perceived depth of the enumeration tree is shallow relative to its problem size, highlighting the resource trade-off that motivates the GOP-RDR and highlights the benefit of the Algorithm B&B.

the objective function (15a), each exponential term associated with search region j in the random search model computes detection probability by accumulating total search effort for all searchers.

Algorithm B&B can be used to solve the multiple searcher SSP. We modify the notation in Algorithm B&B by requiring that the nodes of the enumeration tree be viewed as (s, j)-pairs, where $s \in S, j \in N$. We also vectorize I_ℓ and $\boldsymbol{\delta}_{\ell,n+1}$ in $\mathbf{RP}(\hat{\boldsymbol{x}}_\ell)$ to account for path completion with respect to each searcher, and modify the path completion criterion in step 3 to require that $(s, j) = (s, n + 1), \forall s \in S$. In principle, this can be done for an arbitrary number of searchers, however the size of the enumeration tree is exponential in the number of searchers $|S|$. Fortunately, real-world applications we consider call for search planning with a very limited number of searchers. Thus, for operational and computational reasons we limit our numerical results to 2-searcher SSPs.

In order to provide a good initial guess to Algorithm B&B, accounting for 2 searchers, we perform our SSP heuristic sequentially as follows. We set the temporal clustering parameter $\Delta = 6$ hours. We initialize the path for searcher 2 to consist only of arc $(0, n + 1)$. This ensures a feasible, but certainly not optimal, path for searcher 2. We then run the (single-searcher) SSP heuristic for searcher 1 and fix the resulting path. We then do the same for searcher 2. We improve the search plan by considering the removal of targets from searcher 1's path, performing Phase V of the SSP heuristic. We then do the same for searcher 2. Lastly, we attempt to improve the plan by allowing the searchers to swap their entire search paths. The modified heuristic returns the best search plan encountered after the aforementioned steps are completed. We do not consider pairwise target swaps between searchers nor do we consider a parallel implementation of the SSP heuristic. Our aim is to quickly provide a good initial solution to Algorithm B&B. While we expect that a parallel heuristic with pairwise target swapping would produce higher quality solutions on average, it is our view that the marginal improvement in solution quality would not be worth the increased computational cost and complexity.

To illustrate the merits of the SSP-MS model, we consider a 2-searcher, 10-target SSP example. As the numerical results in Section 4 indicate, a significant amount of computing time is usually

	Number of Targets				
	3	5	7	8	10
Avg Time (sec)	0.19	0.24	0.30	0.31	0.34
Min Time (sec)	0.14	0.17	0.22	0.19	0.22
Max Time (sec)	0.40	0.42	0.65	0.55	0.67
Num Optimal	80	42	41	31	29
Avg Gap (%)	1	3	2	3	2
90th Percentile Gap (%)	3	7	6	6	6
Max Gap (%)	29	34	23	48	30

Table 9: SSP Heuristic performance results. The heuristic presented in Section 3.3 produces runtimes that are at or below the half-second mark for all 3, 5, 7, 8, and 10-target SSP instances. Accuracy in finding the optimal solution appears to diminish as problem size increases, however the average, 90th percentile, and maximum optimality gaps remain fairly stable.

required to solve a 10-target SSP. When another searcher is considered in an already difficult problem, squaring the number of possible search plans, we should expect that the B&B runtime would increase substantially. We consider search planning to take place on a 24-hour cycle. Thus, runtimes that exceed 24 hours are clearly unacceptable. It is apparent that solving the MINLP directly using either BONMIN(ECP) or DICOPT would not produce a solution in an acceptable amount of time. For this reason, we proceed using B&B.

The spatial distribution of the targets in this example is split (see Figure 6), with 4 of them in the southwest region and 6 of them in the northeast region. The speeds, travel times, and expected detection value of these targets are varied (see Table 11). Given two aerial search assets, a P-3 and an HC-130 (see Table 10), operating from the same home station, we wish to develop a plan that maximizes the expected detection value. Complete searcher and target data for this problem instance are provided in Table 10 and Table 11 respectively.

Searcher	V	$\hat{V}$	T	D	W
A	325	205	10	24	15
B	280	240	10	24	15

Table 10: 2-searcher SSP example data is based on known performance specifications for a P-3 (searcher A) and a HC-130 (searcher B).

We select this particular problem out of the 100 randomly generated 10-target problem instances because it exhibits many of the interesting features we have observed in our study of these problems. From a planner's perspective, it is not clear how to form a good plan in this scenario, let alone an optimal one. Targets 1,2,3, and 5 have high expected detection values q. Good judgment would dictate that these targets would likely be included in the optimal plan, but it is difficult to determine the search order, the searcher-to-target assignment, and whether or not other targets will be searched at all.

