Generalized Periodic Vehicle Routing and Maritime Surveillance

Maria Fleischer Fauske[1], Carlo Mannino[2], and Paolo Ventura[3]

[1]Norwegian Defence Research Establishment, NO-2027 Kjeller, Norway, maria.fauske@ffi.no
[2]SINTEF ICT Applied Mathematics, NO-0314 Oslo, Norway; and University of Oslo, NO-0316 Oslo, Norway, carlo.mannino@sintef.no
[3]IASI-CNR, Rome, Italy, paolo.ventura@iasi.cnr.it

August 14, 2018

Abstract

Planning maritime surveillance activities in military operations and in long-term defense planning is a huge task that is done manually today. As maritime surveillance resources are extremely expensive, the potential cost savings of using optimization models to do such planning are large. In this paper, we developed a methodology for making maritime surveillance planning more efficient. The purpose of our tool is to find routes for the force elements involved in maritime surveillance operations where the goal is to keep a maritime picture sufficiently updated. Our problem may be viewed as a variant of the classical Periodic Vehicle Routing Problem, but it differs from this problem in some major aspects. To cope with the specific issues of our problem, we introduce a novel time-indexed formulation, where each variable is associated with a set of contiguous time periods. To tackle instances of practical size, we applied delayed column generation and developed efficient heuristic techniques. We show how our approach can plan up to 72-hour realistic missions with routing ships.

Keywords: Mixed integer linear programming, dynamic programming, column generation, maritime surveillance, military operations planning

1 Introduction

The Norwegian coastline is one of the longest in the world. Norways's economic and fishery zone consists of almost 2 000 000 km^2 of waters, which is about 80 percent the size of the Mediterranean sea. Surveillance of this area is a huge task which involves several Norwegian authorities, such as the Coast Guard and the Navy, including technical facilities, e.g. maritime patrol aircraft, stationary sensor systems, and satellites. The Norwegian Defence Research Establishment (FFI) has developed several models and tools to improve the use of force elements (FE) in maritime surveillance (MS), and to analyze future needs for maritime surveillance capacities. A simulation model used by FFI for this purpose is described by Vatne and Gisnås

(2014). In this paper we describe a model for optimizing the movements of force elements in a maritime surveillance operation.

Maritime surveillance resources are extremely expensive. Planners of maritime surveillance operations always want to use as few resources as possible, as effectively as possible. In this paper we develop a methodology for making the planning of maritime surveillance more efficient. A working tool may be included as a part of the planning process for organizations doing maritime surveillance. In Norway, such a tool is especially useful for the Norwegian Joint Headquarters, which is responsible for planning maritime surveillance missions in Norway. At FFI, the tool is suitable for studying alternative future force structures in long-term defence planning (LTP). LTP at FFI is conducted as a direct support to the Norwegian Ministry of Defence, about the development of the Norwegian Armed Forces.

There may be several purposes for monitoring a maritime area. One may be to keep a recognized maritime picture sufficiently updated. Another may be to maximize the probability of detecting unknown vessels, or to maximize the ability to react to a detection in order to obtain more information. The model we describe in this paper is used to study the movement of force elements in order to keep a recognized maritime picture sufficiently updated. What is considered sufficient will depend on both location and situation. Some areas need to be scanned for vessels at a higher frequency than others. In peace time, the scanning frequency requirement will be lower than in crisis or war. In our study we do not consider details concerning e.g. technologies, communication or decisions. It is a study of structures, where we look at the different force elements' ability to collect information. This ability is mainly a function of the force elements' movement (speed) and sensor capabilities (range). In our study we are given a fleet of FEs. The maritime area of interest is partitioned into a grid P of hexagonal cells. The FEs move from cell to cell. The FE's sensor range decides which cells the FE can observe at any given time. Each cell should be observed a certain amount of times during the planning horizon, and the lag between successive observations must not exceed a given threshold, which depends on the given cell. Some cells should be observed continuously, while other, less important areas, only have to be observed e.g. every 24 or 48 hours.

The *Maritime Surveillance Problem* (MSP) may be viewed as a variant of the classical *Periodic Vehicle Routing Problem* (PVRP) in which one wants to route and schedule a fleet of vehicles to repeatedly visit a number of clients scattered on the territory (see Cacchiani et al. (2014); Mingozzi (2005); Pirkwieser and Raidl (2012)). Each client must be visited with a given frequency during the planning horizon, according to some predefined patterns (e.g Monday and Thursday or Tuesday and Friday, . . .). The MSP differs from the standard PVRP in some major aspects. First, cells (clients) do not need to be directly visited, but can simply be observed. In other words, many cells can be observed without the route of an FE going through it. More precisely, PRVP corresponds to the special MSP case when every cell p can be observed only from the same cell p, so MSP generalizes PVRP. A second aspect is that in PVRP, the planning horizon is typically subdivided into days, every route is started and completed in a single day of the planning horizon. Then a number of subsets (*patterns*) of days are defined. Each client must be assigned one pattern so that the days in that pattern satisfy the periodicity requirement of the client. Then a minimum cost set of routes should be assigned to cover the selected patterns. In contrast, in MSP the basic period may differ significantly from cell to cell. So, some cells must be observed once a day, while others every hour or even continuously. Also, routes may last several periods, or less than one period.

To cope with all these additional issues, in this paper we introduce a novel time-indexed Integer Linear Program (ILP) formulation for the problem. Time-indexed formulations were first introduced in the context of job-shop scheduling problems (see Dyer and Wolsey (1990)), in order to return tighter bounds than the more natural (big-M) formulations (for an interesting theoretical comparison see Queyranne and Schulz (1994)). The planning horizon is discretized, and a binary variable is associated with every operation and every lag in the planning horizon. Time-indexed formulations have been successfully applied to different transport routing and scheduling problems. For instance, in train scheduling Harrod (2011) and Caprara et al. (2002); airplane routing and scheduling Avella et al. (2017) and Kjenstad et al. (2013); and classical traveling salesman such as in Dash et al. (2012) and Ilavarasi and Joseph (2014).

