Toward blackbox optimization for planning maritime search and rescue operations

Amirhossein Esmaeilpour*
Department of Operations and Decision Systems
Université Laval, Qc, Canada
amirhossein.esmaeilpour@ulaval.ca

Michael Morin
Department of Operations and Decision Systems
Université Laval, Qc, Canada

Irène Abi-Zeid
Department of Operations and Decision Systems
Université Laval, Qc, Canada

ABSTRACT

In search and rescue operations, optimization-based decision support systems can assist search mission coordinators in planning searches with higher probabilities of success, potentially resulting in more lives saved. However, traditional model-and-solve techniques, such as integer programming, are not easily applicable in maritime searches where there is a need to conduct simulations to compute the value of the objective function. In this paper, we show how we can still use mathematical programming to propose maritime search and rescue plans, even when simulations are used. In addition, we take into account operational constraints such as airspace deconfliction for security reasons. Our model, implemented for a suitable solver using a surrogate to estimate search simulation results on scenarios with two helicopters, proved flexible and fast—we implemented operational constraints in a problem-specific model, solved by a general solver, which provided quality solutions in a short time frame.

Keywords

Maritime search operations planning, Blackbox optimization, Modeling for real-life applications, Search and rescue

INTRODUCTION

The act of searching is an important part of many humanitarian operations, such as search and rescue (SAR), mine countermeasures, and of many surveillance operations for the purpose of protecting individuals, the environment, resources, or infrastructure. But how and where to search? The answer lies in good and efficient search planning that ensures the best use of scarce and constrained resources such as aircraft or vessels. Search theory, a sub-discipline of operations research, can be applied to help allocate the available resources to search areas in such a way to maximize the chances of finding one or more search objects under operational constraints (Abi-Zeid & Frost, 2005; Frost, 1999; Stone, 1975).

The a priori knowledge on an object's location is usually encoded in a probability distribution, called the probability of containment for stationary objects (POC). The probability of a search unit[1] detecting an object located in its search area is the conditional probability of detection (POD). It is a function of the search effort available, the type of sensor used for searching, the parameters of the search unit, the type of search object and the environmental

*corresponding author

[1]We use "search unit" to designate the "searcher" which is, in the context of maritime search and rescue operations, often an aircraft or a vessel with crew on-board. We employ the term "search unit" instead of "search and rescue unit" to simplify the text, although some search units have rescue capabilities.

conditions (Frost, 1999; Stone et al., 2016). A common figure of merit for evaluating the quality of search plans, i.e., the assignment of a search area to a search unit, is the probability of success (POS), the product of the POC and the POD, to be maximized. When multiple search units are used, the objective function is the total probability of success of the search unit assignments to their respective search areas.

The available search effort for a search unit can be defined by the time spent on-scene or by the length of the search pattern. The probabilities of containment, detection, and success are concepts needed in models to provide optimal allocations of search units to rectangular search areas, e.g., (Abi-Zeid & Frost, 2005; Morin et al., 2017; Richardson & Discenza, 1980). Operational constraints, such as minimal track spacing for parallel patterns and deconfliction of the search units' patterns to ensure non-overlapping search areas during the same time periods, restrict the space of feasible allocations. An optimal search plan for a set of search units is a set of search rectangles, each uniquely assigned to one search unit such that the rectangles are disjoint in space.

For moving search objects, some form of probabilistic motion model is usually assumed (Stone et al., 2016). Recent optimization-based decision support systems for maritime SAR use Monte Carlo based drift simulations to estimate the likely trajectories of the search objects based on current and weather data (Abi-Zeid et al., 2019; Breivik & Allen, 2008; Kratzke et al., 2010). A trajectory consists of predicted particles positions at specific time intervals, e.g., every five minutes. When n particles are seeded, each particle has initially the probability $1/n$ of being the search object. In order to evaluate a candidate search plan, search units' patterns are simulated and the probability of detecting a particle by a search unit is computed at the closest point of approach between a particle and a search leg. SAROPS (Kratzke et al., 2010) and SAR Optimizer (Abi-Zeid et al., 2019) are two examples of systems that employ search simulations to evaluate the probability of success of searching for drifting objects. The main downside of this simulation-based approach is that mathematical programming, more specifically modeling and solving the problem directly, is not easily done. This is due to the fact that the objective function, namely, the probability of success, is computed using a simulation, which requires that the optimization uses problem-specific algorithms and heuristics (Abi-Zeid et al., 2019; Kratzke et al., 2010; Laperrière-Robillard et al., 2022). However, a model-and-solve approach, where the model is problem-specific and the solver is general, would enable using, or at least testing, existing solvers to improve the solving time and the search plan's quality. Furthermore, it can enable developers and practitioners to integrate or test operational constraints faster, without modifying the algorithms of the solvers, and without creating algorithms from scratch.

In this paper, we present preliminary results using a model-and-solve approach for SAR Optimizer, the simulation and optimization module of the Advanced search planning tool (ASPT) of the Canadian Coast Guard, also known as CANSARP. The contributions are the following:

1. we show how a solver enabling blackbox optimization—in our case the Hexaly solver which uses a surrogate model to estimate the probability of success instead of only using simulations—can be used to implement a model-and-solve approach;
2. in response to an operational need, we formulate a complete model integrating deconfliction constraints that are not yet implemented in SAR Optimizer's heuristics as presented in Abi-Zeid et al., 2019; and
3. we evaluate our approach on various scenarios.

