Route 'Em and Count 'Em: A Two-Stage Stochastic Programming Model for Anti-Submarine Operations

Vanessa Ann Beddoe Sawkmie
Jeff Linderoth
Department of Industrial and Systems Engineering
Wisconsin Institute of Discovery, University of Wisconsin-Madison
sawkmie@wisc.edu linderoth@wisc.edu

Abstract

Tracking targets in undersea warfare requires successful detection by an active search asset. Maximizing detection likelihood requires strategic placement and routing of the search assets in the search region over the planning horizon. We develop a two-stage stochastic integer programming model that maximizes the expected total reward for target detections under uncertainty in target motion and sensor detection performance, where time-dependent rewards allow decision-makers to prioritize early detections. Leveraging the problem structure, we derive a closed-form solution for the second-stage subproblem, enabling efficient solution of large-scale instances via Benders decomposition. Computational experiments demonstrate the scalability of the decomposition approach and the quality of solutions obtained through sample average approximation.

Keywords: Two-stage stochastic programming; mixed-integer programming; ASW operations;

1 Introduction

Anti-Submarine Warfare (ASW) operations require detecting highly stealthy submarines using search assets such as submarines, destroyers, and aircraft that deploy sensors to detect signals from targets. Each asset type has distinct characteristics including speed, sensor range, endurance, and other operational constraints. Two primary sources of uncertainty further complicate the planning of these operations. Motion uncertainty arises from lack of knowledge of target missions and movements. Detection uncertainty arises from factors affecting sensor performance, including environmental conditions such as weather, sea-life

concentration, and commercial shipping traffic, as well as oceanographic factors such as bathymetry, temperature, and water density variations that alter sound profiles and sensor accuracy (Urick, 1975). The combination of limited search assets and target and environment uncertainty makes effective routing and placement critical for ASW operation success.

Since exact target locations and sensor performance cannot be known in advance, deterministic planning strategies may lead to poor performance. We propose a stochastic programming approach and formulate the problem as a two-stage stochastic program with the objective to maximize the expected reward from detecting targets. The model optimizes asset deployment in a search theatre over a finite planning horizon, while incorporating stochastic target movements and detection events. We consider a one-sided search (Stone, Royset, & Washburn, 2016), in which targets follow their own mission trajectories without reacting to or evading search assets.

We use a discrete-time, discrete-space representation of the ASW search planning problem. The planning horizon is divided into uniform time periods, and the search theater is partitioned into a finite set of geographic cells. This representation provides a finite planning structure in which asset positions, target movement, and detection outcomes can be indexed by time period and cell. Finer discretizations provide more detailed representations of motion and detection opportunities, while coarser discretizations reduce problem size. The spatial and temporal resolutions thus govern the tradeoff between model fidelity and computational tractability. We examine this tradeoff computationally in Section 4.4.

We distinguish two classes of search assets. Route-based assets, such as submarines and destroyers, search continuously along transit paths using sonar arrays. Pattern-based assets, such as aircraft, execute pre-defined search patterns by deploying sonobuoys. For route-based assets, we determine the location of each asset at each time period, thereby constructing a path through the operational area. For pattern-based assets, we select which search pattern each asset executes and when it is executed.

1.1 Literature review

Search problems in the literature vary by the number of searchers and targets, the mobility of the targets, whether searchers and targets interact, and whether time and space are modeled in discrete or continuous form. Most studies consider multiple searchers (G. Brown, Kline, Thomas, Washburn, & Wood, 2011; Kress & Royset, 2008; Laan, Barros, Boucherie, Monsuur, & Noordkamp, 2020; Lejeune, Royset, & Ma, 2024; Royset & Sato, 2010; Sidoti et al., 2017; Thomas, 2008) and multiple targets (An et al., 2012; G. Brown et al., 2011; J. Foraker, Royset, & Kaminer, 2016b; Sidoti et al., 2017; Thomas, 2008), though single-searcher (Eagle & Yee, 1990; Sato & Royset, 2010) and single-target (Eagle & Yee, 1990; Laan et al., 2020; Lejeune et al., 2024; Sato & Royset, 2010) formulations exist. Most existing studies adopt discrete representations of both time and space. Some exceptions include a continuous-time model (Thomas, 2008), a continuous-space model (Akbori, 2004), and few models that consider continuity in both time and space (J. Foraker, Royset, & Kaminer,

2016a; J. Foraker et al., 2016b; J. C. Foraker, Royset, & Kaminer, 2011). Our model falls within the discrete-time, discrete-space framework with multiple searchers and targets.

Target movement is commonly modeled using Markovian processes where future locations depend only on the current state (Ding, 2018; Eagle & Yee, 1990; Lejeune et al., 2024; Sato & Royset, 2010). Our approach also uses a Markovian process where transitions depend on the current state, but incorporates destination-based movement where angular deviations in heading direction scale with the remaining distance to the destination. This aligns with how submarines navigate during missions when executing goal-oriented maneuvers toward specific destinations (see Section 3.1.1).

For coordinating surveillance assets (which we refer to as pattern-based assets), An et al. (2012) partition the search area into rectangular boxes whose sizes are determined heuristically during allocation. Our framework pre-defines these search patterns and distinguishes them based on the asset.

Most prior works maximize the probability of detection or minimize the probability of non-detection (S. S. Brown, 1980; Ding, 2018; Eagle & Yee, 1990; Lejeune et al., 2024; Royset & Sato, 2010; Sato & Royset, 2010; Sidoti et al., 2017; Thomas, 2008). When multiple assets search for the same target over multiple time periods, the objectives become products of individual non-detection probabilities, introducing nonlinearities in the model. Our formulation maximizes the expected reward from detecting targets and avoids nonlinearities through a two-stage structure where the second stage is a counting problem rather than a complicated recourse action. By enforcing that each target contributes at most once to the detection count, we maintain linearity of our model, albeit at the cost of introducing additional variables.

Existing solution approaches to search problems span a wide range of techniques. Mixed-integer programming has been applied in (Eagle & Yee, 1990; Lejeune et al., 2024; Royset & Sato, 2010). A two-stage stochastic programming approach is studied by Kress and Royset (2008), where the model determines optimal locations for ground control units and assigns search schedules for unmanned aerial vehicles. Game-theoretic models have been applied to capture adversarial interactions between searcher and target (G. Brown et al., 2011; Laan et al., 2020; Thomas, 2008). Dynamic programming has been employed to handle sequential decision-making under uncertainty (Sidoti et al., 2017), decomposing the asset allocation problem into sequential stages over the planning horizon. Other algorithmic approaches include Gauss-Seidel algorithms with rollout strategies for asset allocation (An et al., 2012), branch-and-bound for path planning (Sato & Royset, 2010), and forward-backward iterative methods (Ding, 2018).

To the best of our knowledge, Benders decomposition has not been applied to search and detection problems in the literature. We solve the problem using the branch-and-cut Benders decomposition algorithm (Fortz & Poss, 2009; Naoum-Sawaya & Elhedhli, 2013). Leveraging the problem structure, we derive closed-form expressions for generating Benders cuts directly, eliminating the need to solve subproblems iteratively. This substantially reduces

computational cost and enables solutions for large operational areas.

1.2 Contributions

Our work makes the following contributions:

- We introduce a two-stage stochastic integer programming framework for ASW operations that accommodates heterogeneous route-based and pattern-based assets.
- We incorporate time-dependent rewards that allow decision-makers to prioritize early detections, with the expected number of detected targets recovered as a special case when rewards are uniform across targets and time periods.
- We formulate the second stage as a counting problem, maintaining model linearity while ensuring each target contributes at most once to the objective.
- We model target motion as a goal-oriented Markovian process in which each target transits toward a fixed goal destination under stochastic angular and speed perturbations, providing a realistic representation of mission-driven movement.
- We derive closed-form expressions for the second-stage subproblem. These expressions allow for efficient generation of Benders cuts and reduce the computational cost of solving large-scale instances.
- We conduct computational experiments that demonstrate the scalability of the decomposition approach, evaluate solution quality through sample average approximation, and report improvements over a greedy baseline representative of current operational planning.

The remainder of the paper is organized into four sections. Section 2 presents the two-stage stochastic programming model. Section 3 outlines the solution approach and the scenario generation methods. Section 4 presents the computational experiments and results. Finally, Section 5 summarizes the key findings and discusses potential future research directions.

2 Model description and formulation

2.1 Problem description

We formulate the ASW search problem as a two-stage stochastic integer program for deploying search assets in the search region over the planning horizon. Let I denote the set of route-based assets, such as submarines and destroyers, that traverse the search region along

prescribed routes. Let F denote the set of pattern-based assets, such as Maritime Patrol and Reconnaissance Aircraft (MPRAs), that deploy sonobuoy fields.

The geographical search region is partitioned into a set $A = \{a_1, a_2, \ldots, a_n\}$ of mutually disjoint cells or areas that cover the entire search region. The planning horizon T_{hor} is discretized into $\tau = \lceil T_{hor}/\Delta t \rceil$ time periods, forming the set $T = \{1, \ldots, \tau\}$, where Δt denotes the period length. Route-based assets transit between adjacent cells each period, while pattern-based assets execute patterns from the set P of pre-defined search patterns, where each pattern spans multiple adjacent cells.

The objective of the optimization model is to maximize the expected reward from detecting targets from the set J, where r_{jt} denotes the reward for detecting target $j \in J$ during period $t \in T$. This formulation allows decision-makers to prioritize early detections by assigning higher rewards to earlier time periods, reflecting the strategic importance of timely intelligence in ASW operations. When $r_{jt} = 1$ for all $j \in J$ and $t \in T$, the objective reduces to maximizing the expected number of targets detected.

The fidelity of discretization in time and space affects both model accuracy and computational tractability. Finer discretization allows more accurate representation of search dynamics but increases computational effort. In Section 4.4, we examine this trade-off and its effect on solution quality.

2.2 Uncertainty modeling

The model has two sources of uncertainty: motion uncertainty and detection uncertainty. For each target $j \in J$, motion uncertainty is represented by a discrete-time stochastic process

$$\mathcal{T}_j = \{\, \mathcal{T}_j(t, \omega) : t \in T \,\},$$

where $\mathcal{T}_j(t, \omega)$ denotes the position of target j at time period $t \in T$ under realization $\omega \in \Omega$, and $\mathcal{T}_j(t, \omega) \in D \subset \mathbb{R}^2$, where D is the domain of interest.

To connect the continuous spatial domain with our discrete area model, we define a region mapping $\mathcal{M} : D \to A$ that assigns each spatial coordinate to its corresponding cell:

$$\mathcal{M}(x) = a_k \qquad \Longleftrightarrow \qquad x \in a_k. \tag{1}$$

Detection uncertainty is modeled using Bernoulli random variables under realization $\omega \in \Omega$, conditional on the asset and target occupying the same area. For route-based assets $i \in I$, we define $\gamma^{\text{route}}_{ijat}(\omega) \in \{0, 1\}$, where $\gamma^{\text{route}}_{ijat}(\omega) = 1$ indicates that asset i detects target j given both are located in area a at time t. For pattern-based assets $i \in F$, we define $\gamma^{\text{pat}}_{ijpt}(\omega) \in \{0, 1\}$, where $\gamma^{\text{pat}}_{ijpt}(\omega) = 1$ indicates that asset i executing pattern p detects target j given j is located within the coverage area of pattern p at time t.