In standard time-indexed formulations for vehicle scheduling, the planning horizon is partitioned into a finite set of periods, and each binary variable is associated with one vehicle and one period. In order to represent multiple periodicities, in our approach we introduce a new set of binary variables. Indeed, the new approach allows for an effective representation of multiple periodicity observations as well as of the distinction between visiting a cell while traveling, and observing a cell. In particular, besides the standard time-lags partition of the planning horizon used for modeling travels, we introduce a new discretization by covering the horizon with families of time intervals. In each family, the time intervals are equal in size. Each interval is associated with a specific set of observations. For instances of realistic size, the resulting ILPs turn out to be very large. To tackle this difficulty, we developed a master/slave, column generation approach which allowed us to solve to optimality small instances and to find good solutions to medium sized instances in reasonable computing time. We also prove that the pricing problem is NP-hard; however we show how some simple heuristic approaches may provide satisfactory solutions.

There are a few papers in the literature describing how to find optimal routes for single or multiple vehicles in the context of maritime surveillance, using different models and solution techniques. However, to our knowledge no such papers actually cover all the aspects which must be considered when solving (our version of) the MSP. For instance, Marlow et al. (2007) describes a problem where the purpose is to route an aircraft such that the number of visited classified targets is maximized. A variation of the classical TSP is used to model the problem, then it is solved by classical heuristic approaches (such as genetic algorithms and 2-opt). Dridi et al. (2012) describes a multi-objective optimization approach for solving a maritime surveillance problem where a set of resources are assigned to a specific set of tasks: again the periodical nature of the problem and the routing part are neglected. Grob (2006) describes a simulation model to study the deployment of FEs to obtain a recognized surface picture. An interesting, very recent paper on aircraft mission planning by Quttineh et al. (2015) presents a few features in common with our approach. In particular, they also exploit a time-indexed ILP formulation for their problem, equipped with a suitable column generation. They also need to route several vehicles. However, since they are planning single missions with several coordinated aircraft, they do not tackle periodicity. Also, each target point must actually be "visited" by an aircraft. A very detailed model, which also incorporates speeds and observations, is provided in Grob (2006). Again, however, periodicity is neglected. Finally, in Stumpt and Michael (2011), an application to robot surveillance is considered. Interestingly, the paper tackles the problem of periodic visits to the targets. The idea is to create several copies of a specific target, each representing a distinct visit, with an associated time window. This is an approximation of the original period constraint, and it may return solutions which are far from being periodic, as can easily be seen. In contrast, our modeling approach is exact, in that the required visiting period for each cell is respected by every feasible solution to our model. Indeed the observations of a given cell may happen more often than the minimum required frequency.

2 The Problem

Roughly speaking, the MSP consists of finding an operational plan for a given force structure doing maritime surveillance in a given area during a specific planning horizon H. In particular, we want a plan which maximizes the ability to keep a recognized maritime picture sufficiently updated. In the model, this ambition is given by the desired observation frequency for specific regions or points in the area of interest. Given that the area of interest will in most cases be too large for the force structure to fulfill the ambition for all cells, we will content ourselves with maximizing the degree of ambition fulfillment.

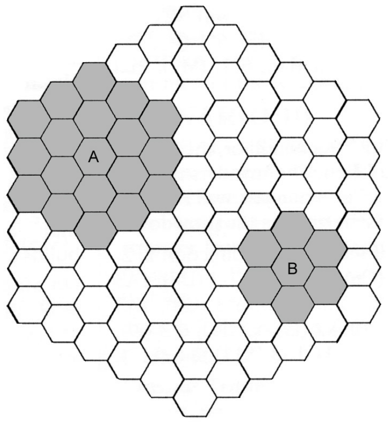
In order to represent our problem, the area of interest is partitioned into a grid P of hexagonal cells $p_1, p_2, \ldots, p_{|P|}$. Two adjacent cells are considered to be at distance 1. The planning horizon is discretized in *time-steps* or *time-lags*, i.e. $H = \{1, 2, \ldots, |H|\}$. A cell p must be observed at least once every T_p periods. We are given a fleet F of Force Elements (FE), such as vessels, aircraft, etc. Each Force Element $f \in F$ is characterized by a number of parameters, such as

- σ_f: speed, given as number of adjacent cells that f can cross in a single time-step.
- ρ_f: sensor range; f can observe any cell at distance at most ρ_f from it (see Figure 1). Denote by $Q_f(p)$ such set of cells and recall that, when f is in cell $p \in P$, it is assumed to be in the center of p and that two adjacent cells are at distance 1.
- λ_f: endurance, i.e. the number of time-lags the FE can operate before it has to go back to its base. When f can travel for the entire planning horizon H without having to go back to its base, $\lambda_f = |H|$.
- ϕ_f: initial location.

These parameters will vary significantly among the FEs. Satellites and stationary sensor systems have a large sensor range, but do not move. An aircraft may travel 50 times faster than a vessel. The endurance of an aircraft is also different from the endurance of a vessel. An aircraft is typically only airborne for a few hours, while a vessel can sail for weeks. In other words, some of the FEs in the model must return to their base after a short amount of time, while others may start in one cell and end up in a totally different cell at the end of the planning horizon. FEs with similar features and the same initial location are grouped, so defining a partition $\{F_1, \ldots, F_k\}$ of F. Briefly, the MSP can be stated as follows:

Problem 2.1 *Given a set of cells P and a fleet F of FEs with their specific features, find feasible routes for the FEs so to maximize the total number of cell observations.*

Figure 1 A: sensor range = 2, B: sensor range = 1



3 The Model

In this section we introduce a pure binary linear programming model for the MSP. As mentioned in the introduction, the MSP can be considered as a variant of the PVRP, which in turn can be viewed as a generalization of the classic VRP. Concerning the discretized planning horizon H, we assume that time-step $t \in H$ starts at time $t-1$ and ends at time t, and it corresponds to the half-open real interval $[t-1, t)$. Next, we need to model the movements of the force elements in the area of interest; and we need to represent and combine this with the demand of periodic observations of cells.