The paper is structured as follows. We first present related work on decision support systems for maritime search operations planning. Second, we describe the problem formulation in greater details and provide the solution approach, including the proposed model. Third, we present the numerical experiments and discuss our results.

RELATED WORK

The United States Coast Guard Computer-Assisted Search Planning system (CASP) was the first maritime SAR decision support system that had the ability to recommend optimal search plans. To our knowledge, the problem of allocating multiple search units to rectangular areas in order to maximize the probability of success was first formulated as a mathematical program by Discenza, 1978. The problem, often called the multiple rectangular search areas problem (or MRSA), is usually formulated for a stationary object, e.g., in the SARPlan decision support system (Abi-Zeid & Frost, 2005). Developed in the early 1970s, CASP optimization module employed heuristics based on search theory, and assumed a stationary object (Richardson & Discenza, 1980). CASP was later replaced by the Search and Rescue Optimal Planning System (SAROPS) (Kratzke et al., 2010). SAROPS uses drift and search simulation and an optimization algorithm based on a heuristic that first assigns rectangular areas to the search units using a probability of success estimate. It then refines this initial solution (Kratzke et al., 2010). Both

SAROPS and CASP provide deconflicted search patterns without overlap as does SARPlan for stationary search objects (Abi-Zeid & Frost, 2005).

SAR Optimizer, the search optimization and simulation module used by the Canadian Coast Guard (Abi-Zeid et al., 2019), can be used either as an optimizer, to define the time-constrained highest probability of success search plan, namely the assignment of rectangular search areas to available search units, or as an evaluator to calculate the probability of success and other figures of merit of manual search plans. SAR Optimizer algorithms integrate operational constraints such as the minimal track spacing of the search patterns. A search area for a search unit is defined by a rectangle's center (longitude and latitude), length, width, and orientation (θ) such as for example, the mean direction of the drift.

Although search simulations in SAR Optimizer tend to be relatively fast (less than a second on some hardware), many are needed to find a good solution to the problem. That is, an effort allocation with a sufficiently high probability of success is found by trying many allocations. Nonetheless, time is limited in situations where lives are at risk, and a search plan must be provided in under two minutes. To avoid having to conduct multiple simulations multiple times, Laperrière-Robillard et al., 2022 built supervised machine learning models on a small corpus of rectangle assignments for one search unit. These models provided estimates of the probability of success. The best model was then used to order the candidate search plans, thereby increasing the chances of encountering promising search plans faster, which can lead to a better solution in case of early stopping (Laperrière-Robillard et al., 2022). In this paper, we formulate a model-and-solve approach to optimization in maritime search operation planning, by extending the planning problem described in Abi-Zeid et al., 2019 to the case where the tasked search units must have non-overlapping search patterns. This constraint is necessary for safety reasons, to avoid risks of collision when two or more aircraft are on-scene at the same time, at the same altitude.

MODEL-AND-SOLVE FOR MARITIME SEARCH PLANNING

In practice, a search environment can have any orientation or size. Nonetheless, it is often designed to enclose the drifting trajectories of the search object during the time the search units are searching, called the on-scene time. For simplicity purposes, we assume a rectangular area with a north-south grid orientation without rotation and the presence of two search units.

In this problem, a valid solution (search plan) consists of two non-overlapping rectangles R_1 and R_2 that lie within the search environment. A search rectangle contains a parallel search pattern, which total length corresponds to the amount of effort, in distance units, available to the search unit. For security and operational reasons, track spacings are constrained by a lower and an upper bound. This in turn constrains the possible size of a search rectangle as a function of the effort available to the search unit. An optimal solution (search plan) maximizes the total probability of success, calculated via a simulation of the search plan.

Mathematical Model

We propose to use a model-and-solve approach with a solver that enables blackbox optimization. Blackbox optimization involves problems where the objective function and/or constraints can only be evaluated through external processes—whether through computer simulations, existing software systems, or physical experiments (Audet & Hare, 2017; Bajaj et al., 2021). In these cases, direct access to the mathematical form of these functions is not possible. Only output corresponding to a given input is accessible. In the search planning problem based on Monte Carlo drift simulation, the constraints are known, but the objective function, corresponding to the probability of success, is calculated following simulations by a search simulator such as the one found in SAR Optimizer, that can be seen as a blackbox. Our approach is akin to simulation-optimization, blackbox optimization being the general case where the function to evaluate is not necessarily a simulator.

We use a formalism developed for the solver Hexaly, previously called LocalSolver (Hexaly, 2021), a global optimization solver that combines heuristic and exact algorithms, to solve various types of problems such as combinatorial, mixed-integer, or continuous. In addition to linear relationships between continuous or integer variables found in linear solvers, its modeling language supports logical constraints with logical operators. Hexaly also supports blackbox optimization, and uses Gutmann's radial basis function substitution method surrogate (Gutmann, 2001). Basically, the solver proceeds as follows: Once the objective function is evaluated by an external program, a radial basis function (surrogate) is used to approximate it—the algorithms alternate between exploitation of good feasible solutions and exploration (Tenaud, 2022). Along with the surrogate, the solver enables the use of constraints, enabling a model-and-solve approach.

In the following, we first define the parameters of the model, its variables, objective function, and constraints. We use the following convention: Parameter names are in lower case with underscores to separate words (also called

snake case). Variable names are in uppercase. Variable names containing *UL*, *LR* refer to the upper left and lower right corners of a rectangle respectively. *X* is used to denote a variable related to a west-east coordinate and *Y* to a north-south coordinate. The same convention is used for parameters, but with lower case letters.