The complete uncertainty realization under scenario ω is then characterized by

$$\xi(\omega) = \Big(\{\mathcal{M}(\mathcal{T}_j(t, \omega))\}_{j \in J, t \in T},\ \{\gamma^{\text{route}}_{ijat}(\omega)\}_{i \in I, j \in J, a \in A, t \in T},\ \{\gamma^{\text{pat}}_{ijpt}(\omega)\}_{i \in F, j \in J, p \in P, t \in T} \Big),$$

which captures both the spatial uncertainty of all target locations (discretized to areas) across all time periods and the stochastic nature of all detection events. Let $\Xi = \{\xi(\omega) : \omega \in \Omega\}$ denote the set of all scenario realizations.

2.3 Model notation

We define the sets and parameters used in the optimization model formulation in Table 1.

Sets	
I	Set of route-based assets
F	Set of pattern-based assets
J	Set of targets
A	Set of discrete areas (cells)
N_{ia}	Set of areas that asset $i \in I$, when located in area $a \in A$, can move to in the subsequent period
P	Set of patterns
T	Set of time periods
Parameters	
κ_p	Number of buoys required to execute pattern $p \in P$
ν	Total number of buoys available
$\mathfrak{e}_i$	Endurance of asset $i \in F$ (total periods the aircraft can be away from base)

Table 1: Sets and parameters used in the two-stage stochastic program

A pattern $p \in P$ executed by asset $i \in F$ starting at period $t \in T$ is characterized by the pair (A_p, I_{ipt}), where $A_p \subseteq A$ denotes the set of areas covered by pattern p, and $I_{ipt} \subseteq T$ represents the set of active search periods during which asset i actively conducts searches if it starts executing pattern $p \in P$ at time $t \in T$. The active search periods are given by the set,

$$I_{ipt} = \{\, \bar{t} \in T \;\mid\; t + \alpha_{ip} \leq \bar{t} \leq t + \beta_{ip} \,\},$$

where α_{ip} denotes the number of periods from the start of pattern p until asset $i \in F$ begins active search (including transit from base location and buoy deployment time), and β_{ip} denotes the number of periods from the start of pattern p until asset $i \in F$ ceases active search. Figure 1 illustrates the set I_{ipt}.

The parameters α_{ip} and β_{ip} are derived from continuous-time quantities such as transit distance and asset speed, which may not divide evenly into the discrete period length Δt. When the resulting time does not correspond to an integer number of periods, α_{ip} is rounded up to the next whole period, i.e., $\alpha_{ip} = \lceil \tilde{\alpha}_{ip}/\Delta t \rceil$, where $\tilde{\alpha}_{ip}$ is the continuous-time equivalent. The same rounding applies to β_{ip}. This ceiling operation is conservative in that it delays the onset of active search, thereby slightly underestimating the true active search window. Consequently, the resulting deployment plans are feasible with respect to continuous-time

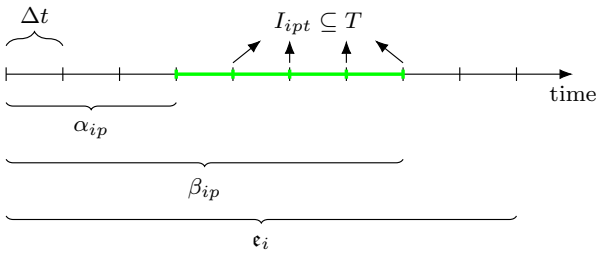

Figure 1: Timeline showing the active search periods $I_{ipt} \subseteq T$ and associated timing parameters for asset i executing pattern p.

asset operations, since the ceiling ensures the asset is never scheduled to search before it physically arrives.

2.4 Two-stage stochastic program

We formulate the search problem as a two-stage stochastic program. The first-stage decisions determine the routes for route-based assets and the pattern selections for pattern-based assets. The first-stage binary decision variables are:

Variable	**Description**
x_{iat}	Equals 1 if asset $i \in I$ searches area $a \in A$ during period $t \in T$
y_{ipt}	Equals 1 if asset $i \in F$ starts pattern $p \in P$ at beginning of period $t \in T$

Unlike general two-stage stochastic programs where the second-stage involves complex recourse actions, our second stage reduces to a counting problem. Once uncertainty is revealed, the second stage optimization problem is structured to count the number of targets detected given the first-stage deployment decisions. The second-stage binary variable z_{jt} indicates whether target $j \in J$ is detected at time $t \in T$ under scenario ξ.

For convenience, we define the following scenario-dependent sets. For each target $j \in J$, time period $t \in T$, and scenario realization $\xi \in \Xi$, let

$$U_{jt\xi} = \left\{(i,a) \in I \times A \ : \ \mathcal{M}(\mathcal{T}_j(t,\xi)) = a, \ \gamma^{\text{route}}_{ijat}(\xi) = 1\right\}$$
$$V_{jt\xi} = \left\{(i,p,t') \in F \times P \times T \ : \ t \in I_{ipt'}, \ \mathcal{M}(\mathcal{T}_j(t,\xi)) \in A_p, \ \gamma^{\text{pat}}_{ijpt}(\xi) = 1\right\},$$

so $U_{jt\xi}$ denotes the set of route-based (asset, area) pairs that can detect target j at time t in scenario ξ, and $V_{jt\xi}$ denotes the set of pattern-based (asset, pattern, start-time) triples that can detect target j at time t in scenario ξ.

We let $Q(x, y, \xi)$ denote the target capture reward given first-stage decisions (x, y) and scenario realization ξ. The two-stage stochastic integer program is then formulated as follows:

$$\max \quad \mathbb{E}_\xi(Q(x,y,\xi)) \tag{2a}$$

$$\text{s.t.} \quad x_{iat+1} \le \sum_{b\in N_{ia}} x_{ibt} \quad \forall i\in I, a\in A, t\in\{1,\dots,\tau-1\}, \tag{2b}$$

$$\sum_{a\in A} x_{iat} = 1 \quad \forall i\in I, t\in T, \tag{2c}$$

$$\sum_{p\in P}\sum_{t'=t}^{t+\mathfrak{e}_i} y_{ipt'} \le 1 \quad \forall i\in F, t\in\{1,\dots,\tau-\mathfrak{e}_i\}, \tag{2d}$$

$$\sum_{i\in F}\sum_{p\in P}\sum_{t\in T} \kappa_p y_{ipt} \le \nu, \tag{2e}$$

$$x_{iat}\in\{0,1\} \quad \forall i\in I, a\in A,\ t\in T, \tag{2f}$$

$$y_{ipt}\in\{0,1\} \quad \forall i\in F, p\in P, t\in T. \tag{2g}$$

The recourse function $Q(x,y,\xi)$ is defined as

$$\max \quad \sum_{j\in J}\sum_{t\in T} r_{jt} z_{jt} \tag{3a}$$

$$\text{s.t.} \quad z_{jt} \le \sum_{(i,a)\in U_{jt\xi}} x_{iat} + \sum_{(i,p,t')\in V_{jt\xi}} y_{ipt'} \quad \forall j\in J, t\in T, \tag{3b}$$

$$\sum_{t\in T} z_{jt} \le 1 \quad \forall j\in J, \tag{3c}$$

$$z_{jt}\in\{0,1\} \quad \forall j\in J, t\in T. \tag{3d}$$

To prevent assets from being positioned near the target starting points, we may impose constraints $x_{ia1} = 0$ for all $i \in I$ and $a \in A' \subset A$, where A' denotes the set of restricted starting areas. These constraints are applied in all computational experiments.

In the first-stage model, constraint (2b) ensures that a route-based asset can occupy area a at time $t+1$ only if it was in a neighboring area at time t. Constraint (2c) requires each route-based asset to occupy exactly one area per period. Constraint (2d) enforces a minimum number of periods between consecutive pattern executions for each pattern-based asset, capturing the endurance limitations of such assets. Constraint (2e) enforces a limit on the total number of buoys used.

In the second-stage model (3), the objective function given in (3a) maximizes the reward of target detections. Constraint (3b) ensures target j is counted as detected only if at least one asset is deployed to an area where a detection event occurs. Constraint (3c) ensures that each target is counted as detected at most once across all time periods. Constraint (3d) defines the detection indicator variable z_{jt} as binary.

3 Solution approach

The presence of the expected value $\mathbb{E}_\xi(Q(x, y, \xi))$ in the objective makes it challenging to solve the two-stage stochastic model exactly. In fact, even evaluating $\mathbb{E}_\xi(Q(x, y, \xi))$ is #P-complete in general (Dyer & Stougie, 2006), which motivates the use of sampling-based methods that exploit the structure of the problem. We therefore apply the Sample Average Approximation (SAA) method (Kleywegt, Shapiro, & Homem-de Mello, 2002), obtaining high-quality solutions by optimizing over a finite set of scenarios drawn from Ξ. We consider $\xi \in \{\xi^1, \ldots, \xi^{|S|}\}$ with $\xi^s \in \Xi, \mathbb{P}(\xi = \xi^s) = p_s$, $s \in S$. We then approximate the expectation by taking a sample average as,

$$\mathbb{E}_\xi[Q(x, y, \xi)] \approx \sum_{s=1}^{|S|} p_s Q(x, y, \xi^s) \stackrel{\text{def}}{=} \sum_{s \in S} p_s Q_s(x, y). \tag{4}$$

For convenience, we also define the sets $U_{jts} \stackrel{\text{def}}{=} U_{jt\xi^s}$ and $V_{jts} \stackrel{\text{def}}{=} V_{jt\xi^s}$ for every $j \in J$, $t \in T$, and $s \in S$.

3.1 Scenario generation

To apply SAA, we sample a finite set of scenarios $\{\xi^1, \ldots, \xi^{|S|}\}$ via Monte Carlo simulation, drawing each scenario independently according to the motion and detection models described below. Each scenario ξ^s specifies the complete realization of target trajectories and detection outcomes.

3.1.1 Motion model

The optimization framework is compatible with arbitrary probabilistic motion models, but for demonstration purposes, in this work, we build the following model for target motion. Each target $j \in J$ has a mission to transit from a starting location to a destination. For each scenario $s \in S$, we sample the set of trajectory realizations $\{\mathcal{T}_j^s\}_{j \in J}$ in continuous space. For each $j \in J$, the initial position $\mathcal{T}_j^s(0) \in \mathbb{R}^2$ is drawn from a probability distribution $P_{j,\text{start}}$ defined over candidate starting locations. The trajectory then progresses at discrete time steps of length Δt toward a destination $d_j \in \mathbb{R}^2$ sampled from the probability distribution $P_{j,\text{end}}$ over candidate destination locations. At each time step t, the speed u_{jt} is drawn from a specified speed distribution for target j.