Force Element trajectories: time-expanded network. We start by representing the movements of a single force element (FE) f. To this end, we introduce the time-expanded network $G_f = (N, A)$ associated with the cells P, planning horizon H and FE f, as the following directed graph. We assume that the trajectory of f is defined by joining with segments the centers of the cells it passes by. At each time-step, the number of crossed cells is, at most, the value of the speed σ_f. Then, the maximum length of a single trip of f is $S_f = \sigma_f \cdot \min\{H, \lambda_f\}$. Informally, S_f may be viewed as an *expanded* planning horizon for force element f, with shorter time-lags, in order to take into account the fact that in one original time-lag of H, f can visit up to σ_f cells. In contrast, in one time-lag of this expanded horizon, f can reach at most one neighboring cell. This fact will be represented by the following time-expanded network:

- node set: there is a node for every cell and every $s = 0, \dots, S_f$, namely $N = \{(p, s) : p \in P, s = 0, \dots, S_f\}$. When no confusion arises, we denote node (p, s) by ps.
- arc set A: there is an arc $e = ((p, s), (q, s+1))$ if and only if $p = q$ or q is a neighboring cell of p. Observe that the first case represents the situation in which the FE f is moving slower than its full speed σ_f.

The following is a fundamental property of G_f:

Property 3.1 *The time-expanded network $G_f = (N, A)$ does not contain directed cycles.*

The proof is immediate, since every arc is of type $(ps, qs + 1)$.

We recall that for any directed graph $F = (W, A)$, a *topological order* of F is an order of the nodes $w_1, w_2, \ldots, w_n$, such that $(w_i, w_j) \in A$ implies $j > i$. In other words the arcs can only be directed from lower to higher indexed nodes. It is well known that a graph F admits a topological order if and only if F does not contain directed cycles (see Schrijver (2003)).

A topological order $\succ$ of the time expanded network $G_f = (N, A)$ can be immediately obtained by arbitrarily ordering the cells in P and by letting $(q, v) \succ (p, s)$ if (i) $v > s$ or (ii) $v = s$ and $q > p$.

Any directed path (or *route*) in the graph G_f that starts from node $(\phi_f, 0)$ corresponds to a possible trajectory of the FE f in the observed area. Since the graph contains no parallel arcs, we may represent such a path as an ordered sequence of nodes of G_f (and omit to write the arc between successive nodes, which is uniquely identified). In particular, the di-path $(p_0 0, p_1 1, \ldots, p_l l)$ with $p_0 = \phi_f$ has a straightforward interpretation: at any step $0 \leq s \leq l$ the FE f is in (arrives at, if $s > 0$) the center of cell p_s (from the center of cell p_{s-1}, if $s > 0$). This means that, at each time $t = 1, \ldots, \lfloor \frac{l}{\sigma_f} \rfloor$, f goes from cell $p_{(t-1)\cdot\sigma_f}$ to cell $p_{t\cdot\sigma_f}$ in σ_f steps. In particular, the first cell to be visited after $p_{(t-1)\cdot\sigma_f}$ will be $p_{(t-1)\cdot\sigma_f+1}$, and so on following the sequence $p_{(t-1)\cdot\sigma_f}, p_{(t-1)\cdot\sigma_f+1}, p_{(t-1)\cdot\sigma_f+2}, \ldots, p_{t\cdot\sigma_f-1}$.

Modeling periodic observations. For every cell $p \in P$ we assume that the maximum time-lag T_p allowed between successive observations of p is an integer number of time-lags (with $T_p \leq |H|$). Now, let $I = \{t, \ldots, t + T_p - 1\}$ be an interval of size T_p in H. Clearly, H contains precisely $|H| - T_p + 1$ such intervals, namely the family of *observation intervals* $\mathcal{I}(p) = \{\{1, \ldots, T_p\}, \{2, \ldots, T_p + 1\}, \ldots, \{|H| - T_p + 1, \ldots, |H|\}\}$.

We will exploit the following property:

Property 3.2 *Cell p is observed at least once every T_p periods (in the time planning horizon) if and only if it is observed (at least once) in every interval of H of size T_p, namely in every observation interval of $\mathcal{I}(p)$.*

The proof is immediate and we omit it.

Now, let p be a cell and $I \in \mathcal{I}(p)$. Then we say that the ordered couple (p, I) is an *observation* and we denote by $\mathcal{O} = \{(p, I) : p \in P, I \in \mathcal{I}(p)\}$ the set of all possible observations. Moreover, let $u = (q, s)$ be a node in the time expanded network G_f. Recall that u corresponds to the possible location of f in cell q at time $\lfloor \frac{s}{\sigma_f} \rfloor$. Therefore we say that u *covers* (p, I) if i) $p \in Q_f(q)$ (i.e. p can be observed by f from q, according to the sensor range ρ_f); ii) $\lfloor \frac{s}{\sigma_f} \rfloor \in I$. By extension, we also say that a route of G_f covers (p, I) if it contains a node u that covers (p, I) and we denote by $R(p, I)$ the set of all such routes. Moreover, we denote by $O(r)$ the set of observations covered by route r.

Modeling the problem. Given the fleet of available FEs and corresponding time-expanded networks, we denote by $\mathcal{R}$ the family of all feasible routes. Every route $r \in \mathcal{R}$ is associated with a specific FE f of the fleet and in particular it corresponds to a directed path of the time-expanded network G_f. A solution to the MSP consists of a set of routes $R = \{r_1, \dots, r_{|F|}\} \subseteq \mathcal{R}$ and we denote by $= O(R) \subseteq \mathcal{O}$ the set of observations covered by at least a route of R, i.e. $O(R) = \bigcup_{r \in R} O(r)$. By Property 3.2, finding a set of routes for our available fleet such that every cell p is observed at least every T_p periods is equivalent to finding a subset of feasible routes $R \subseteq \mathcal{R}$ such that $O(R) = \mathcal{O}$. In general, this request may be impossible to achieve, so we will content ourselves with the more reasonable objective of maximizing the total number of covered observations.

Then the MSP can be stated as follows: find a subset $R \subseteq \mathcal{R}$ of given size such that maximizes $|O(R)|$. We may associate a value with every observation and maximize the total value of the covered observations.