With two search assets, it seems reasonable that a search planner would, based on spatial regions, assign targets 2 and 3 to one searcher, and targets 1 and 5 to the other searcher. As discussed in Section 3.3, we assume ascending order of expected arrival times. Furthermore, since the targets 2 and 3 appear to be farther away from the home station, it makes sense to assign the faster P-3 to

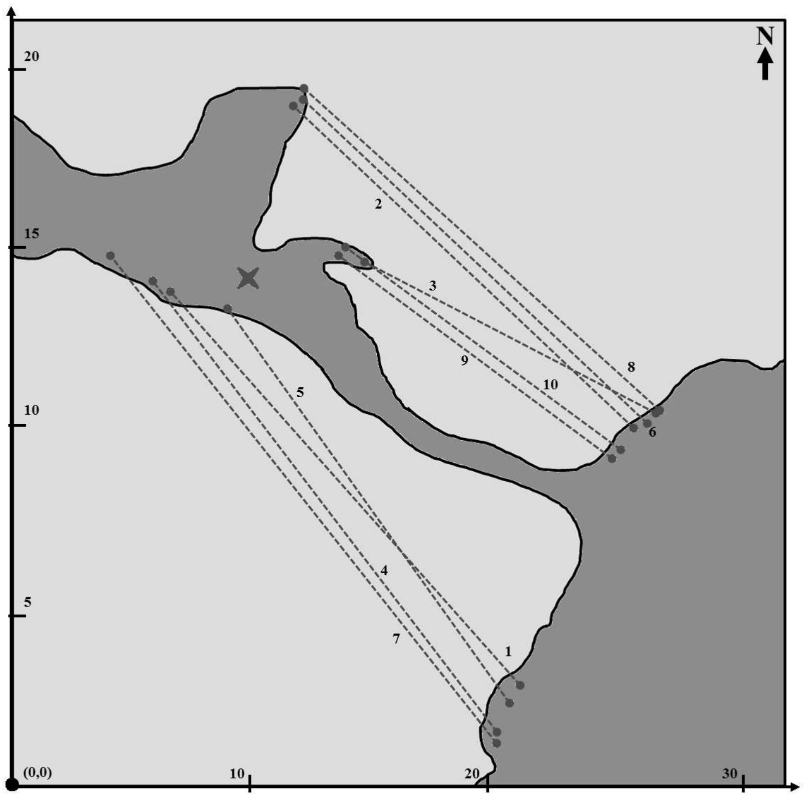

Figure 6: 10-target, 2-searcher example SSP scenario map. 4 targets moving in the southwest region (lower left) and 6 in the northeast region (upper right). Movement tracks are labeled corresponding to target numbers shown in Table 11.

those targets. This search plan is reflected in Table 12 as the Manual Plan for 2 searchers.

If the search planner only has one search asset available, developing a good plan somehow appears even more difficult. While planning to search all the high value targets uses a substantial amount of travel time, which is a limited resource, it may be too difficult for the planner to determine which of these targets to rule out. Therefore, the planner may assign all of them to the searcher, again in order ascending arrival time. This search plan is reflected in Table 12 as the Manual Plan for 1 searcher.

For convenience, we define a *p%-solution* to be a solution that has an optimality gap of p percent. Table 12 shows a comparison between results for the manual plans, search plans derived from the SSP heuristic, and search plans computed using Algorithm B&B. The manual plan for a single searcher is clearly an inferior course of action, a 17%-solution. More time is spent transiting between search regions than is spent actually searching for targets. From a resource trade-off perspective, we can certainly do better. The single searcher heuristic, a 3%-solution, produces a plan that is close to optimal. Two of the high value targets selected (2 and 5) are correct and in the correct order, but the heuristic fails to recognize the value of adding target 1 to the plan. The optimal single searcher plan adds target 1 because the additional transit resource expenditure is worth the incremental bang-for-buck in searching this additional target.

The manual 2-searcher plan has an optimality gap of 3%, much better than its single searcher counterpart. This is possible because this problem instance happens to have target movement tracks for high value targets, two in each region, that coincide nicely with having two search assets. We