At each time step $t \in T$, let $\phi_j^s(t)$ denote the bearing angle from the current position $\mathcal{T}_j^s(t)$ to the destination d_j. To model navigation uncertainty, we introduce angular perturbation $\varepsilon_t \sim \mathcal{U}(-\delta_t, \delta_t)$. The resulting direction of motion is therefore the perturbed angle $\tilde{\phi}_j^s(t) = \phi_j^s(t) + \varepsilon_t$, and the target advances a distance $u_{jt}\Delta t$ in that direction. The position at time

$t+1$ is then computed as:

$$\mathcal{T}_j^s(t+1) = \mathcal{T}_j^s(t) + u_{jt}\Delta t \begin{bmatrix} \cos(\tilde{\phi}_j^s(t)) \\ \sin(\tilde{\phi}_j^s(t)) \end{bmatrix}.$$

This procedure repeats iteratively until the planning horizon is reached, generating a complete continuous trajectory $\{\mathcal{T}_j^s(t)\}_{t\in T}$ for scenario s. The discrete area occupied by the target at each time t is then obtained via the region-classification mapping defined in (1), $\mathcal{M}(\mathcal{T}_j^s(t)) \in A$.

3.1.2 Detection model

For each scenario $s \in S$, we sample detection outcomes $\gamma_{ijat}^s \sim \mathsf{BERN}(q_{ijat})$ for route-based assets $i \in I$ and $\gamma_{ijpt}^s \sim \mathsf{BERN}(\hat{q}_{ijpt})$ for pattern-based assets $i \in F$. Building on Koopman (1946), we use an exponential detection model to calculate the detection probabilities

$$q_{ijat} = 1 - e^{-\eta_{ijat}\Delta t}, \quad \text{where } \eta_{ijat} = \frac{w_{ijt}v_i}{\text{Area}(a)}, \quad \forall i \in I, \tag{5}$$

$$\hat{q}_{ijpt} = 1 - e^{-\rho_{ijpt}\Delta t}, \quad \text{where } \rho_{ijpt} = \frac{n^2\pi r^2\hat{\rho}_{ijt}}{\text{Area}(p)}, \quad \forall i \in F, \tag{6}$$

where Δt is the duration of time period t. For route-based assets, v_i is the speed of asset $i \in I$, w_{ijt} is the sweep width (determined by sensor capability of asset $i \in I$, acoustic characteristics of target $j \in J$, and environmental conditions at time $t \in T$), and $\text{Area}(a)$ is the area of cell a. For pattern-based assets, as illustrated in Figure 2, sonobuoys are deployed in an $n \times n$ square grid within a pattern p, with each buoy having a circular detection range of radius r. The total area covered by the n^2 sonobuoys, assuming no overlap in their coverage ranges, is $n^2\pi r^2$, and $\text{Area}(p)$ is the total area of pattern p. The ratio $n^2\pi r^2/\text{Area}(p)$ represents the fraction of the pattern area covered by sonobuoy detection ranges. The parameter $\hat{\rho}_{ijt}$ is the detection rate of each sensor in the pattern against a target of type $j \in J$ during period $t \in T$. The detection rate of the pattern is then ρ_{ijpt} in (6). The detection probabilities depend on the spatiotemporal discretization, specifically the size of the areas $a \in A$, the pattern $p \in P$ and the length of the time period Δt.

This sampling procedure, combined with the motion model, generates a complete scenario realization $\xi^s = \left(\{\mathcal{M}(\mathcal{T}_j^s(t))\}_{j\in J, t\in T},\ \{\gamma_{ijat}^s, \gamma_{ijpt}^s\}_{i,j,a,p,t}\right)$ for each $s \in S$.

3.2 Benders decomposition

After sampling, the SAA form of the optimization model in (2) and (3) can be formulated into a large-scale mixed-integer program, often called the *extensive form* or *deterministic equivalent* (Birge & Louveaux, 2011). For completeness, we provide a description of this model in Appendix A. The optimization model can be solved directly with commercial

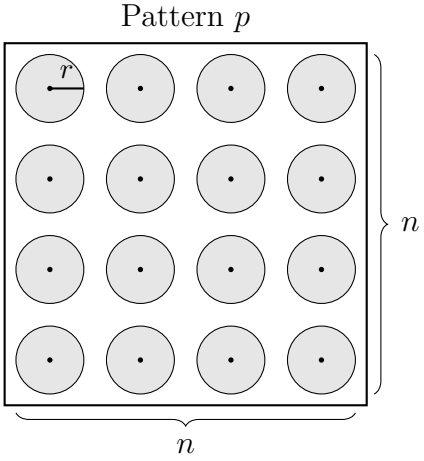

Figure 2: Sonobuoy pattern layout with n^2 buoys arranged in an $n \times n$ grid, each with detection radius r.

or open-source software designed for mixed-integer programs. However, the deterministic equivalent can become difficult to solve directly for large numbers of scenarios due to the size of the model, as we will demonstrate in Section 4.2. As an alternative to the deterministic method, we propose a decomposition method, breaking the problem into a sequence of smaller, more manageable subproblems. We exploit the structure of our problem to use Benders decomposition (Benders, 1962) for solving two-stage stochastic programs.

In Benders decomposition, the optimization problem is partitioned into a master problem and a set of subproblems. Using linear programming duality, the master problem is expressed without explicitly including the subproblem variables. Instead, it incorporates auxiliary variables that represent bounds on the optimal values of the subproblems. These variables are aggregated in the master problem so that, for this maximization problem, the master problem objective value provides an upper bound on the true optimal. At each iteration, the master problem is solved to obtain candidate first-stage decisions, which are fixed in the subproblems. At each iteration, the subproblem solutions generate optimality cuts that are added to the master problem to tighten the bounds. This process continues until an optimal solution is found or the optimality gap falls below a prescribed tolerance.

The primal sub-problem for scenario s, $Q_s(x, y)$, with time-indexed rewards is written as:

$$Q_s(x,y) = \max \quad \sum_{j\in J}\sum_{t\in T} r_{jt}\, z_{jt} \tag{7a}$$

$$\text{s.t.} \qquad z_{jt} \le \sum_{(i,a)\in U_{jts}} x_{iat} + \sum_{(i,p,t')\in V_{jts}} y_{ipt'} \quad \forall j \in J,\ t \in T, \tag{7b}$$

$$\sum_{t\in T} z_{jt} \le 1 \qquad \forall j \in J, \tag{7c}$$

$$z_{jt} \in \{0,1\} \qquad \forall j \in J,\ t \in T \tag{7d}$$

The constraint matrix of (7) is totally unimodular (TU). To see this, observe that each variable z_{jt} appears with coefficient 1 in exactly one constraint of type (7b) and exactly

one of type (7c), so the constraint matrix is the node-arc incidence matrix of a bipartite graph. Such matrices are well known to be TU (Schrijver, 1998). The right-hand side of the subproblem is integer-valued for any integer first-stage solution (x, y). Combined with total unimodularity, this guarantees that every extreme point of the LP relaxation is integral (Hoffman & Kruskal, 2010). The integrality constraints (7d) can therefore be relaxed to $z_{jt} \geq 0$, which permits us to formulate the LP dual of the subproblem and apply Benders decomposition (the L-shaped method) (Van Slyke & Wets, 1969) to solve our stochastic program.

We define the dual variables associated with the constraints (7b) and (7c) as $\boldsymbol{\lambda} = \{\lambda_{jt} |\ j \in J, t \in T\}$ and $\boldsymbol{\mu} = \{\mu_j |\ j \in J\}$, respectively. The dual of the subproblem, $Q_s^d(x, y)$, for each scenario $s \in S$ can be formulated as follows:

$$Q_s^d(x,y) = \min \quad \sum_{j\in J}\sum_{t\in T}\left(\lambda_{jt}\sum_{(i,a)\in U_{jts}} x_{iat} + \lambda_{jt}\sum_{(i,p,t')\in V_{jts}} y_{ipt'}\right) + \sum_{j\in J}\mu_j \tag{8a}$$

$$\text{s.t.} \quad \lambda_{jt} + \mu_j \geq r_{jt} \quad \forall j \in J,\ t \in T, \tag{8b}$$

$$\lambda_{jt} \geq 0 \quad \forall j \in J,\ t \in T, \tag{8c}$$

$$\mu_j \geq 0 \quad \forall j \in J \tag{8d}$$

The expected reward is computed via (4), where $Q_s(x, y)$ is obtained by solving the primal subproblem (7) for scenario $s \in S$. A two-stage stochastic program is said to have *relatively complete recourse* if for every feasible first-stage decision and every uncertain realization, the second-stage problem admits a feasible solution. The model satisfies this property since for any feasible first-stage solution, $z_{jt} = 0\ \forall j \in J, t \in T$ is always feasible in the second stage. The objective is also bounded by the maximum total reward. By strong duality, both primal (7) and dual subproblems (8) are feasible and bounded, and thus have optimal solutions.

We introduce variables θ_s such that $\theta_s \leq Q_s^d(x, y),\ \forall s \in S$. Let Π_s denote the polyhedron defined by the constraints of the dual problem (8) for scenario s, and let $\mathbb{V}(\Pi_s)$ denote its set of extreme points. Our model in (2) and (3) is reformulated as follows:

$$\max \sum_{s\in S} p_s\theta_s \tag{9}$$

$$s.t.\ \text{(2b)} - \text{(2g)}\ \text{holds} \tag{10}$$

$$\theta_s \leq \sum_{j\in J}\sum_{t\in T}\left(\sum_{(i,a)\in U_{jts}} x_{iat} + \sum_{(i,p,t')\in V_{jts}} y_{ipt'}\right)\lambda_{jt} + \sum_{j\in J}\mu_j \quad \forall(\boldsymbol{\lambda},\boldsymbol{\mu}) \in \mathbb{V}(\Pi_s),\ \forall s \in S \tag{11}$$

The reformulation (9)–(11) contains exponentially many Benders cuts. A natural solution

approach is to start from a relaxed model where these constraints are dropped. Iteratively, we add cuts to the problem as violated constraints are found. We solve this reformulation using a Benders decomposition method implemented within a branch-and-cut framework.

3.3 Benders branch-and-cut algorithm

We propose to solve the Benders master problem (9)- (10) with a branch-and-cut algorithm as in (Fortz & Poss, 2009; Naoum-Sawaya & Elhedhli, 2013). The algorithm proceeds in two phases. In the first phase, we solve the LP relaxation via classical Benders decomposition (Benders, 1962), retaining all generated cuts to initialize the cut pool. In the second phase, the branch-and-cut is initialized with these cuts. At each node in the branch and bound tree, the LP relaxation is solved, and the algorithm branches on a fractional variable if the solution is not integer feasible. When an integer-feasible solution $(\hat{x}, \hat{y})$ is found, we solve the scenario subproblems to check whether $\hat{\theta}_s \leq Q_s(\hat{x}, \hat{y})$ for all $s \in S$ (Fortz & Poss, 2009). If violated Benders cuts are found, they are added as lazy constraints. If no violated cuts exist, $(\hat{x}, \hat{y})$ is accepted as feasible and its value $Q_s(\hat{x}, \hat{y})$ used to prune the tree.