We are now ready to formulate the MSP as a binary linear program. For any route $r \in \mathcal{R}$, let y_r be a binary variable which is 1 if and only if route r is selected. Also, with every $(p, I) \in \mathcal{O}$ we associate a binary variable x_{pI} which is 1 if and only if (p, I) is covered by some selected route. The routes in $\mathcal{R}$ are grouped in clusters $R_1, R_2, \dots R_k$ each corresponding to a vehicle type (i.e. set of vehicles with the same values for σ, ρ, λ, and ϕ), and we let n_q be the number of vehicles of type q, for $q = 1, \dots, k$.

We associate a value $V_{pI} \geq 0$ with every observation $(p, I) \in \mathcal{O}$ and then look for solutions maximizing the total value of the covered observations. Hence the MSP can be written as

$$
\begin{array}{lll}
 & \max & \displaystyle\sum_{(p,I) \in \mathcal{O}} V_{pI} x_{pI} \\
s.t. & & \\
(i) & & x_{pI} - \displaystyle\sum_{r \in R(p,I)} y_r \leq 0 \quad (p, I) \in \mathcal{O}, \\
(ii) & & \displaystyle\sum_{r \in R_q} y_r \leq n_q \quad q = 1, \dots, k, \\
 & & y \in \{0,1\}^{\mathcal{R}}, \;\; x \in \{0,1\}^{\mathcal{O}}.
\end{array}
\tag{1}
$$

By constraints (1.i), a variable x_{pI} can assume value 1 only if observation (p, I) is covered by some selected route. Constraints (1.ii) ensure that not too many routes are chosen for each class of vehicles.

Problem (1) contains a large number of rows (one for each observation) and potentially an exponential number of columns (one for each route for each vehicle type): this may result in intractably large instances. In order to tackle this difficulty, we apply delayed row and column generation. We assume the reader to be familiar with the approach (for details see Alvras and Padberg (2001); Desaulniers et al. (2006)). We start by solving a problem corresponding to a subset of columns, relative to a subset of routes $\bar{\mathcal{R}} \subseteq \mathcal{R}$. Correspondingly, we have a set $\bar{O} = \bar{O}(\bar{\mathcal{R}})$ of observations, namely the subset of pairs $(p, I) \in \mathcal{O}$, that can be covered by at least one route in $\bar{\mathcal{R}}$. This is the initial *restricted master program.* The linear relaxation (RelMast) of the restricted master is solved to optimality (after including upper bounds on the relaxed binary variables):

$$
\begin{array}{lll}
 & \max & \sum_{(p,i)\in\bar{O}} V_{pI}x_{pI} \\
s.t. & & \\
(i) & & x_{pI} - \sum_{r\in R(p,I)\cap\bar{\mathcal{R}}} y_r \le 0 \quad (p,I)\in\bar{O}, \\
(ii) & & \sum_{r\in R_q\cap\bar{\mathcal{R}}} y_r \le n_q \quad q=1,\dots,k, \\
(iii) & & x_{pI} \le 1 \qquad (p,I)\in\bar{O}, \\
(iv) & & y_r \le 1 \qquad r\in\bar{\mathcal{R}}, \\
 & & y\in I\!R_+^{\bar{\mathcal{R}}}, \qquad x\in I\!R_+^{\bar{O}}
\end{array}
\tag{RelMast}
$$

Observe that the restricted master depends (only) on the current set $\bar{\mathcal{R}}$ of columns, since the restricted observation set $\bar{O} = \bar{O}(\bar{\mathcal{R}})$ also depends only on $\bar{\mathcal{R}}$. We make explicit this dependency by denoting the current master as RelMast($\bar{\mathcal{R}}$). The restricted master contains thus a subset of variables and a subset of constraints of the original program. Then, we add new variables and constraints to the restricted master, solve again the associated linear relaxation, and iterate. The process terminates either if we cannot further add "profitable" columns or time limit is reached. More details will be given in Section 4.

The column generation (CG) algorithm is inspired by the simplex method for linear programming (see e.g., Bertsimas and Tsitsiklis (1997)). In particular, the new variables are chosen among those with largest reduced cost. Recall also that the reduced costs of the variables already included in the current master program are non-positive. To compute reduced costs, we need the optimal dual variables (with respect to the current restricted master). So, after solving the relaxation of the current restricted master program, let $\bar{\pi}$, $\bar{\lambda}$ be the optimal dual vectors associated with constraints RelMast.i, RelMast.ii respectively[1]. Note that at the next iteration, the newly introduced variables may extend existing constraints or may appear only in new constraints. In contrast, by construction, the new constraints only contain new variables.

Next, for $(p,I)\in\mathcal{O}$, let $\bar{V}_{pI}$ be the reduced cost of variable x_{pI}. Then, for $(p,I)\in\bar{O}$, we have $\bar{V}_{pI}\le 0$, while, for $(p,I)\in\mathcal{O}\setminus\bar{O}$, we have $\bar{V}_{pI} = V_{pI}$ (because the variable does not extend any constraint of the current master).

For a route $r\in\bar{\mathcal{R}}$ and associated variable y_r, we have a reduced cost $\bar{c}_r\le 0$, while, for $r\in R_q\setminus\bar{\mathcal{R}}$, the reduced cost of variable y_r reads $\bar{c}_r = \sum_{(p,I)\in O(r)\cap\bar{O}} \bar{\pi}_{pI} - \bar{\lambda}_q$.