Target	U	τ	$\tilde{\tau}$	$\boldsymbol{\rho}$	$\bar{\boldsymbol{\rho}}$	$\tilde{\rho}$	q
1	59.5	3	4	(24.9, 3.1)	(9.1, 14.3)	64	4282
2	62.3	9	2	(25.7, 10.1)	(12.7, 20.2)	55	4554
3	61.7	3	2	(27.0, 11.2)	(16.3, 15.2)	85	3236
4	57.0	10	3	(21.2, 1.9)	(8.0, 14.7)	59	553
5	56.7	4	3	(21.5, 2.6)	(11.5, 13.2)	27	4687
6	59.2	11	1	(26.6, 10.8)	(13.0, 20.9)	91	937
7	55.3	12	1	(21.1, 1.8)	(5.8, 15.7)	93	661
8	64.5	3	4	(27.7, 11.7)	(12.9, 20.6)	61	837
9	64.4	5	4	(24.4, 9.1)	(14.3, 15.9)	45	1317
10	58.8	7	1	(24.9, 9.5)	(15.8, 15.4)	66	1290

Table 11: 10-target SSP example data is randomly generated as described in Section 4.

see that the 2-searcher heuristic obtains its plan, as expected, by starting with the single searcher heuristic solution and adding in a plan for the second searcher. The result has each searcher crossing between geographic regions in their respective paths. This plan is slightly better than the manual plan and yields an optimality gap of 2%.

	Expected Detection	Runtime	Search Order	Search Time (hrs)	Transit Time (hrs)
Manual Plan, 1 searcher	6514	<0.1 sec	3-5-1-2	4.4	5.6
Heuristic Plan, 1 searcher	7662	0.4 sec	5-2	6.5	3.5
Optimal Plan, 1 searcher	7889	57.6 secs	1-5-2	6.4	3.6
Manual Plan, 2 searchers	12058	<0.1 sec	A: 3-2	5.8	4.2
			B: 5-1	7.1	2.9
Heuristic Plan, 2 searchers	12106	1.1 secs	A: 5-2	6.5	3.5
			B: 3-1	6.4	3.6
Optimal Plan, 2 searchers	12373	4.7 hrs	A: 3-10-2	5.9	4.1
			B: 1-5	7.2	2.8

Table 12: Comparison of manual, heuristic, and optimal search plans for 10-Target SSP example with 1 and 2 searchers. Search order includes home station (not shown) at the beginning and end of all paths. Plans for 2 searcher instances are labeled A and B, and correspond to searcher data in Table 10. Search time and transit time refer to total time resource expenditure searching and in transit.

The optimal 2-searcher plan, depicted in Figure 7, achieves a more favorable dwell-to-transit resource trade-off with a total of 6.9 hours spent in transit. This is possible because of two not-so-obvious modifications to the manual search plan. First, target 10 is added to searcher A's path. This enables the searcher to collect reward for searching another target while allowing target 2 to get closer to the searcher's location. Thus, more time is spent searching than in transit. Second, the order of targets 1 and 5 are swapped in searcher B's path. This goes counter to earliest arrival time ordering. This plan takes advantage of the facts that the movement tracks become quite close (where the dotted lines cross) and that target 5 is essentially coming straight at the home base of the searcher. Combined, these modifications yield reduced transit time and increased search time.

Observe that in the optimal 2-searcher plan there are no targets searched by both searchers, see Figure 7. One might conclude that solving this problem is equivalent to solving two separate single searcher SSPs. This is not the case. The optimal solutions shown in Table 12 show that solutions are not nested with respect to adding searchers. While the optimal 1-searcher plan searches regions 1, 5, and 2, these regions are not allocated to the same searcher in the optimal 2-searcher plan.

While the runtime required to obtain the optimal solution is relatively long, a breath-first-search of the first three levels of the enumeration tree, done in 9.6 minutes, proves that the objective function value of the heuristic is guaranteed to be within 20% of optimality. In this problem instance a search planner under time pressure could choose to use the heuristic search plan, actually (but unknown to the planner) a 2%-solution, and be assured that this plan is within 20% of optimality.