Parameters The model parameters of a search unit $i \in \{1, 2\}$ consist of its type, the available search effort, the minimum and maximum track spacing, the on-scene time, and the search speed:

- $total\_effort_i$ is the total search pattern length (effort) allocated to rectangle i, calculated based on the search unit speed and total on-scene time;
- $track\_spacing\_min_i$ and $track\_spacing\_max_i$ are lower and upper bounds for the track spacing in nautical miles.

Since track spacing is a function of the search area, the bounds on the track spacing constrain the area of the search rectangles.

- $area\_min_i$ is the minimum allowed search area in squared nautical miles, calculated by Equation (1) where $total\_effort_i$ represents the total search distance and $track\_spacing\_min_i$ is the minimum track spacing.

$$area\_min_i = track\_spacing\_min_i \times total\_effort_i \,. \tag{1}$$

- $area\_max_i$ is the maximum allowed search area given by Equation (2) using $track\_spacing\_max_i$.

$$area\_max_i = track\_spacing\_max_i \times total\_effort_i \,. \tag{2}$$

The search environment (maximum rectangle) is defined by its four corners: (max_ul_x, max_ul_y), (max_ur_x, max_ur_y), (max_lr_x, max_lr_y), and (max_ll_x, max_ll_y).

The decision variables x and y, defining respectively the longitudinal and latitudinal coordinates of a rectangle, in the Mercator projection, are bounded by max_x, max_y, min_x, and min_y, determined by taking the maximum and minimum values across all corner coordinates of the maximum rectangle:

$$max\_x = \max(max\_ul\_x, max\_ur\_x, max\_lr\_x, max\_ll\_x) \,, \tag{3}$$
$$max\_y = \max(max\_ul\_y, max\_ur\_y, max\_lr\_y, max\_ll\_y) \,, \tag{4}$$
$$min\_x = \min(max\_ul\_x, max\_ur\_x, max\_lr\_x, max\_ll\_x) \,, \tag{5}$$
$$min\_y = \min(max\_ul\_y, max\_ur\_y, max\_lr\_y, max\_ll\_y) \,. \tag{6}$$

Finally, the following tolerance parameters are defined:

- b is used to relax the constraints on the inclusion of the rectangles in the search environment and on their area to allow for a larger set of feasible solutions (if needed);
- ob is a minimum separation required between non-overlapping rectangles, it is used as an overlap buffer parameter for the rectangles to adjust the tolerance.

Decision Variables For a search unit $i \in \{1, 2\}$, two pairs of variables (Equations (7) and (8)) describe the corners of the rectangular area where its search pattern will be positioned:

$$(ULX_i \in [min\_x - b, max\_x + b], ULY_i \in [min\_y - b, max\_y + b]) \,, \tag{7}$$
$$(LRX_i \in [min\_x - b, max\_x + b], LRY_i \in [min\_y - b, max\_y + b]) \,. \tag{8}$$

The variable TS_i is the track spacing of search unit $i \in \{1, 2\}$ (with domain (9)). Variable N_i (with domain (10)) is the number of parallel search legs in a search pattern.

$$TS_i \in [track\_spacing\_min_i, track\_spacing\_max_i] \,, \tag{9}$$
$$N_i \in \{1, ..., 10000\} \,. \tag{10}$$

Intermediate Variables In addition, the following intermediate variables are computed from the decision variables:

- $WIDTH_i$, the total distance traveled ($total\_effort_i$) divided by the number of search legs (N_i);
- $HEIGHT_i$, the product of the number of search legs (N_i) and track spacing (TS_i);
- *HSEP* and *VSEP*, boolean variables indicating if the two rectangles have a horizontal or vertical separation (or both). *HSEP* = True encodes that rectangle 2 is at the left of rectangle 1 or that rectangle 2 is at the right of rectangle 1. *VSEP* = True encodes that rectangle 2 is above rectangle 1 or that rectangle 2 is below rectangle 1.

Objective Function The objective function (Equation (11)) is the total probability of success of all search units. It is obtained from an external blackbox function that takes as input the decision variables corresponding to each search unit's search pattern characteristics:

$$\text{Maximize } POS = \mathsf{external}(ULX_1, ULY_1, LRX_1, LRY_1, TS_1, N_1, ULX_2, ULY_2, LRX_2, LRY_2, TS_2, N_2)\,. \tag{11}$$

Here, external is an external function, declared as such in the model implementation for the Hexaly solver. It first generates the search pattern corresponding to each rectangle, creates the input files required by the evaluator of SAR Optimizer for search simulation, launches the search simulation, and captures the calculated total probability of success. In our implementation, search patterns are currently assumed to have an east-west orientation.