If we include a new route $r\in\mathcal{R}\setminus\bar{\mathcal{R}}$, we also include the set of corresponding additional observations $O(r)\setminus\bar{O}$. Consequently, the new master program will contain, besides all former variables and the new variable y_r, also variables x_{pI}, for $(p,I)\in O(r)\setminus\bar{O}$. So, instead of looking to individual variables and individual reduced costs, we will look at the total contribution to the reduced cost of all new variables. The sum of the reduced costs "associated with" route r

[1]Note that we do not need the dual variables associated with the other constraints to compute the reduced cost of "new" variables. To see this, consider the column generation as an iteration of the primal simplex method applied to the linear relaxation RelMast($\mathcal{R}$) of the overall original Problem (1). Let $\bar{y},\bar{x}$ the current solution. Then a "new" column/variable y_r to generate corresponds to a non-basic variable (implying $\bar{y}_r = 0$) with positive reduced cost. But then y_r does not appear in any constraint of type RelMast($\mathcal{R}$).iii and it appears in only one constraint of type RelMast($\mathcal{R}$).iv, namely $y_r\le 1$. However, since $\bar{y}_r = 0$, then the associated dual variable is 0. A similar argument can be used for the x variables.

is then

$$\bar{c}(r) + \sum_{(p,I)\in O(r)\setminus\bar{O}} V_{pI} = \sum_{(p,I)\in O(r)\cap\bar{O}} \bar{\pi}_{pI} - \bar{\lambda}_q + \sum_{(p,I)\in O(r)\setminus\bar{O}} V_{pI}$$

.

Now, for each observation $(p, I) \in \mathcal{O}$, we define a weight

$$w_{pI} = \begin{cases} \bar{\pi}_{pI} & \text{if } (p,I) \in \bar{O} \\ V_{pI} & \text{if } (p,I) \in \mathcal{O}\setminus\bar{O} \text{ otherwise.} \end{cases} \tag{2}$$

For $r \in \mathcal{R} \setminus \bar{\mathcal{R}}$, let $w(r)$ be the total weight of the associated observations, namely $w(r) = \sum_{(p,I)\in O(r)} w_{pI}$. Therefore, the column generation task corresponds to finding a route r in a time-expanded network of an FE f^q of type q which maximizes $w(r) - \bar{\lambda}_q$.

4 Column generation

In this section we describe alternative approaches to identifying the next route(s) to add to the restricted master program. We can iteratively focus on a specific FE type and then find the best route for each type. Then, any feasible route of f will be identified by a directed path in the expanded network G_f (as described in Section 3) that starts from the node $(\phi_f, 0)$ (denoted by o) and ends up in a *final* node $(p, \lambda_f \cdot \sigma_f)$, with $p \in P$. We denote by $E \subseteq \{(p, \lambda_f \cdot \sigma_f) | p \in P\}$ the set of candidate final nodes. Notice that the final cell of the route can either: i) be given for any force element type; ii) coincides with ϕ_f, if the route must actually be a tour (a closed route); iii) be any cell of the grid ($E = \{(p, \lambda_f \cdot \sigma_f) | p \in P\}$). All three possibilities can be implemented by the algorithms and models we will discuss in the following.

Now we can state more formally our pricing problem.

Problem 4.1 *[Largest Set Path Problem] (LSPP) Let $G = (N, A)$ be an acyclic, connected directed graph, with one source $o \in N$. We are given a ground set O, and a weight function $w : O \to I\!R_+$. With every arc $e \in A$, we associate a subset O_e of the ground set O. For any directed path $Q = (e_1, \ldots, e_q)$ in G, we let $O(Q) = \bigcup_{e\in Q} O_e$ and we let $w(Q) = \sum_{\omega\in O(Q)} w(\omega)$ be the weight of Q. We want to find a directed path Q^* in G of maximum weight $w(Q^*)$.*

Note that an instance of the LSPP is identified by the tuple $(G(N, A), O, w, O_1, \ldots, O_{|A|})$. In our application, the ground set O is the set $\mathcal{O}$ of observations, whereas the weights correspond to the values $w_{p,I}$ as in (2). Moreover, for any $e = (u, v) \in A$, let $O_e \subseteq O$ be the set of observations that are covered by node u (and therefore by arc e) and, for each observation $(p, I) \in \mathcal{O}$, let $A(p, I) = \{e : (p, I) \in O_e\}$. We remark here that, for a given node u, every outgoing arc will cover the same set of observations, which are indeed associated with u. In the following, we will address LSPP also as the *pricing* problem and, since each FE f is associated with a graph G_f, we denote it by $LSPP(w, f)$.

To model the above problem as an ILP, we proceed in standard fashion by introducing a binary variable z_e for every $e \in A$, which is 1 if and only if arc e is taken in the chosen path r. Also, similarly to what we did in Problem (1), for each observation $(p, I) \in \mathcal{O}$, we introduce a binary variable x_{pI}, which will be 1 if and only if $(p, I) \in O_e$ for some $e \in r$. Also, for each $v \in N$, $\delta^+(v)$ and $\delta^-(v)$ denote the outgoing and incoming star of v respectively. The following is an ILP formulation for Problem 4.1:

$$
\begin{array}{lll}
 & \max & \sum_{p\in P}\sum_{I\in\mathcal{I}(p)} w_{pI}x_{pI} \\
s.t. & & \\
(i) & & \sum_{e\in A(p,I)} z_e - x_{pI} \geq 0 \quad (p,I)\in\mathcal{O}, \\
(ii) & & \sum_{e\in\delta^+(o)} z_e = 1 \\
(iii) & & \sum_{e\in\delta^+(v)} z_e - \sum_{e\in\delta^-(v)} z_e = 0 \qquad v\in N\setminus\{E\cup\{o\}\} \\
 & & z\in\{0,1\}^A, \;\; x\in\{0,1\}^{\mathcal{O}}
\end{array} \tag{3}
$$

Constraints (3.i) allow variable x_{pI} to be 1 only if the corresponding observation is covered by some selected arc. Constraints (3.ii) and (3.iii) ensure that $z \in \{0,1\}^A$ is the incidence vector of a directed path starting from node o and ending in one of the nodes in E. Observe that, in contrast with the standard integer programming formulation for the general case (see, e.g., Drexl and Irnich (2014)), we do not need to include here the exponentially many subtour elimination constraints, as G is acyclic (Property 3.1).

The pricing problem LSPP is hard to solve, as we will show formally in Section 5. In the following, we present three fast heuristic algorithms and two exact approaches to the problem. The computational strength of the methods will be discussed in Section 7.