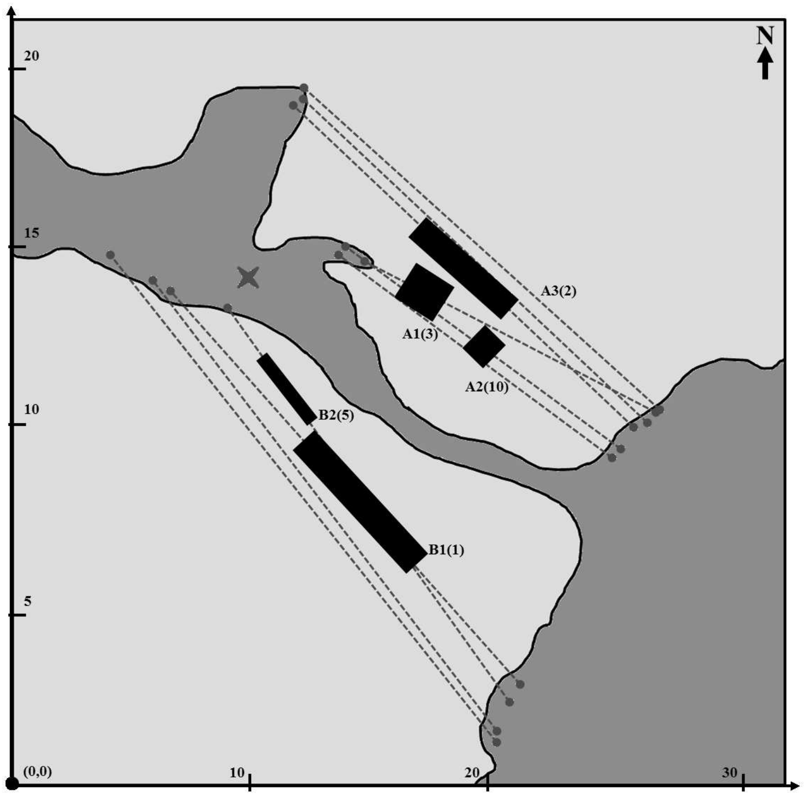

Figure 7: 10-target, 2-searcher example SSP optimal solution map. Labels indicate search plan ordering. For example, label “A2(10)” is the searcher A to target 10 assignment, where this search region appears 2nd in searcher A’s path. All search regions are searched in the bottom-right to top-left direction due to the direction of target motion. The size of each rectangular block corresponds to the total area of respective search region during the time when the searcher is performing search actions in that region.

5.2 Complex Target Motion

Real-world scenarios where SSPs arise can require the use of models that are more complex than those discussed thus far. For example, a particular target’s movement track may not be along a straight line. The target may be traveling along a track that follows a particular stretch of coastline, or the target may navigate around islands or other geographic obstacles. It is also possible that the search region associated with a target changes as the target moves, perhaps due to changing weather or intelligence-driven changes to the uncertainty ranges themselves. The speed of the target may also change with ocean state conditions as a smuggler travels. All of these considerations can be modeled with piecewise linear target movement tracks.

We consider the situation where target motion is nonlinear, but can be approximated by piecewise linear segments. We model the nodes N as the segmented search regions. The nodes represent search regions as in **SSP**, but here they do not necessary correspond to unique targets. Let $F \subseteq \hat{N}$ be the set of *first* segment target paths, one for each actual target. Let $B(i) \subseteq \hat{N}, i \in F$ denote the set of search region segments for each target. The piecewise linear target motion model is shown below.

Problem SSP-PWL:

$$\begin{aligned} \max_{\boldsymbol{a},\boldsymbol{d},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y}} \quad & \sum_{i \in F} q_i \left(1 - \exp\left\{-\sum_{j \in B(i)} \alpha_j d_j y_j\right\}\right) \\ \text{s.t.} \quad & \text{constraints (9b) through (9m)} \end{aligned}$$

SSP-PWL only differs from **SSP** in the objective function. Similarly to **SSP-MS**, we sum total search effort in the exponential. In this case the summation is over all segments in the piecewise linear target movement track. We set τ_j^{min} and τ_j^{max} in (9f) and (9g) respectively to define the connections between target path line segments.

Runtime can be improved by reducing the size of the arc set A, thus reducing the number of partial path extensions that must be considered in step 2 of Algorithm B&B. Clearly, any arc (i, j) where j precedes $i \in B(j)$ should be eliminated from A. Performing this elimination procedure for all target path line segments reduces the dimension of $\hat{\boldsymbol{X}}$ in Algorithm B&B. We have observed that this can have a noticeable effect on computation time.

To illustrate the merits of the SSP-PWL model, we consider a 2-searcher, 5-target SSP with piecewise linear target tracks. The additional segments make this problem equivalent in size to a 12-target SSP. In this scenario, one of the targets, target 1, navigates around an island in the southwest region. Once past the island, the track of the target becomes more and more uncertain. Intelligence estimates are solid in estimating the first part of this target's track, but analysts are less certain about the target's arrival location. This target track is modeled by four segments (1a, 1b, 1c, and 1d) with increasing ranges of location uncertainty. Another target, target 2, moves along a track that follows the coastline in the northeast region. The piecewise linear track for this target is captured by five segments (2a, 2b, 2c, 2d, and 2e). The other three targets move along linear tracks. The target movement tracks for this scenario are shown in Figure 8.

Target data for this scenario is given in Table 13. Observe that the location uncertainty $\tilde{\rho}$ for target 1 increases by 25nm in each subsequent segment. This has the effect of increasing the area of the search region by 50 nm^2. The expected departure times τ in subsequent segments, $j \in N \backslash F$, are derived by simply propagating the first segment forward in time, setting τ to the expected arrival time of the previous segment. Target departure and arrival locations are chosen to fit the scenario description and to make the map illustration readily viewable. All other values are randomly generated as described in Section 4.