Benders cuts approximate the value function of each scenario. We consider both multi-cut and single-cut variants of Benders decomposition. In the multi-cut variant, each scenario generates its own cut as in (11). Each iteration may generate one cut per scenario, i.e., up to $|S|$ cuts. In the single-cut variant, the per-scenario variables θ_s are replaced by a single variable Θ representing the expected recourse value across all scenarios. The objective (9) becomes max Θ, and (11) is replaced by the single aggregated cut in (12),

$$\Theta \leq \sum_{s \in S} p_s \left[\sum_{j \in J} \sum_{t \in T} \left(\sum_{(i,a) \in U_{jts}} x_{iat} + \sum_{(i,p,t') \in V_{jts}} y_{ipt'} \right) \lambda^s_{jt} + \sum_{j \in J} \mu^s_j \right], \tag{12}$$

so that each iteration adds exactly one cut to the master problem. The single-cut approach, as shown in Section 4.3, performs poorly for our problem instances.

3.4 Closed form for Benders cuts

In this section, we present the derivation of the closed-form solution for the second-stage subproblem. This closed-form solution avoids the need to call an LP solver to evaluate (7) or its dual (8).

Definition 1 (Coverage)**.** *For a given first-stage solution $(\hat{x}, \hat{y})$ and scenario s, define the* coverage *of target j at time t with respect to $(\hat{x}, \hat{y}, s)$ as*

$$c_{jt}(\hat{x}, \hat{y}, s) = \sum_{(i,a) \in U_{jts}} \hat{x}_{iat} + \sum_{(i,p,t') \in V_{jts}} \hat{y}_{ipt'}.$$

Definition 2 (Critical index)**.** *For a given first-stage solution $(\hat{x}, \hat{y})$ and scenario s, define*

the critical index *for target j with respect to $(\hat{x}, \hat{y}, s)$ as*

$$k_j^*(\hat{x}, \hat{y}, s) = \max\left\{ k \in \{0, 1, \ldots, |T|\} : \sum_{t=1}^{k} c_{jt}(\hat{x}, \hat{y}, s) \leq 1 \right\}.$$

Theorem 1 (Closed-form solution for the time-indexed reward subproblem)**.** *For a first-stage solution $(\hat{x}, \hat{y})$ and scenario s, let $c_{jt}(\hat{x}, \hat{y}, s)$ denote the coverage of target j at time t, and let $k_j^*(\hat{x}, \hat{y}, s)$ denote the critical index defined in Definitions 1–2. Assume without loss of generality that time periods are ordered so that the rewards are monotonically decreasing, i.e., $r_{j1} \geq r_{j2} \geq \cdots \geq r_{j|T|}$. Then an optimal solution to the primal* (7) *is*

$$z_{jts}^* = \begin{cases} c_{jt}(\hat{x}, \hat{y}, s) & \text{if } t \leq k_j^*(\hat{x}, \hat{y}, s), \\ 1 - \sum_{t'=1}^{k_j^*(\hat{x}, \hat{y}, s)} c_{jt'}(\hat{x}, \hat{y}, s) & \text{if } t = k_j^*(\hat{x}, \hat{y}, s) + 1, \\ 0 & \text{if } t > k_j^*(\hat{x}, \hat{y}, s) + 1, \end{cases} \tag{13}$$

and an optimal solution to the dual (8) *is*

$$\mu_{js}^* = \begin{cases} r_{j,k_j^*(\hat{x}, \hat{y}, s)+1} & \text{if } k_j^*(\hat{x}, \hat{y}, s) < |T|, \\ 0 & \text{if } k_j^*(\hat{x}, \hat{y}, s) = |T|, \end{cases} \tag{14}$$

$$\lambda_{jts}^* = \begin{cases} r_{jt} - \mu_{js}^* & \text{if } t \leq k_j^*(\hat{x}, \hat{y}, s), \\ 0 & \text{if } t > k_j^*(\hat{x}, \hat{y}, s). \end{cases} \tag{15}$$

Proof. We consider three cases: $k_j^*(\hat{x}, \hat{y}, s) = 0$, $0 < k_j^*(\hat{x}, \hat{y}, s) < |T|$, and $k_j^*(\hat{x}, \hat{y}, s) = |T|$. It is easy to verify that the proposed solutions satisfy primal and dual feasibility for all cases (shown in Appendix B). We are left to show optimality, which we do by showing that the primal and dual objective values are equal.

For notational convenience in the proof, we write k_j^* for $k_j^*(\hat{x}, \hat{y}, s)$ and c_{jt} for $c_{jt}(\hat{x}, \hat{y}, s)$. The primal objective is:

$$\begin{aligned} Q_s(\hat{x}, \hat{y}) &= \sum_{j \in J} \sum_{t \in T} r_{jt}\, z_{jts}^* \\ &= \sum_{j : k_j^* = 0} r_{j1} + \sum_{j : 0 < k_j^* < |T|} \left[\sum_{t=1}^{k_j^*} r_{jt} c_{jt} + r_{j,k_j^*+1} \left(1 - \sum_{t'=1}^{k_j^*} c_{jt'} \right) \right] + \sum_{j : k_j^* = |T|} \sum_{t=1}^{|T|} r_{jt} c_{jt} \end{aligned}$$

The dual objective is:

$$\begin{aligned} Q_s^d(\hat{x}, \hat{y}) &= \sum_{j \in J} \sum_{t \in T} c_{jt} \lambda_{jts}^* + \sum_{j \in J} \mu_{js}^* \\ &= \sum_{j \in J : k_j^* = 0} r_{j1} + \sum_{j \in J : 0 < k_j^* < |T|} \left[\sum_{t=1}^{k_j^*} c_{jt}(r_{jt} - r_{j,k_j^*+1}) + r_{j,k_j^*+1} \right] + \sum_{j \in J : k_j^* = |T|} \sum_{t=1}^{|T|} c_{jt} r_{jt} \\ &= Q_s(\hat{x}, \hat{y}) \end{aligned}$$

Since both solutions are feasible and attain equal objective values, optimality follows from strong duality. □ □

The Benders cut is obtained by substituting the optimal dual solution (λ^*, μ^*) from Theorem 1 into constraint (11), giving:

$$\theta_s \leq \sum_{\substack{j \in J: \\ k_j^* = 0}} r_{j1} + \sum_{\substack{j \in J: \\ 0 < k_j^* < |T|}} \left[\sum_{t=1}^{k_j^*} r_{jt} c_{jt} + r_{j,k_j^*+1} \left(1 - \sum_{t=1}^{k_j^*} c_{jt} \right) \right] + \sum_{\substack{j \in J: \\ k_j^* = |T|}} \sum_{t=1}^{|T|} r_{jt} c_{jt} \tag{16}$$

where $k_j^* = k_j^*(\hat{x}, \hat{y}, s)$ is fixed from the incumbent solution, and

$$c_{jt} = c_{jt}(x, y, s) = \sum_{(i,a) \in U_{jts}} x_{iat} + \sum_{(i,p,t') \in V_{jts}} y_{ipt'}$$

is linear in the master-problem variables (x, y).

The closed-form expression in (16) allows us to generate Benders cuts directly without solving the second stage for each scenario, replacing LP solves with closed-form evaluations within the branch-and-cut framework.

4 Computational experiments

We assess the performance of the proposed model and solution approach through a series of computational experiments, organized as follows. We begin by comparing the deterministic formulation and the branch-and-cut decomposition algorithm described in Section 4.2, confirming that decomposition method often requires significantly less computational effort. In Section 4.3 we show that the multi-cut approach dominates the single-cut variant, which often fails to converge within computational limits. Based on these results, subsequent experiments employ the multi-cut strategy and the decomposition approach.

In Section 4.4, we analyze the impact of spatial discretization, assessing the trade-offs between computational cost and solution fidelity. Having established the appropriate solution method and modeling resolution, we then evaluate the quality of our solutions in Section 4.5 and compare our approach against a heuristic baseline in Section 4.6. Finally, we demonstrate the applicability of our model as a strategic analysis and decision-support tool by quantifying the marginal value of deploying additional resources in Section 4.7.

4.1 Experimental setup and implementation details

We implemented our models and algorithms in Python 3.12.0 and solved them using Gurobi 10.0.1. To ensure a consistent comparison between experimental results, we use Gurobi's *work unit* as the measure of computational effort, where 1 work unit $\approx$ 1 *second* according to Gurobi's documentation. The branch-and-cut algorithm was implemented via a callback function that dynamically adds Benders cuts during the branch and bound process. Experiments in Sections 4.2 and 4.5 were conducted on the UW-Madison Center for High Throughput Computing (CHTC) cluster with a work unit limit of 20,000, while all other experiments were run on a MacBook Air with an M2 chip and 16GB RAM with a work unit limit of 3,600.

We examine the effect of spatial discretization in Section 4.4, considering cell sizes of 6×6, 12×12, and 24×24 square nautical miles. Based on the analysis described in Section 4.4, we adopt the 12×12 resolution as the spatial discretization for the model for experiments described outside of that section. The planning horizon consists of discrete time periods of length $\Delta t = 1$ hour. To model operational constraints on initial asset positioning, we restrict route-based assets from occupying cells in $A' \subset A$ at $t = 1$, enforced by setting $x_{ia1} = 0$ for all $i \in I$ and $a \in A'$.

We generate scenarios for our computational experiments using the framework described in Section 3.1. Target trajectories are sampled according to the motion model described in Section 3.1.1. Detection outcomes are sampled according to the detection model described in Section 3.1.2, with several simplifying assumptions. First, detection probabilities are assumed to be independent of target characteristics, constant over time, and uniform across all grid cells. Second, for this unclassified demonstration, we also assume the same sweep width w_i and speed v_i for all assets in I. Under these assumptions, all route-based assets share a common detection probability q, computed according to Equation (5). Although q is homogeneous across all (i, j, a) pairs, detection outcomes are realized by independent Bernoulli sampling for each pair within each scenario $s \in S$.

For the reward structure, we assume r_{jt} is identical across all targets $j \in J$, depending only on time, so we write r_t in place of r_{jt}. To prioritize earlier detections, we set $r_t = |T| - t + 1$, which assigns linearly decreasing rewards over the planning horizon.

In this demonstration, each pattern-based asset $i \in F$ deploys sonobuoys in a square arrangement, forming a sonobuoy pattern. Each pattern consists of 25 sonobuoys arranged in a 5×5 grid, with each buoy having sensor radius $r = 1.5$ nautical miles, spanning an area of 24×24 square nautical miles. Patterns are positioned across the search area with horizontal spacing equal to the pattern side length and vertical spacing equal to half the pattern side length, creating overlapping coverage in the vertical direction. Under the 12×12 discretization, each pattern covers four grid cells. In a practical implementation, α_{ip} would depend on the MPRA base location in relation to the pattern search area. In this example, we set the active search window parameters as $\alpha_{ip} = 3$ and $\beta_{ip} = 7$ time periods for all (asset, pattern) tuples. The detection rates for sonobuoys is set to $\hat{\rho}_{ijt} = 1/\text{hour}$ for all targets and times.