Let $G = (N,A)$ be a directed acyclic graph, with lengths $l \in I\!R^A$ associated with the arcs. Let $1,\dots,n$ be a topological order of the nodes of G. We want to find the length L_v of a maximum length path P_v^* from vertex 1 to any vertex $v \in E$. We denote by $N^-(v)$ the negative neighborhood of v, namely the set of nodes $u \in N : uv \in A$.

Algorithm 1 Dynamic Programming (DP)

```
1: $L_v = -\infty$, $v = 2,\dots,n$. $L_1 = 0$
2: for $v = 2$ to $n$ do
3:     $L_v = max_{u\in N^-(v)}\; L_u + l_{uv}$
4:     $pred[v] := argmax_{u\in N^-(v)}\; L_u + l_{uv}$
5: end for
```

The longest path P_v^* can be easily constructed by the vector $pred[.]$ which gives, for each $v = 2,\dots,n$, the previous node on the optimal path from node 1.

The first of our heuristic procedures is basically the straightforward application of Algorithm DP to the expanded network $G = (N,A)$ with arc length $l_e = \sum_{(p,I)\in O_e} w_{pI}$ for $e = (u,v) \in A$. In other words, the length l_e of an arc e is the total (reduced) value of the observations which are carried out from e. The problem with this approach is that when we consider a route r of G, an observation $(p,I) \in \mathcal{O}$ can be covered by different arcs of the route, and so the same observation can contribute more than one time to the total length of the route. In other words, for any given route r, the length $l(r) \geq w(r)$ is not less than the weight of the route. So the longest route r^* (with respect to l) may not be the one with maximum reduced value w^*. However, $l(r^*)$ provides an upper bound on the optimal value.

A possible way to refine Algorithm DP is to take into account the actual total value of the observations, so that a specific observation is considered only once.

To this end, for each $v \in N$ we introduce the set $Obs[v]$, which is the set of observations carried out in the sub-route from 1 to v (according to the vector $pred$). For a set of observations $O \subseteq \mathcal{O}$, let $\bar{w}(O) = \sum_{(p,I)\in O} w_{pI}$ be its value.

Algorithm 2 Modified Dynamic Programming (MDP)

```
1: $Obs[v] = \emptyset$ for all $v = 1, \dots, n$
2: for $v = 2$ to $n$ do
3:     $u^* = argmax_{u \in N^-(v)}\ \bar{w}(Obs[u] \cup O_{(u,v)})$
4:     $pred[v] := u^*$
5:     $Obs[v] = Obs[u^*] \cup O_{(u^*,v)}$
6: end for
```

The crucial step is the first in the **for** loop, where we choose the predecessor to be the node u^* that maximizes the value of the current route going through u^* plus the contribution of the additional observations carried out in arc (u^*, v). Moreover, observe that for each node v of the network, we calculate and store the set of observations associated only to the most promising $(1-v)$-path. This implies that the final path P^* provided by the algorithm could not be the optimal one. Also notice that both algorithms DP and MDP require to visit every node and every arc of the graph.

In order to speed up the procedure and reduce memory usage, we introduce a simplified, “greedy" version of Algorithm MDP. The idea is to visit only a small subset of nodes and arcs of G. First, observe that the time-expanded network G_f is a layered graph. Indeed, recall that the nodes are associated with pairs (p, s) where p is a cell and $s \in S$ is a time-step in the expanded planning horizon $S = \sigma_f \cdot \lambda_f$. In particular, there is a layer of nodes for every time-step $s \in S$ and the arcs of G_f can only go from nodes in a layer to nodes in the next layer. Consequently, every directed path $P = (v_1, v_2, \dots)$ necessarily visits nodes according to the sequence of layers, and so v_i belongs to layer i. In the following algorithm, at iteration s we select the s-th node v_s from layer $s \in S$. The $s+1$-th node v_{s+1} is then restricted to belong to the set $N^+(v_s)$ of the neighbors of v_s in layer $s+1$.

Algorithm 3 Greedy Dynamic Programming (GDP)

```
1: $v = o$, $Obs = \emptyset$
2: for $s = 1$ to $S$ do
3:     $u^* = argmax_{u \in N^+(v)}\ \bar{w}(Obs \cup O_{(v,u)})$
4:     $pred[u^*] := v$
5:     $Obs = Obs \cup O_{(v,u^*)}$
6:     $v := u^*$
7: end for
```

Finally, we discuss how to solve LSPP to exact optimality. One way to do that is to solve the ILP program (3) using the branch & bound procedure of a commercial ILP solver. In the following, we will refer to this method as the EXI algorithm. In the following, we also present an exact combinatorial algorithm to solve the pricing problem. Such an algorithm follows a branch & bound scheme based on a recursive enumeration of the paths of the graph G.

We denote by $maxPath(u, O)$ the maximum value of a subset of observations $O \subseteq \mathcal{O}$ which can be covered by a path from the origin o to node u. More formally, for all possible paths from o to u, let P_u^* be one that maximizes $w(O(P_u^*) \cap O)$. Then, we define $maxPath(u, O) = w(O(P_u^*) \cap O)$. The following recursive expression holds for $maxPath(v, O)$.

$$maxPath(v, O) = max_{u \in N^-(v)} maxPath(u, O \setminus O(u, v)) \tag{4}$$

Note that, if the maximum is attained for $u = t$, and $P^*(t)$ is the path associated with $maxPath(t, O \setminus O_{(u,t)})$, then $P_v^* = P_w^* \bullet (t, v)$, that is the optimal path associated with $maxPath(v, O)$ is obtained by concatenating the optimal path associated $maxPath(t, O \setminus O_{(u,t)})$ with arc (t, v). Then, if z is the destination node, our problem is equivalent to computing $maxPath(z, \mathcal{O})$, with the optimal route being P_z^*.

We implemented the recursive formula (4) by Algorithm EXC[2].