This scenario presents a difficult challenge for a search planner. With 2 search assets and 5 targets there are enough resources to make an effort to search all targets, but it is not clear

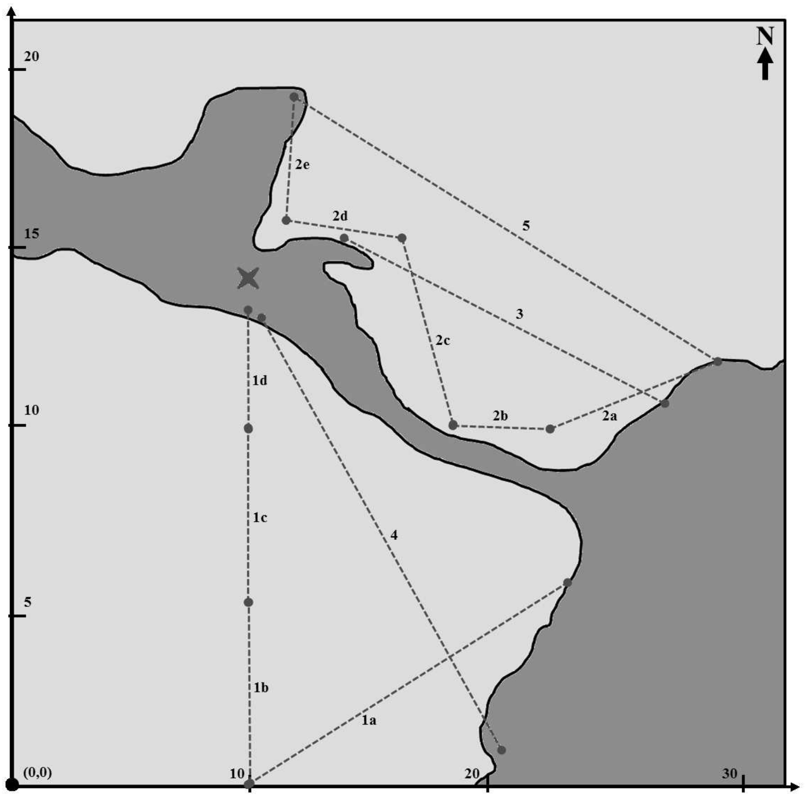

Figure 8: 5-target piecewise linear example map. Target 1 navigates around an island and has increasing range of location uncertainty. Target 2 moves along a track that follows the coastline.

how to order the search or how determine the searcher-to-target assignment. Perhaps the most difficult aspect is that the search region for target 1 increases in size as it gets closer to the home station. Nominally, a good search plan would have a searcher wait at home station while the target approaches, but doing so in this case makes it harder to detect the target later in the mission execution period as the search region area increases.

The optimal solution, shown in Figure 9, is obtained in 10.5 hours by using Algorithm B&B with settings described in Section 5.1. It is clear from the map that the path for searcher A achieves a favorable dwell-to-transit resource trade-off as search regions are visited at a point that seems close to the home station. Searcher B is forced to travel a relatively long distance in order to search for target 1 at a point where its search region has a small area. This allows target 5 to be searched at a point that is close to the home station. By inspection, this optimal search plan makes sense, but developing it without using optimization techniques entails making nonintuitive assignments in the search plan.

A breath-first-search of the first three levels of the enumeration tree, accomplished in 2.7 hours, provides a solution that is guaranteed to be within 51% of optimality. The solution turns out to be a 1%-solution. Given only a couple of hours for planning, a search planner could choose to use this search plan in lieu of the heuristic plan; obtained in 0.7 seconds, but with no optimality gap guarantee. The heuristic gives a 53%-solution.

Target	U	τ	$\tilde{\tau}$	$\boldsymbol{\rho}$	$\bar{\boldsymbol{\rho}}$	$\tilde{\rho}$	q
1a	64.1	0	2	(23, 4)	(10, 0)	25	2670
1b	64.1	12.7	2	(10, 0)	(10, 5)	50	2670
1c	64.1	17.7	2	(10, 5)	(10, 10)	75	2670
1d	64.1	22.7	2	(10, 10)	(10, 14)	100	2670
2a	61.4	2	3	(28, 12)	(21, 10)	45	890
2b	61.4	9.1	3	(21, 10)	(18, 10)	45	890
2c	61.4	12.03	3	(18, 10)	(17, 16)	45	890
2d	61.4	18.0	3	(17, 16)	(13, 17)	45	890
2e	61.4	22.0	3	(13, 17)	(13, 20)	45	890
3	62.7	3	1	(26, 11)	(15, 16)	80	2739
4	62.6	10	3	(21, 1)	(12, 13)	62	3914
5	55.5	5	1	(28, 12)	(13, 20)	91	2547

Table 13: 5-target piecewise linear SSP example data.