Constraints The first set of constraints (Constraints (12)–(13)) ensures that each rectangle maintains a valid rectangular shape with aligned with the orientation of the search environment. That is, the upper-left corner is positioned above and to the left of the other corners. For $i \in \{1, 2\}$:

$$ULX_i < LRX_i\,, \tag{12}$$
$$LRY_i < ULY_i\,. \tag{13}$$

The rectangles need to meet the area requirement as defined by the parameters $area\_min_i$ and $area\_max_i$. This is captured by Constraints (14)–(16).

$$WIDTH_i = LRX_i - ULX_i\,, \tag{14}$$
$$HEIGHT_i = ULY_i - LRY_i\,, \tag{15}$$
$$area\_min_i - t \leq WIDTH_i \times HEIGHT_i \leq area\_max_i\,. \tag{16}$$

The deconfliction constraints (Constraints (17)–(19) where ob is an overlap buffer parameter) ensure that two rectangles are separated either horizontally or vertically or both horizontally and vertically:

$$\text{True} = HSEP \vee VSEP \tag{17}$$
$$HSEP = LRX_2 + ob \leq ULX_1 \vee LRX_1 + ob \leq ULX_2\,, \tag{18}$$
$$VSEP = ULY_1 + ob \leq LRY_2 \vee ULY_2 + ob \leq LRY_1\,. \tag{19}$$

This formulation ensures that rectangles are separated by requiring either horizontal separation (left/right) or vertical separation (above/below) or both.

Finally, we define the number of legs as a function of the total effort of search unit $i \in \{1, 2\}$ and of the width of its rectangle, and as a function of the rectangle height and of the track spacing (Constraints (20) and (21)).

$$N_i = \frac{total\_effort_i}{WIDTH_i}\,, \tag{20}$$
$$N_i = \frac{HEIGHT_i}{TS_i}\,. \tag{21}$$

These ensure that the rectangle's height is covered by N_i parallel lines with a track spacing of TS_i, and that the total length of the search pattern does not exceed the total available effort for a search unit ($total\_effort_i$).

Table 1. Instance parameters

Instance	Simulation Start Time	Search Start Time	Total Endurance (hours)	Search object Class
A2	2023-02-24 05:00:00	2023-02-24 18:30:00	2	LIFE-RAFT-DB-11
B2	2023-02-24 06:00:00	2023-02-24 18:30:00	2	SKIFF-4
C2	2023-03-22 10:00:00	2023-03-22 18:30:00	2	SPORT-BOAT
D2	2023-04-12 10:00:00	2023-04-12 18:30:00	2	FISHING-VESSEL-6
E2	2023-03-28 11:00:00	2023-03-28 18:30:00	2	PERSON-POWERED-VESSEL-2
A3	2023-02-24 05:00:00	2023-02-24 17:30:00	3	LIFE-RAFT-DB-11
B3	2023-02-24 06:00:00	2023-02-24 17:30:00	3	SKIFF-4
C3	2023-03-22 10:00:00	2023-03-22 17:30:00	3	SPORT-BOAT
D3	2023-04-12 10:00:00	2023-04-12 17:30:00	3	FISHING-VESSEL-6
E3	2023-03-28 11:00:00	2023-03-28 17:30:00	3	PERSON-POWERED-VESSEL-2

Model Relaxation

In our model, the track spacing variables, N_i, are defined on an integer domain (see Equation (10)). Having N_i as integers is restrictive from a planning perspective since it constrains the feasible rectangles to have an area fitting a fixed amount of effort $total\_effort_i$ using an integer number of legs. For this reason we tested a model with a continuous relaxation of N_i (Equation (22)).

$$N_i \in [1, 10000]\,. \tag{22}$$

The impact of relaxing the model is that the solver might recommend adding a fractional (incomplete) leg at the end of the search pattern. When $N_i = 10.5$, for instance, it recommends 10 and a half legs. This can be corrected in the external function used to provide the search pattern and call the simulator (described by Equation (11)) by rounding down the N_i value to the nearest integer. As we discuss in our experiments, although this relaxation leads to search patterns with a small decrease in the allocated effort compared to the $total\_effort_i$ values, it can improve the performance of the solver and lead to higher quality solutions in some cases.

EXPERIMENTS

In order to test our model, we used five drifts to generate ten problem instances. Each drift file contains 5000 possible trajectories of a search object with way points every 5 minutes. The drift simulation length, in hours, varies. Table 1 shows the parameters of the drift of all the instances. Each problem instance involves two search units, two helicopters with the same following parameters: Speed of 90 knots (nautical miles per hour), a minimum track spacing of 0.15 nautical miles, maximum track spacing of 200 nautical miles, and altitude of 500 feet. For each drift file, we use two configurations—one with 2 hours for endurance time and the other with 3—leading to two instances per drift file. Different search objects are used in the instances.

The buffer parameter b is set to 10% of the search environment's height to constrain the domain of the decision variables. Finally, we set the overlap buffer ob about 0.0054 nautical miles (10 meters) to ensure a minimum separation between rectangles for tolerance purposes.