We remark that, differently from most models in the VRP literature, here we could not apply some adaptation of the "standard" labeling pricing algorithm. Basically, all the pricing algorithms presented for different VRP variants, from the seminal paper of Christofides et al. (1981) up to the more recent contributions of Fukasawa et al. (2006), Baldacci et al. (2011) and Martinelli et al. (2014), rely on the fact that the pricing problem can be solved by finding a minimum cost simple path P_v^* (which often has to satisfy additional capacity constraints or time windows) from the depot to each node v in the input graph. Therefore, its value $c(P_v^*)$ can be calculated by the recursive formula $c(P_v^*) = \min \{c(P_u^*) + d_{uv} | u \in N^-(v)\}$ (where d_{uv} is the cost of arc (u, v)) and the main computational effort of the algorithm is devoted to ensure, by sophisticated relabeling techniques, that such a path stays simple.

On the contrary, for our problem LSPP, the cost of an optimal path cannot be calculated by means of a similar recursive formula. This is basically because the same observation can be covered by different nodes of the graph G (corresponding to different cells of the original grid visited in different instants of time) and, therefore, a sub-path of a largest set path is not necessarily a largest set path to a previous node. Moreover, standard dominance techniques based on capacity of time window constraints cannot be exploited in LSPP. However, we could apply another "classical" dominance technique, namely *upper bounding*, to limit the size of the search.

In particular, consider a path P_z from the origin node o to a destination node $z \in E$ and let $v \neq z$ be a node in P_z. Then, let P_v be the sub-path of P_z from o to v and let P_{vz} be the remaining sub-path, i.e. $P = P_v \bullet P_{vz}$. We have:

$$w(O(P_z)) \leq w(O(P_v)) + w(O(P_{vz}))$$

because P_v and P_{vz} may share some observations. Now, let UB_{vz} be any upper bound on the value of any path from v to z. Then we have:

$$w(O(P_z)) \leq w(O(P_v)) + w(O(P_{vz}) \leq w(O(P_v)) + UB_{vz}$$

[2]For simplicity, although the formula is a backward recursion, our Algorithm EXC is presented as a forward recursion

.

Now, suppose we have at hand a feasible solution (*incumbent*) with value LB. Let P_v^* be an optimum path from o to v, namely a path maximizing $w(O(P_v))$ for any path P_v from o to u. If we have that

$$w(O(P_v^*)) + UB_{vz} \leq LB$$

then no path improving the incumbent can go through node v.

We apply this idea in Algorithm EXC, where UB_{vz} is pre-computed by Algorithm DP for each pair v, z.

Algorithm 4 Exact Combinatorial Algorithm (EXC)

```
 1: maxPath($(o, \emptyset)$)

 2: function maxPath($(u, O)$)
 3:     for $v \in N^+(u)$ do
 4:         if $(v = n)$ then
 5:             if $(w(O \cup O_{(u,v)}) > LB)$ then
 6:                 $LB = w(O \cup O_{(u,v)})$
 7:             end if
 8:         else
 9:             if $(w(O \cup O_{(u,v)}) + UB_{(vn)} > LB)$ then
10:                 maxPath($(v, O \cup O_{(u,v)})$)
11:             end if
12:         end if
13:     end for
14: end function
```

5 Complexity of the pricing problem LSPP

We show here that the Largest Set Path Problem Problem (Problem 4.1) is NP-hard, by reduction from the (unweighted) *Maximum Coverage Problem* (MCP), known to be NP-hard (see Hochbaum (1997)).

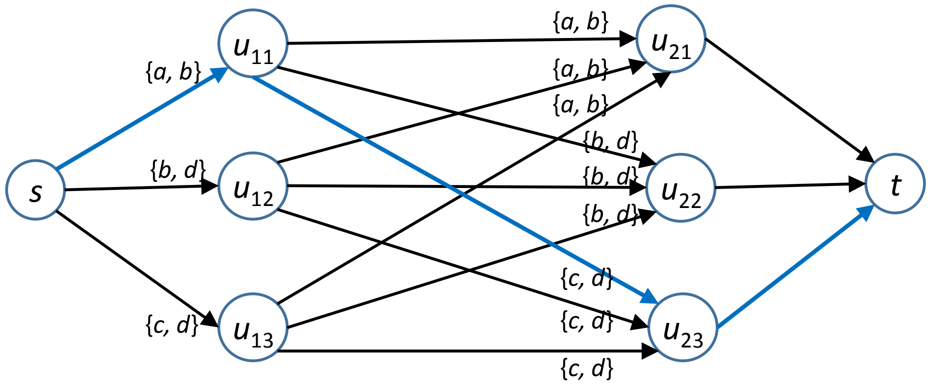
Problem 5.1 (Maximum Coverage Problem) *Given a ground set B, a family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of B, and a positive integer p, the* Maximum Coverage Problem *(MCP) is to find a family $\mathcal{S}^* \subseteq \mathcal{S}$ with $|\mathcal{S}^*| = p$ that maximizes $|\bigcup_{S_i \in \mathcal{S}^*} S_i|$.*

To simplify the following discussion, we may let $\mathcal{S}^*$ contain multiple copies of sets in $\mathcal{S}$. It is easy to see that this version is equivalent to the original one where $\mathcal{S}^*$ is a simple set. We have that:

Theorem 5.2 *The Largest Set Path Problem (Problem 4.1) is NP-hard.*

The proof is by reducing MCP to LSPP. Let $(B, \mathcal{S}, p)$ be an instance of the Maximum Coverage Problem and, w.l.o.g., assume $p \leq m$.

Figure 2 The picture illustrates the graph G associated with an MCP instance where $B = \{a, b, c, d\}, S_1 = \{a, b\}, S_2 = \{b, d\}, S_3 = \{c, d\}$, and $p = 2$. The blue arcs show an optimal solution of the corresponding LSPP instance.



Then construct an equivalent instance $(G = (N, A), O, w, O_1, \dots, O_{|A|})$ of the LSPP as follows. We let $O = B$, and $w(\omega) = 1$ for all $\omega \in O$. Note that since all weights are one, the LSSP reduces to a cardinality problem. Next, we introduce graph $G = (N, A)$ with $N = \{u_{ki} : k = 1, \dots, p, i = 1, \dots, m\} \cup \{s, t\}$ and we let A be partitioned into A_s, A_t, and A_u, where

- $A_s = (s, u_{1i}), \forall i = 1, \dots, m;$
- $A_t = (u_{pi}, t), \forall i = 1, \dots, m;$
- $A_u = (u_{ki}, u_{k+1j}), \forall k = 1, \dots, p-1, \forall i, j = 1, \dots, m,.$

Graph G is a layered graph and arcs go from every node in one layer to every node in the next layer. There are $p + 2$ layers: The first layer contains only the source node s and the last layer contains only the sink node t. Each intermediate layer contains a node for each set in $\mathcal{S}$: so node u_{ki} can be interpreted as the representative of set S_i in layer k.