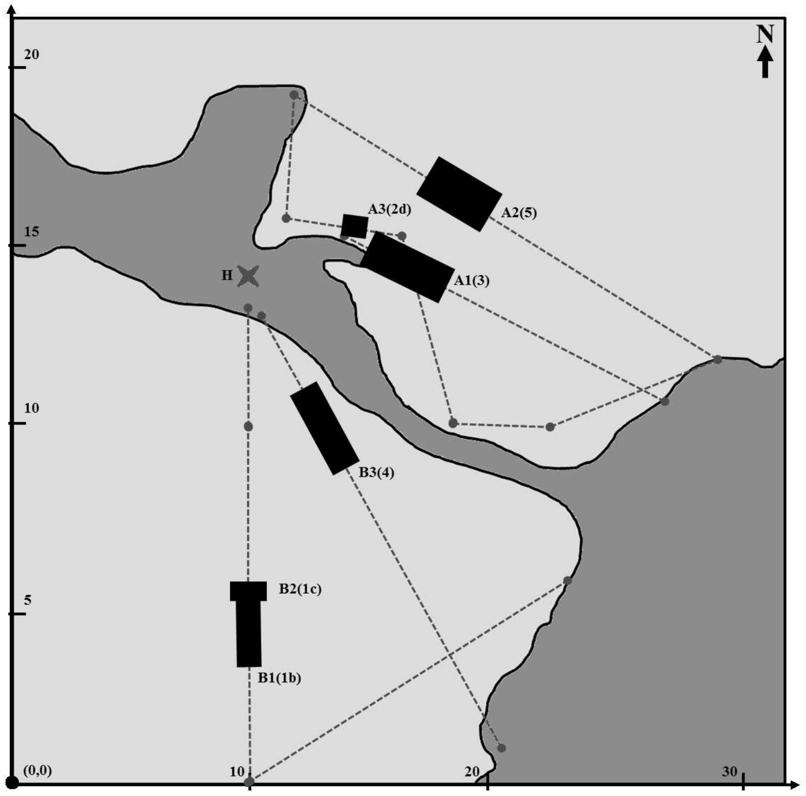

Figure 9: 2-searcher, 5-target piecewise linear optimal solution map. Searcher A is assigned the path 3-5-2d, searching segment d of target 2 last. Searcher B is assigned the path 1b-1c-4, searching segments b and c of target 1 first. The size of each rectangular block corresponds to the total area of respective search region during the time when the searcher is performing search actions in that region.

6 Conclusions

This article introduces a routing problem GOP-RDR and presents a specialized branch-and-bound algorithm that is built upon partial path relaxations which exploit resource trade-offs that are inherent in these problems. We formulate a search problem SSP as an important special case of a GOP-RDR and provide an efficient heuristic for computing high quality solutions in a small amount of time.

Numerical testing on randomly generated SSP instances reveals that our branch-and-bound algorithm outperforms standard MINLP solvers for problems with 7 or more targets. In large problem instances, with 10 targets, our algorithm is currently the only viable approach to obtain optimal solutions within 30 minutes of computing time.

We propose extensions to the SSP which allow practitioners to model realistic search planning scenarios that involve multiple heterogeneous searchers and complex target motion. We observe that optimal search plans can usually be explained by an intuitive story, but obtaining these search plans without using sophisticated optimization techniques would be a difficult task.

Acknowledgement

The second author acknowledges financial support from the Office of Naval Research, Mathematical Optimization and Operations Research Program.

A SSP Heuristic Algorithm

The SSP heuristic begins by defining Δ as the *temporal clustering parameter*, which controls how close we allow target clusters to be with respect to time. We assume that the problem of interest can be separated into spatial clusters $\sigma \in \Sigma$ based on geographical boundaries. This is the case in our SSP example where smugglers are transiting through water either side of a large land mass. Furthermore, since we are concerned with seagoing smugglers they cannot move from one region to another. We denote by K_σ the set of targets that belong to spatial cluster σ.

SSP Heuristic:

Phase I. *Initialization*

1. Initialize cluster count $k = 1$, order index $o = 1$, and null path $\boldsymbol{x} = \mathbf{0}$. Solve $\mathbf{RSSP}(0)$ and record the optimal solutions $\boldsymbol{d}^*$. Construct the set of searchable targets $\check{N} = N \backslash \{j \in N : d_j^* = 0\}$.
2. Compute bang-for-buck ratios $\beta_j = q_j/(\tilde{\tau}_j \tilde{\rho}_j U_j), \forall j \in \check{N}$.