Since SAA optimizes over a finite scenario sample, the resulting objective value z^* is a biased estimator of the true optimum. We refer to z^* as the perceived objective value. To obtain unbiased performance estimates, we evaluate solutions on an independent set of scenarios, reporting evaluated objective values, as described in Section 4.5.

In Sections 4.2-4.5, we analyze two test instances involving the search for two targets. The first instance consists of one submarine and one P-8 aircraft with a $216 \times 216nm$ search area over 20 time periods. The second instance consists of two submarines and one P-8 aircraft with a $168 \times 168nm$ search area over 15 time periods. Both instances use a 12×12 square nautical mile grid cell discretization. Instance 1 has an 18×18 grid, while Instance 2 has a 14×14 grid. Table 2 summarizes all experimental parameters.

Table 2: Experimental parameters

Parameter	**Value**	**Description**
Motion Model		
u_{jt}	Tri$(6, 7, 8)$ knots	Target transit speed for target $j \in J$ at time t
δ_t	$5 \cdot \max(\min(d_t/5,\ 12),\ 3)^\circ$	Maximum angular deviation bound*
Detection Model: Route-based assets		
w	3 nm	Sweep width
v	10 knots	Asset speed
Area(a)	144 nm^2	Area of a grid cell
Detection Model: Pattern-based assets		
n^2	25	Number of sonobuoys per pattern
r	1.5 nm	Sonobuoy sensor radius
Area(p)	576 nm^2	Area of a pattern
Model parameters		
Δt	1 hour	Time period length
r_t	$\lvert T\rvert - t + 1$	Reward for detecting a target at time $t \in T$
α	3 hours	Transit and deployment time of all patterns
β	7 hours	Time from pattern start to end of active search
κ_p	25	Number of buoys in pattern $p \in P$
ν	500	Total sonobuoy capacity
$\hat{\rho}$	1/hour	Sonarbuoy detection rate
$\mathfrak{e}_i$	9 hours	Endurance of the asset $i \in F$

*Here d_t is the distance of the target to its destination at time t. This produces perturbations in the heading direction in the range $[\pm 15^\circ, \pm 60^\circ]$, with smaller deviations near the destination.

4.2 Decomposition method versus extensive form

In our first experiment, we compare the computational efficiency of the branch-and-cut decomposition algorithm against the extensive form formulation by comparing the number of Gurobi work units required to solve *Instance 1* and *Instance 2* as a function of the number of scenarios. Figures 3 and 4 present results for *Instance 1* and *Instance 2*, respectively. For each instance, we vary the number of scenarios, perform 30 independent runs per scenario

size, and plot the average number of work units required. The corresponding numerical data for these figures are provided in Table 8 in Appendix C.

The results demonstrate that the decomposition method requires substantially less computational effort than the extensive form formulation, as shown in Figures 3 and 4. For both instances, the gap in work units increases as the number of scenarios increases. *Instance 2* requires more work units overall due to the larger problem size.

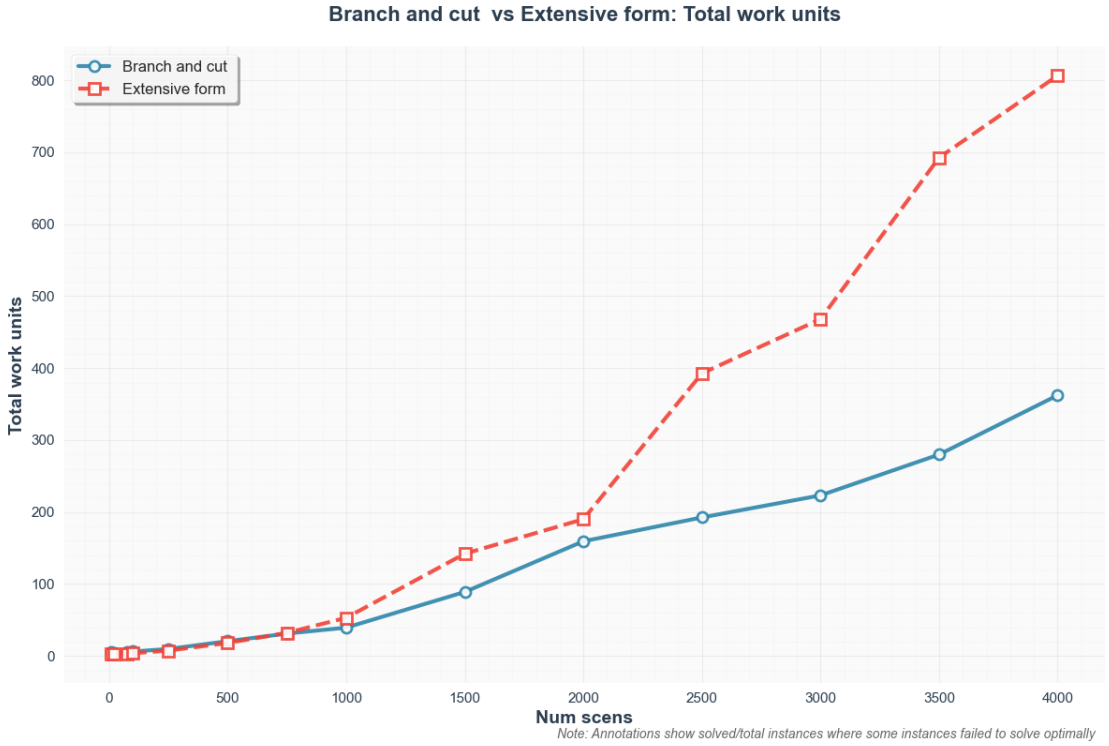

Figure 3: Comparison of work units for the decomposition method versus extensive form for Instance 1

For larger scenario sizes in *Instance 2*, not all runs terminated within the Gurobi work unit limit of 20,000. For scenario sizes beyond 1500, the fractions shown on the graph indicate the proportion of runs that solved to optimality within the work unit limit of 20,000. For instance, at 4000 scenarios, the ratio 1/30 indicates that only 1 out of 30 runs reached optimality within the work unit limit. These results confirm that the decomposition approach scales better than the extensive form and enables solution of problem instances that would otherwise be computationally intractable.

4.3 Single-cut and multi-cut Benders decomposition

We compare the computational performance of single-cut and multi-cut Benders decomposition algorithms using *Instance 1* and *Instance 2* with 1000 scenarios. Table 3 summarizes the results of a single run for each instance. These results demonstrate the computational advantage of multi-cut decomposition. While both methods solve the smaller *Instance 1* to optimality, multi-cut requires substantially fewer work units. For *Instance 2*, the single-cut

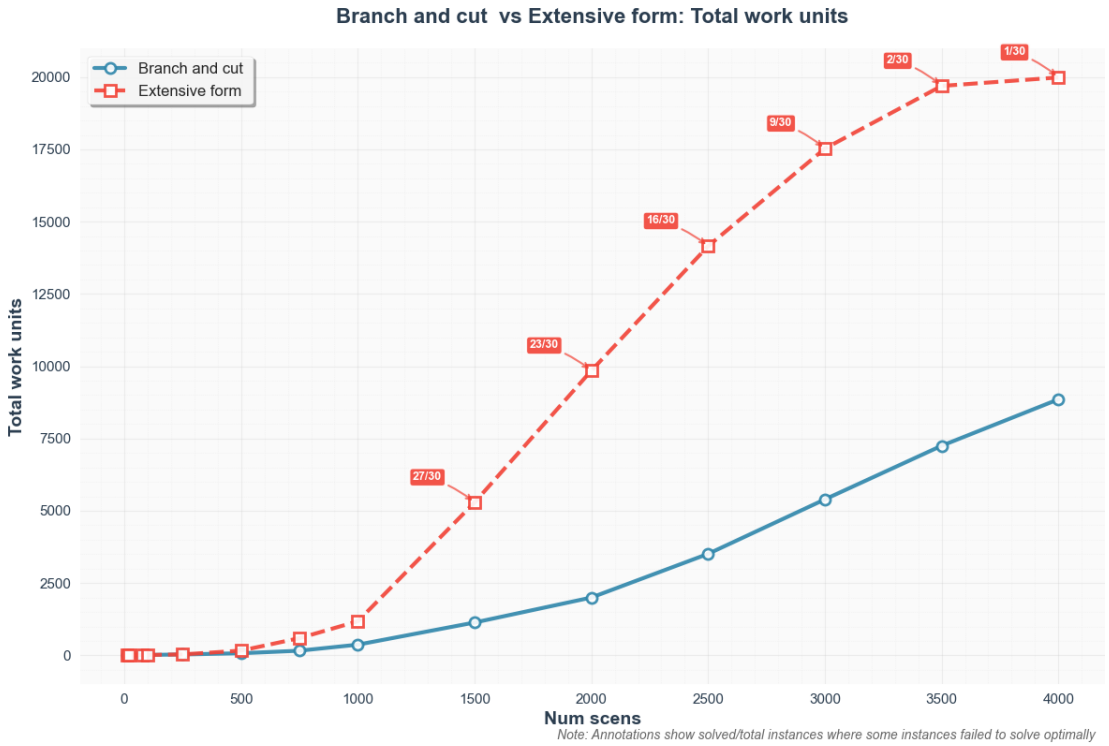

Figure 4: Comparison of work units for the decomposition method versus extensive form for Instance 2

method did not converge within the work unit limit of 3,600, terminating with a 33.1% optimality gap. As problem size increases, single-cut decomposition fails to converge within reasonable time limits, making multi-cut the practical choice for large-scale instances.

	Instance 1		Instance 2	
	multi-cut	single-cut	multi-cut	single-cut
Work units	32.46	2207.35	852.53	3600+
Gap at work unit limit	0%	0%	0%	33.1%

Table 3: Comparison of multi-cut and single-cut Benders decomposition

4.4 Impact of spatial discretization

Discretization plays a critical role in optimization models when continuous dimensions such as space or time must be represented in a finite discrete form. Finer discretization increases modeling accuracy but leads to substantially higher computational cost, whereas coarser discretization reduces runtime at the expense of spatial detail and solution quality.

In this implementation, we consider three discretization levels with cell sizes of 6×6, 12×12, and 24×24 square nautical miles. Table 4 summarizes the parameters for each level. Pattern-based assets cover a fixed physical area of $24 \times 24 = 576$ nm^2, so the number of grid cells spanned by a pattern varies with resolution.

Cell Size (nm^2)	Grid cell area Area(a) (nm^2)	Grid cells per pattern	Grid dimensions
6×6	36	16	28×28
12×12	144	4	14×14
24×24	576	1	7×7

Table 4: Discretization parameters for each grid resolution.