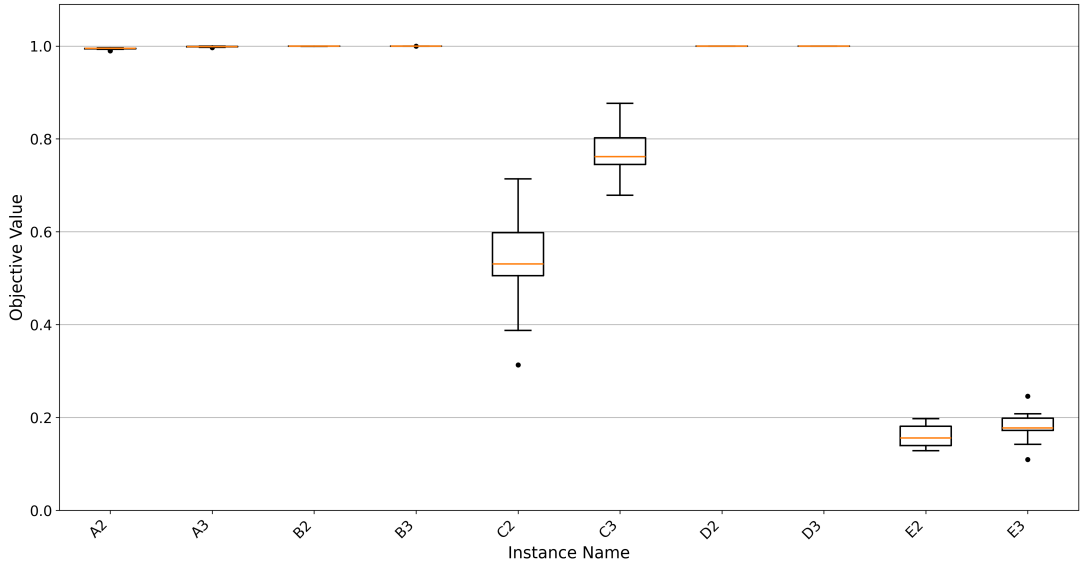
Experiments were run on a GNU/Linux operating system with an AMD Ryzen 9 5900X 12-core processor, and 64 GB of system memory. The code developed for the experiments is in Python. We used the modeling API (application programming interface) of Hexaly solver. Hexaly called SAR Optimizer evaluator as needed when calculating the objective value. A single thread was used for each run.

Hexaly external function can be used with and without activating the surrogate model. Preliminary experiments confirmed that activating the surrogate model is necessary to get good results. All runs were therefore performed with the surrogate. On each instance, we ran our model with the relaxed N_i variables and the one with the integer N_i variables. Since the solver is non-deterministic when using the surrogate, we performed 30 independent runs. Because time is an important resource in any SAR operation, we set the time limit to 1 minute.

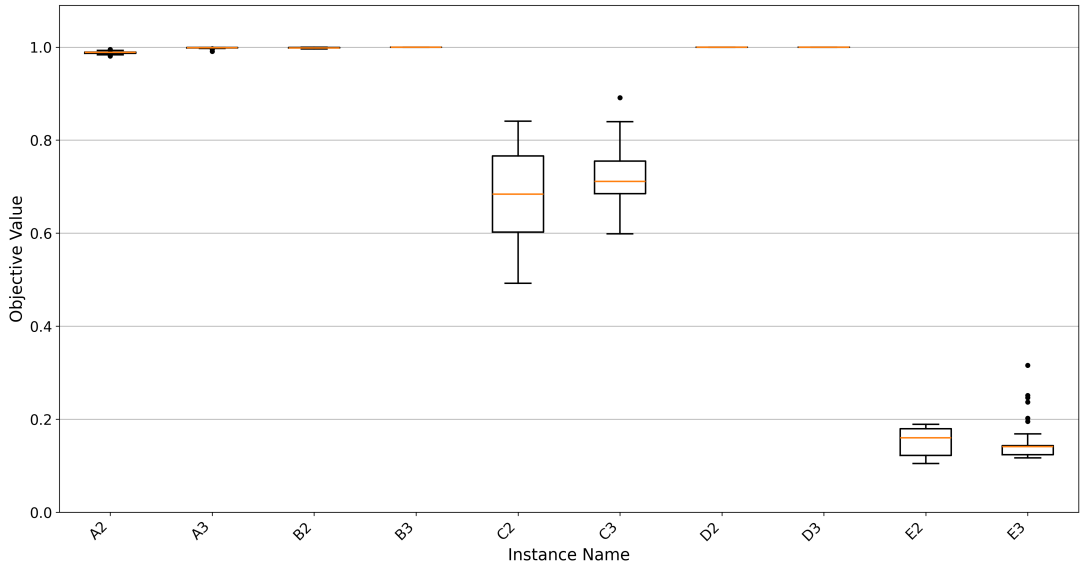
For each instance, we performed a statistical analysis and calculated the mean probability of success across runs involving the surrogate. We also report the probability of success of the best solution found and the coefficient of variation of the probability of success.

RESULTS AND DISCUSSION

Figure 1 shows the box plots of the surrogate-assisted optimization results across all ten instances for the two versions of our models—the one with integer N_i variables and the one obtained by relaxing the N_i variables



(a) Original model (integer N_i)



(b) Relaxed model (continuous N_i)

Figure 1. The box plot of the surrogate-assisted optimization results of 30 runs for all ten instances for both models

to continuous variables. It is important to note that these represent the best solutions obtained in the allocated optimization time, namely one minute, and that we have no guarantee of optimality. Each box shows the range between the 25th percentile (Q1) and the 75th percentile (Q3) of the data. The whiskers extend from the edges of the box to the smallest and largest values within 1.5 interquartile range. Also, we consider data points that fall outside the whiskers as outliers. Each box corresponds to a specific instance. As seen on Figures 1a and 1b, the two boxplots almost overlap in most cases and show minimal differences between the integer and relaxed models. For six instances (A2, B2, D2, A3, B3 and D3), both the integer and relaxed models achieve very high objective value in many of the 30 runs. The boxplots for these instances are at the top of Figures 1a and 1b, showing consistently high performances. However, for the more challenging instances (C2, C3, E2 and E3), the boxes and whiskers are larger and have lower median values, which indicates high variability and, possibly, instances where detection is harder for both integer and relaxed models. Harder detection might be due to the zig-zag shape of these instances (C2, C3, E2, and E3). For the relaxed model, outliers are visible in Figure 1b in a few instances including A2, A3, C3 and E3. For the original model with integer N_i, Figure 1a shows outliers for instances A2, A3, B3, C2 and E3. Outliers are runs where the probability of success value deviated from the majority of the other runs on the same instance.