Phase II. *Target Clustering*

3. For each spatial cluster $\sigma \in \Sigma$:

```
        Initialize time window parameter $\check{\tau}_\sigma = \min_{j \in \check{N} \cap K_\sigma} \{\tau_j^{max}\}$. While $\check{\tau}_\sigma + \Delta < \max_{j \in \check{N} \cap K_\sigma} \{\tau_j^{max}\}$:
            For each target $j \in K_\sigma$:
                Assign target $j$ to cluster $\kappa_k$ if $\tau_j^{max} \in [\check{\tau}_\sigma, \check{\tau}_\sigma + \Delta)$.
            Increment $\check{\tau}_\sigma = \check{\tau}_\sigma + \Delta$.
            If one or more targets are assigned to cluster $\kappa_k$ in this time interval, increment $k = k + 1$.
```

4. For each cluster κ_k:

   Order targets $j \in \kappa_k$ in ascending value of τ_j^{max}. Compute cluster order value $\nu_k = \min\{\tau_j^{max} : j \in \kappa_k\}$.

5. For each cluster κ_k, considered in ascending order value ν_k:

   For each target $j \in \kappa_k$:

   Assign search order $O_j = o$ and increment $o = o + 1$.

6. Assign order $O_0 = 0$ to the home station and order $O_{n+1} = |\check{N}| + 1$ to the recovery location.

7. Form the initial path $\boldsymbol{x}$ by setting $x_{i,j} = 1$ for all i and j with consecutive orderings O_i and O_j. Solve $\mathbf{SSP}(\boldsymbol{x})$. Save incumbent path $\bar{\boldsymbol{x}} = \boldsymbol{x}$. If the problem is feasible, save the heuristic objective value Z_H^* as the optimal objective function value of this problem. Otherwise, set $Z_H^* = -\infty$.

Phase III. *Feasibility Check*

8. If $Z_H^* = -\infty$:

   For all targets $j \in \check{N}$, considered in ascending order β_j:

   Do *Remove j procedure*: { Remove target j from the path $\boldsymbol{x}$ by setting $x_{i,j} = 0$ (for $i : O_i = O_j - 1$), $x_{j,i'} = 0$ (for $i' : O_{i'} = O_j + 1$), and $x_{i,i'} = 1$ (for $i : O_i = O_j - 1$ and $i' : O_{i'} = O_j + 1$). Remove j from the set $\check{N}$. Solve $\mathbf{SSP}(\boldsymbol{x})$. }

   If a feasible solution is found, set Z_H^* to the objective function value of this solution and go to step 9.

Phase IV. *Cluster Seam Refinement*

9. Save incumbent path $\bar{\boldsymbol{x}} = \boldsymbol{x}$. For each seam between clusters κ_{k-1} and κ_k, where $k > 1$:

   Let j be the target in that last order position in cluster κ_{k-1}. Let i' be the target in the first position in cluster κ_k. If $\tau_j^{max} > \tau_{i'}^{max}$ and $\beta_j < \beta_{i'}$, do *Remove j procedure* defined in step 8. If Z_H^* is improved, save incumbent path $\bar{\boldsymbol{x}} = \boldsymbol{x}$. Otherwise, reset incumbent path $\boldsymbol{x} = \bar{\boldsymbol{x}}$.

Phase V. *Greedy Target Removal*

10. For each target $j \in \check{N}$, considered in ascending order β_j:

    Do *Remove j procedure* defined in step 8. Solve $\mathbf{SSP}(\boldsymbol{x})$. If Z_H^* is improved, save incumbent path $\bar{\boldsymbol{x}} = \boldsymbol{x}$. Otherwise, reset incumbent path $\boldsymbol{x} = \bar{\boldsymbol{x}}$.

11. Return heuristic path $\bar{\boldsymbol{x}}$ and solution Z_H^*.

References

[1] C. Archetti, A. Hertz, and M. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, Dec. 2007.

[2] S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. *4OR A Quarterly Journal of Operations Research*, 5(3):211–230, 2007.

[3] S. Butt and T. M. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers and Operations Research*, 21(1):101–111, 1994.

[4] S. Butt and D. Ryan. An Optimal Solution Procedure for the Multiple Tour Maximum Collection Problem Using Column Generation. *Computers and Operations Research*, 26:427–441, 1999.

[5] J. D. Camm, A. S. Raturi, and S. Tsubakitani. Cutting Big M Down to Size. *Interfaces*, 20:61–66, Dec. 1990.