The choice of discretization level impacts detection probabilities. For route-based assets, q is computed from (5), so detection probability decreases as cell area Area(a) increases. For pattern-based assets, $\hat{q}$ is computed from (6) and remains constant at 0.264 since each pattern covers a fixed 576 nm^2 area regardless of discretization. The parameter values used in both expressions are given in Table 2.

We solve each discretization level independently to obtain solutions optimized for their respective grids. To compare solutions across resolutions, we evaluate each on two common grids: a 12×12 grid and a 6×6 grid. When converting a solution to a finer evaluation grid, each coarse cell is disaggregated by randomly selecting one of its contained fine-grid cells as the patrol area. When converting to a coarser grid, fine-grid cells are aggregated into their containing coarse cell. Evaluating on both grids allows us to assess whether solution quality rankings are robust to the choice of evaluation resolution.

Table 5 presents the computational work units and 95% confidence intervals for the evaluated objective values for *Instance 1*, where each model is solved using 2000 scenarios and evaluated out-of-sample on 5000 independent scenarios (the evaluation procedure is described in Section 4.5). The 12×12 discretization achieves the highest evaluated objective value on both grids at a fraction of the computational cost of 6×6. The 6×6 discretization requires substantially more computational effort yet attains a lower evaluated objective value on both grids. The 24×24 discretization is the least expensive but performs poorly regardless of evaluation grid, as targets that appear co-located with assets often occupy different parts of the same large cell. We therefore adopt the 12×12 resolution for the remaining experiments.

Cell Size	Perceived Objective	Work	Evaluated Objective Value (95% CI)	
(nm^2)	Value (z^*)	Units	On 12×12 grid	On 6×6 grid
6×6	6.896	2373.09	$[6.714, 7.098]$	$[6.356, 6.732]$
12×12	7.922	201.28	$[7.228, 7.618]$	$[6.675, 7.055]$
24×24	7.340	10.98	$[2.315, 2.576]$	$[1.618, 1.841]$

Table 5: Perceived objective value, computational work units, and 95% confidence intervals for evaluated objective values across discretization levels, evaluated on two common grids.

4.5 Solution quality of the stochastic model

Several statistical procedures have been proposed for assessing the quality of solutions obtained from sampling-based stochastic optimization methods. The Monte Carlo bounding

framework (Mak, Morton, & Wood, 1999) estimates statistical bounds on the true optimal value by solving multiple independent SAA replications and evaluating candidate solutions on independent samples. Related Monte Carlo procedures for constructing confidence intervals on the optimality gap of candidate solutions have also been developed (Bayraksan & Morton, 2006). Sequential sampling procedures provide another alternative, in which a sequence of candidate solutions is evaluated with increasing sample sizes and the procedure terminates when the estimated optimality gap satisfies a prescribed statistical criterion (Bayraksan & Morton, 2011).

When solving stochastic programs via Sample Average Approximation (SAA), the optimal objective value obtained from a finite scenario set is a biased estimator of the true optimal value (Mak et al., 1999). This bias arises because the optimization selects the best solution for the sampled scenarios, which may not generalize to the full distribution. To quantify solution quality, we compute statistical bounds on the true optimal solution value through repeated sampling and out-of-sample evaluation.

For the upper bound estimation, we solve the SAA problem $N = 30$ times for each scenario set size $n = |S|$, where each replication uses an independently generated scenario set. Let $\hat{z}_n^k$ denote the optimal objective value from the k-th replication for $k = 1, \ldots, N$ with scenario set size n. The sample mean $\bar{z}_n = \frac{1}{N}\sum_{k=1}^{N} \hat{z}_n^k$ serves as an estimator of the upper bound on the true optimal value for scenario set size n. We construct a 95% confidence interval using the sample standard deviation of these N objective values.

For lower bound estimation, we evaluate the out-of-sample performance of a candidate solution. From the N solutions obtained above, we randomly select one solution $(\bar{x}, \bar{y})$ and fix these first-stage decisions. We then generate an independent evaluation set $\mathcal{S}_{\text{eval}}$ containing $|\mathcal{S}_{\text{eval}}| = 5000$ scenarios. For each evaluation scenario $s \in \mathcal{S}_{\text{eval}}$, we compute the number of targets detected given the fixed decisions $(\bar{x}, \bar{y})$. The average detection count across all evaluation scenarios provides an unbiased estimate of the expected objective value for that particular solution. This estimate serves as a lower bound on the true optimal value. We construct a 95% confidence interval for this lower bound using the sample standard deviation of the detection counts across the evaluation scenarios.

By comparing the upper and lower bounds across varying sample sizes, we quantify the gap and assess the quality of the stochastic approximation. A narrow gap between the bounds indicates a high-quality solution, whereas a wider gap suggests that a larger scenario sample is needed to obtain solutions of better quality.

Figures 5 and 6 show the convergence results for *Instance 1* and *Instance 2*, where the expected upper bound corresponds to the average perceived objective value across replications and the out-of-sample evaluator corresponds to the evaluated objective value. As the number of scenarios increases, the confidence intervals narrow and the gap between bounds decreases. The expected upper bound starts high when few scenarios are used and decreases as more scenarios are incorporated into the optimization. The out-of-sample evaluator results remain relatively stable across different scenario counts. For both instances, the bounds

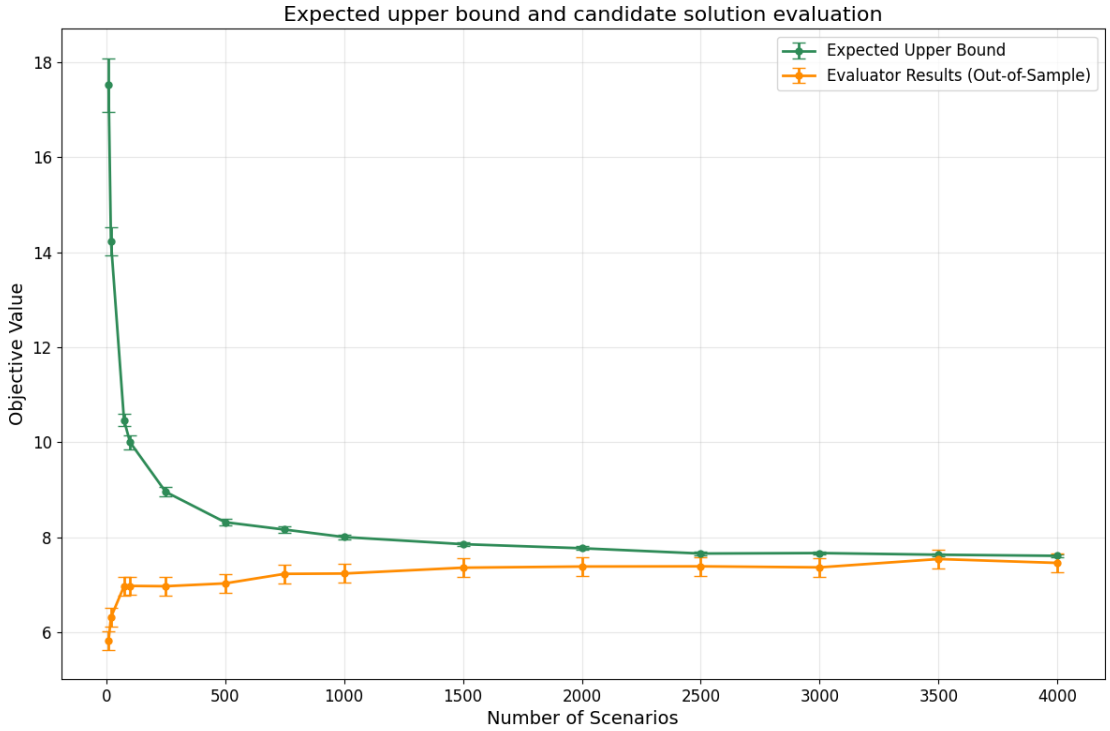

Figure 5: Convergence of the biased and evaluated objective value estimates toward the true optimal value for Instance 1

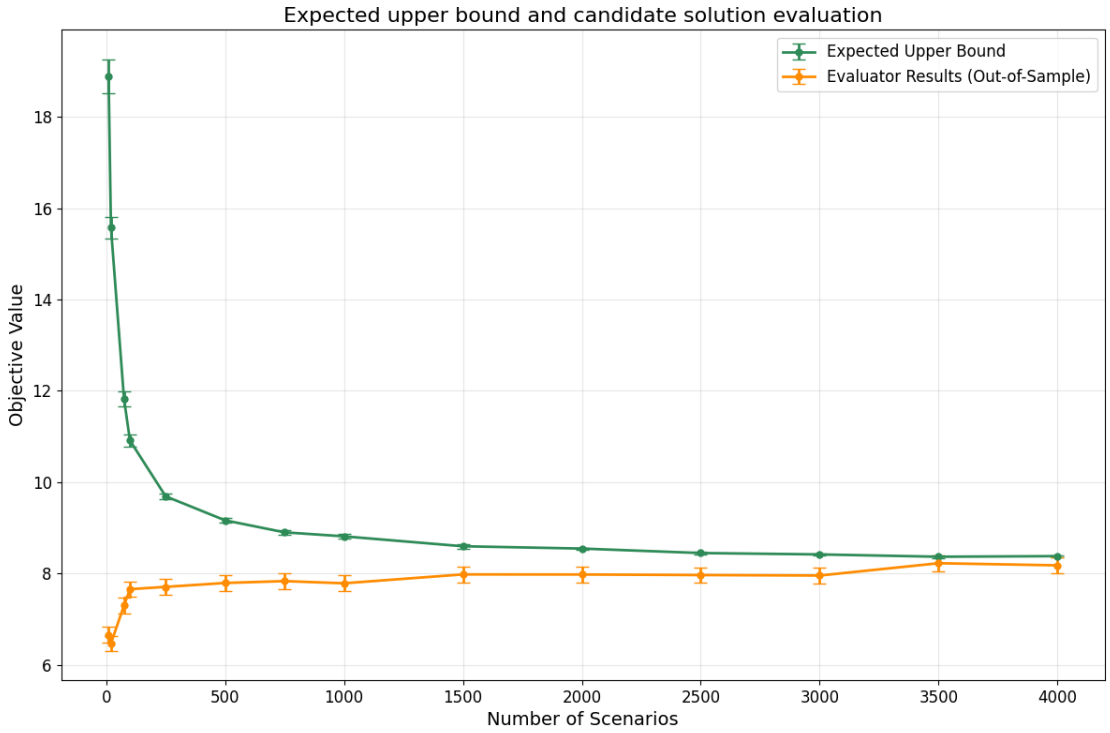

Figure 6: Convergence of the biased and evaluated objective value estimates toward the true optimal value for Instance 2

converge around $|S| = 1500$ to 2000 scenarios, at which point the gap between the upper bound and the out-of-sample evaluation becomes negligible. Table 9 in Appendix D has the corresponding numerical values used in the figures.