Table 2 shows the results with the surrogate for all instances, for the relaxed and integer models. The table includes the best probability of success value across runs (best POS), mean objective value across runs (mean POS) and

Table 2. Optimization results for 30 runs and coefficient of variation (CV) for both models

	Relaxed model (continuous N_i)			Original model (integer N_i)		
Instance	Best POS	Mean POS	CV (%)	Best POS	Mean POS	CV (%)
A2	0.99496	0.98843	0.33981	0.99590	0.99430	0.15729
B2	0.99975	0.99852	0.08921	0.99989	0.99968	0.01250
C2	0.84077	0.68772	12.87485	0.71415	0.54361	17.57799
D2	0.99990	0.99990	0.00000	0.99990	0.99990	0.00000
E2	0.18876	0.15044	20.22864	0.19735	0.15979	14.41007
A3	0.99952	0.99836	0.15970	0.99957	0.99871	0.07738
B3	0.99990	0.99990	0.00000	0.99990	0.99989	0.00007
C3	0.89121	0.72935	8.21421	0.87676	0.77509	5.14090
D3	0.99990	0.99990	0.00000	0.99990	0.99990	0.00000
E3	0.31574	0.15606	30.25077	0.24568	0.18123	13.51374

coefficient of variation of the objective value (CV). In three instances out of ten (C2, C3, and E3), the relaxed model enabled the solver to reach a higher best objective value, whereas for C3 and E3 the average objective value across runs was lower compared to the integer model. On four instances (A2, B2, E2, and A3), the model with an integer number of search legs reached a higher objective value. On the remaining instances (D2, B3, and D3), the performance was similar. On some instances, the probability of success is lower than on others. This does not mean that our approach does not perform well on those instances. A low probability of success is often the result of instances where the detectability of a search object is lower. For instance, in the integer model, on instances E2 and E3, our approach achieved a mean POS of 15.98% and 18.12% respectively, whereas on instances B2 and B3 it achieved a mean probability of success of 99.97% and 99.99% respectively. For this reason, we focus our discussion on the variability of the results across runs on the same instances.

For the integer model, the results on instances E2, E3, C2 and C3 showed the highest variability with a coefficient of variation of 14.41%, 13.51%, 17.58% and 5.14%, respectively, indicating significant differences in the probability of success across 30 runs. In contrast, the continuous model yielded coefficients of variation of 20.23%, 30.25%, 12.87% and 8.21% for instances E2, E3, C2 and C3 respectively. Runs on A3, B3, D2 and D3 were highly consistent for both models, showing coefficient of variation of around 0% in both models, indicating that these results are highly reliable and repeatable across 30 runs. Also, A2 had a coefficient of variation of 0.16% in the integer model and 0.34% in the relaxed model; instance B2 had a coefficient of variation of 0.01% in the integer model and 0.09% in the relaxed model.

Figures 2 and 3 show examples of search rectangles obtained by Hexaly for all 3-hour on-scene endurance instances along with the drifting particles for the on-scene time. In each row of figures, the left one belongs to the output of the relaxed model and the right one belongs to the integer model. As we can see in Figure 2, the low mean POS of instance E3 can be related to its drift shape. The two rectangles couldn't fully cover the on-scene drift in both models. This likely contributed to the lower probability of success of these instances. In contrast, on instances like D3, A3 and B3, the solver might have achieved a high probability of success values with both models, because of shape of the drift. The rectangles show a much better alignment with the drift trajectory. Also, instance C3 appears to be particularly difficult due to its Z-shaped drift (a mean probability of success of 72.94% in relaxed model and 77.51% in the integer model). This zigzag drift presents a particularly challenging scenario for our current rectangle fitting method because the sharp turns in the trajectories of the object make it difficult to achieve high coverage with rectangular shapes since we limit the possible search pattern orientation to east-west. The visualization of the search patterns shows that our approach produces sensible search patterns in A3, B3 and D3 in the both models. However, it also shows some of the current limitations of our models. For instance, in the case of E3 and C3 a potentially better pattern could be obtained by rotating the search rectangles, a feature that we have not yet implemented.

Similarly, it might be beneficial to determine the number of search units and amount of effort necessary as decision variables. This could be done by extending the models. This is not a limitation of the model-and-solve approach itself. Our future work will include these model extensions.

In summary, our approach not only found feasible rectangles, but was also able to consistently find quality solutions in the short allowed time. This confirms the potential of a model-and-solve approach to maritime SAR optimization planning, even when search simulation needs to be used to evaluate the objective function. In addition, with this model-and-solve approach we successfully took into account operational constraints for airspace safety.