[6] I. Chao, B. Golden, and E. Wasil. A Fast and Effictive Heuristic for the Orienteering Problem. *European Journal of Operational Research*, 88(3):475–489, 1996.

[7] R. Dell, J. Eagle, G. Martins, and A. Santos. Using Multiple Searchers in Constrained-Path, Moving-Target Search Problems. *Naval Research Logistics*, 43(4):463–480, 1996.

[8] M. Desrochers and G. Laporte. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991.

[9] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, Jan. 2002.

[10] J. Eagle and J. Yee. An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Operations Research*, 38(1):110–114, 1990.

[11] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987.

[12] I. E. Grossmann, J. Viswanathan, A. Vecchiette, R. Raman, and E. Kalvelagen. DICOPT user manual. 2012.

[13] D. Grundel. Constrained search for a moving target. In *Proceedings of the 2005 international conference on Collaborative technologies and systems*, CTS'05, pages 327–332, Washington, DC, USA, 2005. IEEE Computer Society.

[14] T. Ilhan, S. Iravani, and M. Daskin. The orienteering problem with stochastic profits. *IIE Transactions*, 40(4):406–421, Feb. 2008.

[15] G. Laporte and S. Martello. The selective traveling salesman problem. *Discrete Applied Mathematics*, 26:193–207, 1990.

[16] H. D. Moser. Scheduling and routing tactical aerial reconaissance vehicles. Master's thesis, Naval Postgraduate School, 1990.

[17] B. A. Murtagh, M. A. Saunders, and P. A. Gill. MINOS user manual. 2012.

[18] R. Ramesh and K. M. Brown. An efficient four-phased heuristic for the generalized orienteering problem. *Computers & Operations Research*, 18(2):151–165, 1991.

[19] R. Ramesh, Y. Yoon, and M. Karwan. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing*, 4(2):155–165, 1992.

[20] J. O. Royset and D. Reber. Optimized Routing of Unmanned Aerial Systems for the Interdiction of Improvised Explosive Devices. *Military Operations Research*, 14(4):5–19, 2009.

[21] J. O. Royset and H. Sato. Route Optimization for Multiple Searchers. *Naval Research Logistics*, 57(8):701–717, 2010.

[22] N. Sahinidis and M. Tawarmalani. BARON user manual. 2012.

[23] H. Sato and J. O. Royset. Path Optimization for the Resource-Constrained Searcher. *Naval Research Logistics*, 57(5):422–440, 2010.

[24] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, 2009.

[25] J. Silberholz and B. Golden. The effective application of a new approach to the generalized orienteering problem. *Journal of Heuristics*, 16(3):393–415, 2010.

[26] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. Vanden Berghe, and D. Van Oudheusden. A Personalized Tourist Trip Design Algorithm for Mobile Tourist Guides. *Applied Artificial Intelligence*, 22(10):964–985, Oct. 2008.

[27] H. Tang and E. Miller-Hooks. A TABU search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407, June 2005.

[28] M. Tawarmalani and N. Sahinidis. Global Optimization of Mixed Integer Nonlinear Programs: A Theoretical and Computational Study. *Mathematical Programming*, 99:563–591, 2004.

[29] P. Toth and D. Vigo. *The vehicle routing problem*, volume 9 of *SIAM Monographs on Discrete Mathematics and Applications.* Society for Industrial and Applied Mathematics, 2002.

[30] F. Tricoire, M. Romauch, K. F. Doerner, and R. F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2):351–367, Feb. 2010.

[31] K. Trummel and J. Weisinger. The Complexity of the Optimal Searcher Path Problem. *Operations Research*, 34(2):324–327, 1986.

[32] T. Tsiligirides. Heuristic Methods Applied to Orienteering. *Journal of the Operational Research Society*, 35(9):797–809, 1984.

[33] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, Feb. 2011.

[34] S. Vigerske. COIN user manual. 2012.

[35] Q. Wang, X. Sun, and B. Golden. Using artificial neural networks to solve generalized orienteering problems. In C. Dagli, M. Akay, C. Chen, B. Fernandez, and J. Ghosh, editors, *Intelligent Engineering Systems Through Artificial Neural Networks: Volume 6*, pages 1063–1068. 1996.

[36] X. Wang, B. Golden, and E. Wasil. Using a genetic algorithm to solve the generalized orienteering problem. In B. Golden, S. Ragahavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 264–274. 2008.

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

[38] J. Yi. Vehicle Routing with Time Windows and Time-Dependent Rewards: A Problem from the American Red Cross. *Manufacturing & Service Operations Management*, 5(1):74–77, 2003.