4.6 Comparison with greedy baseline heuristic

We compare our stochastic programming approach against a greedy heuristic that makes deployment decisions similar to current operational practice. To ensure a fair comparison, the greedy method uses identical scenario data and operational constraints. The heuristic first converts the scenario data into asset-specific cell weights. For asset i, cell a, time t, and scenario s, the weight w_{at}^{is} equals the number of targets that asset i would detect if it searched cell a at time t under scenario s:

$$w_{at}^{is} = \begin{cases} \left|\left\{j \in J : \mathcal{M}(\mathcal{T}_j^s(t)) = a,\, \gamma_{ijat}^s = 1\right\}\right| & \text{if } i \in I, \\ \left|\left\{j \in J : \mathcal{M}(\mathcal{T}_j^s(t)) = a,\, \exists p \in P \text{ with } a \in A_p,\, \gamma_{ijpt}^s = 1\right\}\right| & \text{if } i \in F. \end{cases}$$

The aggregate weight $w_{at}^{i} = \sum_{s \in S} w_{at}^{is}$ measures the total detection opportunity available to asset i in cell a at time t across the scenarios.

The greedy heuristic schedules assets sequentially, placing route-based assets before pattern-based assets. Route-based assets are placed first because they must move through adjacent cells over time, making their placement far more rigid than pattern-based assets, which can be assigned to any feasible pattern. At each time step, a route-based asset chooses the highest-weight cell among its current cell and neighboring cells. After all route-based assets are scheduled, each pattern-based asset selects the feasible pattern with the largest total weight over its active search window $[t + \alpha_{ip}, t + \beta_{ip}]$. The pattern-based asset then becomes unavailable for its flight duration before another pattern can be selected.

The algorithm maintains a set C of (time, area) pairs whose detection credit has already been claimed by a deployed asset. Before each deployment decision, weights at pairs in C are set to zero to prevent double-counting. Route-based assets are restricted from selecting any pair already in C, which prevents two submarines from occupying the same cell at the same time. Pattern-based assets are additionally restricted from selecting any pattern whose coverage overlaps spatially with pairs already in C. Algorithm 1 summarizes the procedure.

We evaluate both methods on two test instances using 1000 scenarios for optimization and 5000 independent scenarios for unbiased evaluation. Table 6 reports the out-of-sample performance.

The stochastic program achieves substantial improvements over the greedy baseline, with a 21.1% improvement for Instance 1 and 46.5% for Instance 2. The greedy method's sequential decisions fail to coordinate assets or anticipate future detection opportunities across scenarios. In contrast, our approach jointly optimizes all deployments while hedging against uncertainty through the stochastic programming framework.

Input: Cell weights $\{w^i_{at}\}$ for each asset i
Output: Submarine paths $(a^i_t)_{t\in T}$ for each $i \in I$
Pattern assignments (p^*, t_{start}) for each $i \in F$

$C \leftarrow \emptyset$, $b \leftarrow 0$;

for *each asset* $i \in I$ *(route-based)* **do**
  Set $w^i_{at} \leftarrow 0$ for all $(t, a) \in C$;
  $a_0 \leftarrow \arg\max_{a\in A,\,(0,a)\notin C} \; w^i_{a0}$;
  $C \leftarrow C \cup \{(0, a_0)\}$;
  $w^i_{a_0,0} \leftarrow 0$;
  for $t = 1, \ldots, \tau$ **do**
    $a_t \leftarrow \arg\max_{a\in N_{ia_{t-1}}\cup\{a_{t-1}\},\,(t,a)\notin C} \; w^i_{at}$;
    $C \leftarrow C \cup \{(t, a_t)\}$;
    $w^i_{a_t,t} \leftarrow 0$;
  end
end

for *each asset* $i \in F$ *(pattern-based)* **do**
  Set $w^i_{at} \leftarrow 0$ for all $(t, a) \in C$;
  $t \leftarrow 0$;
  while $t \le \tau$ *and* $b + \kappa_p \le \nu$ *for some* p **do**
    $p^* \leftarrow \arg\max_{p\in P} \displaystyle\sum_{t'=t+\alpha_{ip}}^{t+\beta_{ip}} \sum_{a\in A_p} w^i_{at'}$;
      subject to $A_p \cap \{a \mid (t', a) \in C\} = \emptyset$ for all $t' \in [t + \alpha_{ip}, t + \beta_{ip}]$;
    $C \leftarrow C \cup \{(t', a) \mid t' \in [t + \alpha_{ip^*}, t + \beta_{ip^*}],\, a \in A_{p^*}\}$;
    Set $w^i_{at'} \leftarrow 0$ for all newly added $(t', a) \in C$;
    $b \leftarrow b + \kappa_{p^*}$;
    $t \leftarrow t + \mathfrak{e}_i$;
  end
end

Algorithm 1: Greedy baseline heuristic

Method	Instance 1	Instance 2
Greedy	5.82	5.30
Stochastic Program	7.04	7.77
Improvement	21.1%	46.5%

Table 6: Evaluated objective values for greedy heuristic and stochastic program.

4.7 Marginal gain in asset deployment

The optimization model (2)-(3) is designed as a tool to inform both strategic and operational decision making. We demonstrate this by examining Instance 2, in which two targets are searched over 15 time periods, across four fleet configurations that vary the number of route-based and pattern-based assets. Table 7 reports the evaluated objective value, expected number of detections, and median detection time period for each configuration. Solutions were obtained using 1000 scenarios and evaluated on 5000 scenarios.

Number of Route-based Assets $\|I\|$	Number of Pattern-based Assets $\|F\|$	Evaluated Objective Value	Expected Number of Detections	Median Detection Time Period
1	1	5.665	0.75	6
2	1	7.976	0.94	6
1	2	8.500	1.06	5
2	2	10.702	1.238	5

Table 7: Evaluated objective values across fleet configurations.

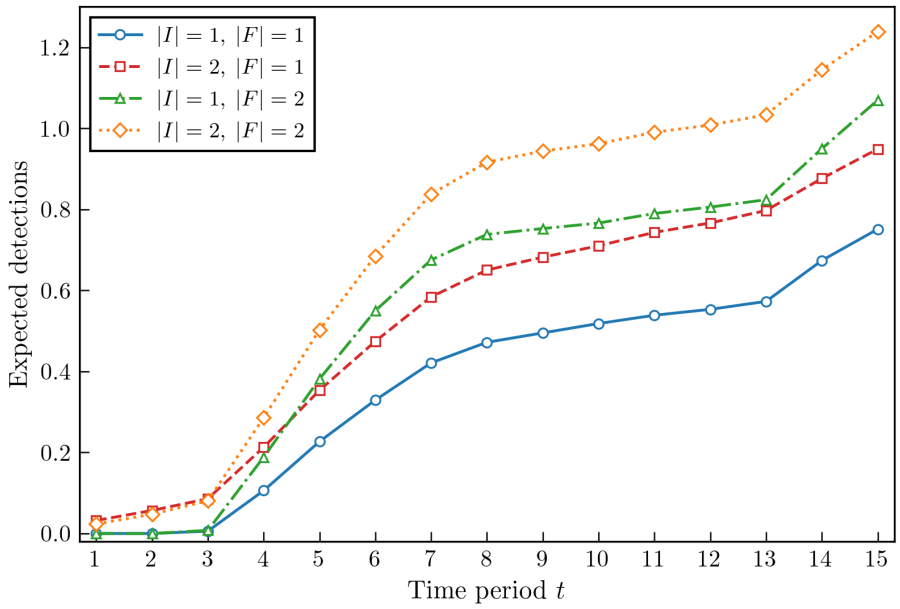

Figure 7: Cumulative expected detections over time for each fleet configuration.

Table 7 and Figure 7 show the cumulative expected detections over the planning horizon for each fleet configuration. Configurations with more assets accumulate detections at a higher rate, and the gap between configurations widens over the planning horizon. The (2,2) configuration achieves the highest detection rate throughout the planning horizon, while the (1,1) configuration shows the slowest accumulation.

By comparing objective values and detection rates across fleet configurations, decision-makers can assess the operational benefit of each additional asset and allocate resources accordingly.

5 Conclusion and future work

We presented a two-stage stochastic integer programming model for optimizing asset deployment in anti-submarine warfare operations. The model maximizes the expected number of targets detected under uncertainty in both target motion and sensor performance. A key contribution is the derivation of a closed-form solution for the second-stage subproblem, which enables efficient generation of Benders cuts without explicitly solving subproblems. This allows the branch-and-cut decomposition algorithm to solve large-scale instances that are intractable with the extensive form.

Computational experiments confirm that the branch-and-cut decomposition requires substantially less computational effort than the extensive form, and that the multi-cut variant dominates the single-cut approach, which fails to converge on larger instances. The stochastic programming solutions attain improvements of 21% and 46% over a greedy baseline on the two test instances, and the bounds obtained via sample average approximation converge reliably at $|S| \approx 1500$ to 2000 scenarios.

While this framework provides an effective approach for asset-deployment planning, several extensions would improve computational scalability and practicality of the model. The primary limitation is the explicit time indexing, which increases problem size and restricts the representation of continuous search operations. The current formulation also distinguishes between route-based and pattern-based assets, whereas in practice all assets are assigned to search regions for specified durations. A unified pattern-based representation would better match operational planning and remove the need for explicit time indexing, resulting in a more compact formulation. Incorporating more sophisticated models of target motion and detection that account for environmental conditions would improve the representation of uncertainty. These extensions would strengthen the model's applicability to real-world anti-submarine warfare planning and are the subjects of ongoing work.

References

Akbori, F. (2004). *Autonomous-agent based simulation of anti-submarine warfare operations with the goal of protecting a high value unit* (Unpublished doctoral dissertation). Monterey, California. Naval Postgraduate School.

An, W., Ayala, D. F. M., Sidoti, D., Mishra, M., Han, X., Pattipati, K. R., ... Hansen, J. A. (2012). Dynamic asset allocation approaches for counter-piracy operations. In *2012 15th international conference on information fusion* (pp. 1284–1291).

Bayraksan, G., & Morton, D. P. (2006). Assessing solution quality in stochastic programs. *Mathematical Programming*, *108*(2), 495–514. doi: 10.1007/s10107-006-0720-x

Bayraksan, G., & Morton, D. P. (2011). A sequential sampling procedure for stochastic programming. *Operations Research*, *59*(4), 898–913. doi: 10.1287/opre.1110.0926

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238–252. doi: 10.1007/BF01386316

Birge, J. R., & Louveaux, F. (2011). *Introduction to stochastic programming*. Springer New York. doi: 10.1007/978-1-4614-0237-4

Brown, G., Kline, J., Thomas, A., Washburn, A., & Wood, K. (2011). A game-theoretic model for defense of an oceanic bastion against submarines. *Military Operations Research*, *16*(4), 25–40.

Brown, S. S. (1980). Optimal search for a moving target in discrete time and space. *Operations Research*, *28*(6), 1275–1289. doi: 10.1287/opre.28.6.1275

Ding, H. (2018). *Models and algorithms for multi-agent search problems* (Unpublished doctoral dissertation). Boston University.