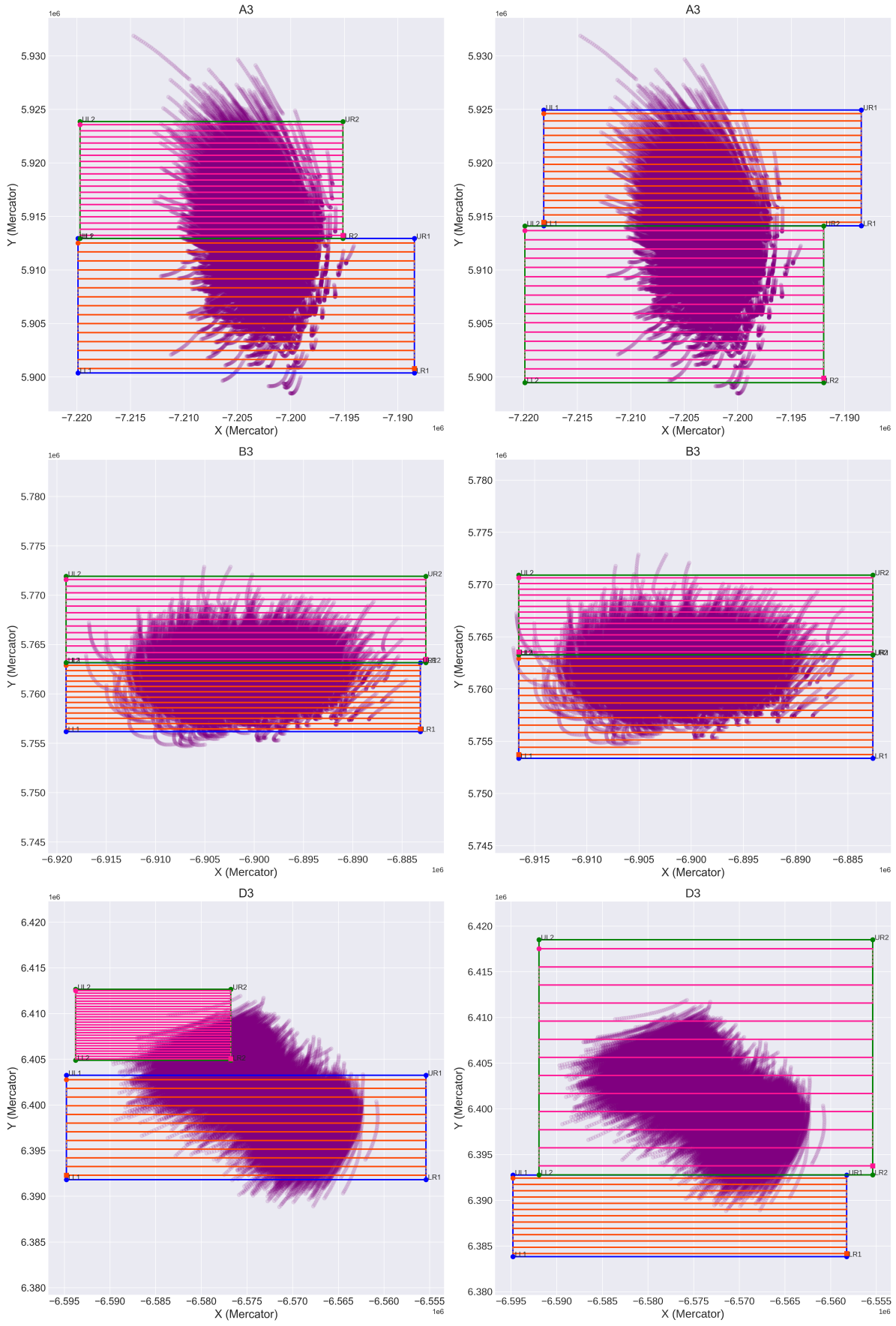


Figure 2. Generated search rectangles and search patterns (the pink and orange lines) with the surrogate across 3 different instances with the relaxed model (continuous N_i) on the left, and the integer N_i on the right

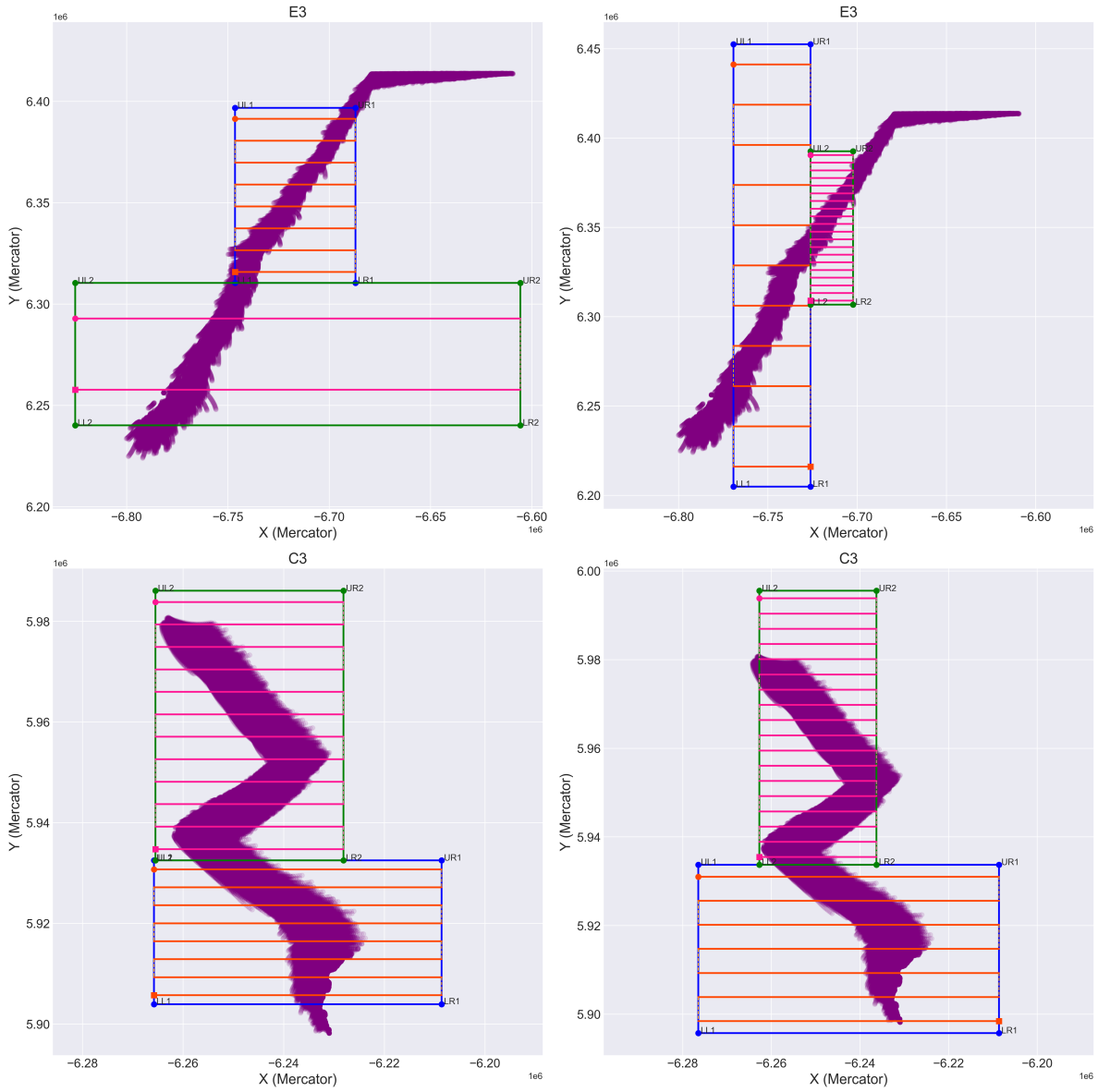


Figure 3. Generated search rectangles and search patterns (the pink and orange lines) with the surrogate across 2 different instances with the relaxed model (continuous N_i) on the left, and the integer N_i on the right

CONCLUSION

In this paper, we addressed the problem of deconfliction of multiple search units in maritime search operations for moving objects. The first contribution of the proposed method was to ensure that search rectangles generated for different search units are non-overlapping, thereby enhancing operational safety.

The second contribution relates to using a blackbox solver with a surrogate to identify feasible and efficient search rectangles across 10 test instances within the 1-minute time limit. This highlights the significant advantage of using surrogate models. This preliminary work was meant as a proof of concept to test black-box optimization for search planning. We were able to illustrate how a model-and-solve approach to maritime SAR operations planning can be applied.

Future work will evaluate the scalability of this approach to accommodate more search units and larger search areas, extending the models to include number of search units and effort as decision variables, and more importantly, generalizing the model to allow search pattern rotations for more realistic search plans.

ACKNOWLEDGMENTS

This project was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) [grants RGPIN-2021-03495 and DGECR-2021-00189].

REFERENCES

Abi-Zeid, I., Morin, M., & Nilo, O. (2019). Decision support for planning maritime search and rescue operations in canada. In J. Filipe, M. Smialek, A. Brodsky, & S. Hammoudi (Eds.), *Proceedings of international conference on enterprise information systems* (pp. 316–327, Vol. 1). SciTePress.

Abi-Zeid, I., & Frost, J. R. (2005). Sarplan: A decision support system for canadian search and rescue operations. *European Journal of Operational Research*, *162*(3), 630–653.

Audet, C., & Hare, W. (2017). *Derivative-free and blackbox optimization*. Springer Cham. https://doi.org/10.1007/978-3-319-68913-5

Bajaj, I., Arora, A., & Hasan, M. F. (2021). Black-box optimization: Methods and applications. In *Black box optimization, machine learning, and no-free lunch theorems* (pp. 35–65). Springer.

Breivik, O., & Allen, A. A. (2008). An operational search and rescue model for the Norwegian Sea and the North Sea. *Journal of Marine Systems*, *69*(1-2), 99–113.

Discenza, J. H. (1978). *Optimal Search with Multiple Rectangle Search Areas* [Ph. D.]. New York University.

Frost, J. R. (1999). Principles of search theory. *Response*, *17*(2), 1–23.

Gutmann, H.-M. (2001). A radial basis function method for global optimization. *Journal of global optimization*, *19*(3), 201–227.

Hexaly. (2021). Constrained black-box optimization.

Kratzke, T. M., Stone, L. D., & Frost, J. R. (2010). Search and rescue optimal planning system. *2010 13th International Conference on Information Fusion*, 1–8.

Laperrière-Robillard, T., Morin, M., & Abi-Zeid, I. (2022). Supervised learning for maritime search operations: An artificial intelligence approach to search efficiency evaluation. *Expert Systems with Applications*, *206*, 117857.

Morin, M., Abi-Zeid, I., Quimper, C.-G., & Nilo, O. (2017). Decision support for search and rescue response planning. *Proceedings of the 10th International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, 973–984.

Richardson, H. R., & Discenza, J. H. (1980). The United States Coast Guard Computer-Assisted Search Planning system (CASP). *Naval Research Logistics Quarterly*, *27*(4), 659–680.

Stone, L. D. (1975). *Theory of optimal search*. Academic Press.

Stone, L. D., Royset, J. O., & Washburn, A. R. (2016). *Optimal search for moving targets*. Springer.

Tenaud, E. (2022). Optimisation de fonctions boîtes noires avec et sans contraintes. *23ème congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*.