Dyer, M., & Stougie, L. (2006). Computational complexity of stochastic programming problems. *Mathematical Programming*, *106*(3), 423–432. doi: 10.1007/s10107-005-0597-0

Eagle, J. N., & Yee, J. R. (1990). An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Operations Research*, *38*(1), 110–114. doi: 10.1287/opre.38.1.110

Foraker, J., Royset, J. O., & Kaminer, I. (2016a). Search-trajectory optimization: Part i, formulation and theory. *Journal of Optimization Theory and Applications*, *169*(2), 530–549. doi: 10.1007/s10957-015-0768-y

Foraker, J., Royset, J. O., & Kaminer, I. (2016b). Search-trajectory optimization: Part ii, algorithms and computations. *Journal of Optimization Theory and Applications*, *169*(2), 550–567. doi: 10.1007/s10957-015-0770-4

Foraker, J. C., Royset, J., & Kaminer, I. (2011). *Optimal search for moving targets in continuous time and space using consistent approximations* (Unpublished doctoral dissertation). Naval Postgraduate School Monterey, CA.

Fortz, B., & Poss, M. (2009). An improved benders decomposition applied to a multi-layer network design problem. *Operations Research Letters*, *37*(5), 359–364. doi: 10.1016/j.orl.2009.05.007

Hoffman, A. J., & Kruskal, J. B. (2010). Integral boundary points of convex polyhedra. In *50 years of integer programming 1958-2008: From the early years to the state-of-the-art* (pp. 49–76). Springer. doi: 10.1007/978-3-540-68279-0_3

Kleywegt, A. J., Shapiro, A., & Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, *12*(2), 479–502. doi: 10.1137/s1052623499363220

Koopman, B. O. (1946). *Search and screening* (Tech. Rep. No. 56). Operations Evaluation Group, Office of the Chief of Naval Operations.

Kress, M., & Royset, J. O. (2008). Aerial search optimization model (asom) for uavs in special operations. *Military Operations Research*, *13*(1), 23–33.

Laan, C. M., Barros, A. I., Boucherie, R. J., Monsuur, H., & Noordkamp, W. (2020). Optimal deployment for anti-submarine operations with time-dependent strategies. *The Journal of Defense Modeling and Simulation*, *17*(4), 419–434. doi: 10.1177/1548512919855435

Lejeune, M., Royset, J. O., & Ma, W. (2024). Multi-agent search for a moving and camouflaging target. *Naval Research Logistics (NRL)*, *71*(4), 532–552. doi: 10.1002/nav.22160

Mak, W.-K., Morton, D. P., & Wood, R. K. (1999). Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*,

24(1-2), 47–56. doi: 10.1016/s0167-6377(98)00054-6

Naoum-Sawaya, J., & Elhedhli, S. (2013). An interior-point benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research*, *210*(1), 33–55. doi: 10.1007/s10479-010-0806-y

Royset, J. O., & Sato, H. (2010). Route optimization for multiple searchers. *Naval Research Logistics (NRL)*, *57*(8), 701–717. doi: 10.1002/nav.20432

Sato, H., & Royset, J. O. (2010). Path optimization for the resource-constrained searcher. *Naval Research Logistics (NRL)*, *57*(5), 422–440. doi: 10.1002/nav.20411

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.

Sidoti, D., Han, X., Zhang, L., Avvari, G. V., Ayala, D. F. M., Mishra, M., ... Pattipati, K. R. (2017). Context-aware dynamic asset allocation for maritime interdiction operations. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(3), 1055–1073. doi: 10.1109/tsmc.2017.2767568

Stone, L. D., Royset, J. O., & Washburn, A. R. (2016). *Optimal search for moving targets*. Springer Cham. doi: 10.1007/978-3-319-26899-6

Thomas, A. J. (2008). *Tri-level optimization for anti-submarine warfare mission planning* (Unpublished doctoral dissertation). Monterey, California. Naval Postgraduate School.

Urick, R. J. (1975). *Principles of underwater sound* (2nd ed.). McGraw-Hill.

Van Slyke, R. M., & Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, *17*(4), 638–663. doi: 10.1137/0117061

A Extensive form for the two-stage model

$$
\begin{aligned}
\max \quad & \sum_{j\in J}\sum_{t\in T}\sum_{s\in S} c_{jts} z_{jts} \\
\text{s.t.} \quad & x_{iat+1} \le \sum_{b\in N_{ia}} x_{ibt} && \forall i \in I, a \in A, t \in \{1,\dots,\tau-1\} \\
& \sum_{a\in A} x_{iat} = 1 && \forall i \in I, t \in T \\
& \sum_{p\in P}\sum_{t'=t}^{t+\mathfrak{e}_i} y_{ipt'} \le 1 && \forall i \in F, t \in \{1,\dots,\tau-\mathfrak{e}_i\} \\
& \sum_{i\in F}\sum_{p\in P}\sum_{t\in T} \kappa_p y_{ipt} \le \nu \\
& z_{jts} \le \sum_{(i,a)\in U_{jts}} x_{iat} + \sum_{(i,p,t')\in V_{jts}} y_{ipt'} && \forall j \in J, t \in T, s \in S \\
& \sum_{t\in T} z_{jts} \le 1 && \forall j \in J, s \in S \\
& x_{iat} \in \{0,1\} && \forall i \in I, a \in A, t \in T \\
& y_{ipt} \in \{0,1\} && \forall i \in F, p \in P, t \in T \\
& z_{jts} \in \{0,1\} && \forall j \in J, t \in T, s \in S.
\end{aligned}
$$

B Verification of Primal and Dual Feasibility for the Closed-Form Solution

We verify primal and dual feasibility for the closed-form solution. The proof partitions targets by the critical index k_j^* and considers three cases.

Primal feasibility. We verify constraints (7b) and (7c) for each $j \in J$.

Case $k_j^* = 0$: Here $z_{j1}^* = 1$ and $z_{jt}^* = 0$ for $t > 1$.

- For (7b):
  - For $t = 1$: $z_{j1}^* = 1 \le c_{j1}$.
  - For $t > 1$: $z_{jt}^* = 0 \le c_{jt}$.
- For (7c): $\sum_{t\in T} z_{jt}^* = 1$.

Case $0 < k_j^* < |T|$: Here $z_{jt}^* = c_{jt}$ for $t \le k_j^*$, $z_{j,k_j^*+1}^* = 1 - \sum_{t=1}^{k_j^*} c_{jt}$, and $z_{jt}^* = 0$ for $t > k_j^* + 1$.

- For (7b):
  - For $t \leq k_j^*$: $z_{jt}^* = c_{jt}$ holds with equality.
  - For $t = k_j^* + 1$: $z_{jt}^* = 1 - \sum_{t=1}^{k_j^*} c_{jt} < c_{j,k_j^*+1}$ (by definition of k_j^*).
  - For $t > k_j^* + 1$: $z_{jt}^* = 0 \leq c_{jt}$.
- For (7c): $\sum_{t \in T} z_{jt}^* = 1$ by construction.

Case $k_j^* = |T|$: Here $z_{jt}^* = c_{jt}$ for all t.

- For (7b): $z_{jt}^* = c_{jt}$ holds with equality for all t.
- For (7c): $\sum_{t \in T} z_{jt}^* = \sum_{t=1}^{|T|} c_{jt} \leq 1$ by definition of k_j^*.

Dual feasibility. We verify constraint (8b) for each $j \in J$.

Case $k_j^* = 0$: Here $\mu_j^* = r_{j1}$ and $\lambda_{jt}^* = 0$ for all t.

- For (8b): $\lambda_{jt}^* + \mu_j^* = r_{j1} \geq r_{jt}$ for all t (by non-increasing rewards).

Case $0 < k_j^* < |T|$: Here $\mu_j^* = r_{j,k_j^*+1}$ and $\lambda_{jt}^* = r_{jt} - r_{j,k_j^*+1}$ for $t \leq k_j^*$, else $\lambda_{jt}^* = 0$.

- For (8b):
  - For $t \leq k_j^*$: $\lambda_{jt}^* + \mu_j^* = (r_{jt} - r_{j,k_j^*+1}) + r_{j,k_j^*+1} = r_{jt}$.
  - For $t > k_j^*$: $\lambda_{jt}^* + \mu_j^* = r_{j,k_j^*+1} \geq r_{jt}$ (by non-increasing rewards).
- For (8c): $\lambda_{jt}^* = r_{jt} - r_{j,k_j^*+1} \geq 0$ for $t \leq k_j^*$ (by non-increasing rewards), and $\lambda_{jt}^* = 0$ otherwise.

Case $k_j^* = |T|$: Here $\mu_j^* = 0$ and $\lambda_{jt}^* = r_{jt}$ for all t.

- For (8b): $\lambda_{jt}^* + \mu_j^* = r_{jt}$ for all t.

C Average work units for the decomposition method and extensive form

Table 8: Average work units for the decomposition method and extensive form (Figs. 3 and 4)

(a) *Instance 1*

Num Scen.	Decomp.	Ext. Form
10	5.751	2.768
20	4.218	2.404
75	5.821	3.263
100	6.147	3.663
250	9.681	7.426
500	20.499	18.006
750	31.115	32.095
1000	39.359	53.029
1500	88.931	142.093
2000	159.366	190.028
2500	192.468	392.721
3000	223.136	468.015
3500	279.862	692.437
4000	362.225	806.496

(b) *Instance 2*

Num Scen.	Decomp.	Ext. Form
10	6.416	0.964
20	8.100	11.994
75	14.122	9.777
100	15.080	12.098
250	31.218	33.943
500	73.409	164.569
750	161.324	595.247
1000	370.507	1179.484
1500	1136.157	5290.591
2000	1999.829	9847.589
2500	3510.519	14153.056
3000	5394.449	17526.773
3500	7249.277	19694.111
4000	8853.844	19986.925

D Biased and unbiased estimates with gap percentage

Table 9: Biased and unbiased estimates with gap percentage (Figs. 5 and 6)

(a) *Instance 1*

Num Scen.	Biased	Unbiased	Gap (%)
10	17.523	5.823	66.8%
20	14.232	6.314	55.6%
75	10.469	6.970	33.4%
100	10.004	6.977	30.3%
250	8.957	6.970	22.2%
500	8.318	7.030	15.5%
750	8.163	7.231	11.4%
1000	8.005	7.237	9.6%
1500	7.856	7.360	6.3%
2000	7.769	7.385	4.9%
2500	7.658	7.389	3.5%
3000	7.668	7.367	3.9%
3500	7.633	7.542	1.2%
4000	7.610	7.460	2.0%

(b) *Instance 2*

Num Scen.	Biased	Unbiased	Gap (%)
10	18.893	6.658	64.8%
20	15.578	6.468	58.5%
75	11.820	7.298	38.3%
100	10.911	7.656	29.8%
250	9.690	7.706	20.5%
500	9.163	7.791	15.0%
750	8.900	7.832	12.0%
1000	8.813	7.771	11.8%
1500	8.595	7.979	7.2%
2000	8.545	7.977	6.6%
2500	8.447	7.965	5.7%
3000	8.417	7.954	5.5%
3500	8.366	8.224	1.7%
4000	8.379	8.177	2.4%