Next, we let $O_e = S_i$ for each arc entering a node u_{ki}, with $k = 1, \dots, p$ and $i = 1, \dots, m$, and let $O_e = \emptyset$ for each arc entering t (see the example in Fig. 2). So, every arc entering the representative node of set $S_i \in \mathcal{S}$ (in every layer) is also associated with S_i.

Consider first an optimal solution $\mathcal{S}^* = \{S_{i_1}, \dots, S_{i_p}\}$ to MCP, with $i_1, i_2, \dots, i_p \in \{1, \dots, m\}$, and value $|S_{i_1} \cup S_{i_2} \cdots \cup S_{i_p}|$. We construct an equivalent solution of our instance of LSPP by choosing path $P^* = \{(s, u_{1i_1}), (u_{1i_1}, u_{2i_2}), \dots, (u_{pi_p}, t)\}$. We have that $w(P^*) = |\bigcup_{e \in P^*} O_e| = |O_{(s,u_{1i_1})} \cup O_{(u_{1i_1},u_{2i_2})} \cup \cdots \cup O_{(u_{pi_p},t)}| = |S_{i_1} \cup S_{i_2} \cdots \cup S_{i_p}|$: so P^* and $\mathcal{S}^*$ have the same value and the optimal solution to LSPP is at least as good as the optimal solution to MCP. Analogously, let $P^* = \{(s, u_{1i_1}), (u_{1i_1}, u_{2i_2}), \dots, (u_{pi_p}, t)\}$ be an optimal solution to LSPP,then we can construct an equivalent solution to MCP by letting $\mathcal{S}^* = \{S_{i_1}, \dots, S_{i_p}\}$ and again the two solutions have the same value. □

6 The overall algorithm

In this section, we finally summarize the overall solution algorithm for the MSP. We follow the exact solution scheme adopted in Baldacci et al. (2011). In particular, in the first loop (steps 2.−9., *Column Generation Loop*) we solve the linear relaxation of Problem (1) by column generation: at each iteration i, we: (i) solve the relaxation RelMast($\mathcal{R}^i$) of the current master problem; (ii) calculate the weight vector w^i as defined in (2); (iii) for any type q of FE, solve the pricing problem $LSPP(w^i, f^q)$ using one of the algorithms described above. If the pricing algorithm is exact (i.e., we use EXI or EXC) and at the last iteration (say z) no routes with positive reduced cost exist, then the value $rVal$ = RelMast($\mathcal{R}^z$) is the optimal value of the linear relaxation RelMast($\mathcal{R}$) of Problem (1), therefore providing an upper bound for the optimal (integer solution) value to the MSP problem.

In some classic (heuristic) approaches to vehicle routing (and other) problems one contents himself with the set of columns (variables) generated so far (i.e. those appearing in RelMast($\mathcal{R}^z$)). Then we can re-stipulate integrality on the variables (we call the generic problem IntMast(●)) and compute the best integer solution to IntMast($\mathcal{R}^z$). However, this solution is in general non-optimal for the original problem (1). This is because any optimal solution may contain at least one route which is not in $\mathcal{R}^z$.

In order to find the optimal solution to Problem (1) we can proceed as next. Let $\bar{\mathcal{R}} = \mathcal{R}^z$ be the set of routes at the termination of the column generation loop. We build a set $\mathcal{R}^* \supseteq \bar{\mathcal{R}}$ which contains all the routes appearing in (at least) an optimal solution. To this end, assume we have at hand a lower bound LB_{MSP} on the optimal integer value of MSP. Then, it is well known that any column with reduced cost not greater than $LB_{MSP} - rVal$ cannot appear in any solution improving LB_{MSP} (*reduced cost fixing*, see for instance Wolsey (1998)).

Then, we can obtain $\mathcal{R}^*$ by generating and adding to $\overline{\mathcal{R}}$ all of the routes with reduced cost greater than $LB_{MSP} - rVal$. Clearly, to be sure of not missing generating any such columns we need an exact pricing procedure.

Algorithm 5 SolveMSP

```
1:  $i = 0$ and $\mathcal{R}^1 = \emptyset$

    ***Column Generation Loop***
2:  repeat
3:      $i = i + 1$
4:      SOLVE(RelMast($\mathcal{R}^i$)). Let $w^i$ be the observations weight vector defined as in (2).
5:      for $q = 1, \ldots, k$ do
6:          SOLVE(LSPP($w^i, f^q$)). Let $r_i^q \notin \mathcal{R}^i$ be the returned route.
7:      end for
8:      $\mathcal{R}^{i+1} = \mathcal{R}^i \bigcup_{q=1,\ldots,k} \{r_i^q\}$.
9:  until some route with positive reduced cost is found
10: $\overline{\mathcal{R}} := \mathcal{R}^{i+1}$ and $\overline{w} := w^{i+1}$

    ***Integrality Gap Pricing***
11: $rVal := opt(\text{RelMast}(\overline{\mathcal{R}}))$
12: $LB_{MSP} := opt(\text{IntMast}(\overline{\mathcal{R}}))$
13: SOLVE(LSPP($\overline{w}, f^q$)) for all $q = 1, \ldots, k$
14: $\mathcal{R}^+ :=$ {all the routes with reduced cost greater than $LB_{MSP} - rVal$}.

    ***Integer Problem Solution***
15: SOLVE(IntMast($\overline{\mathcal{R}} \cup \mathcal{R}^+$)).
```

Remark. Here we want to stress that Algorithm **SolveMSP** returns a certified optimal solution to MSP if its implementation fulfills the following two requirements, that are critical from a computational standpoint:

i. At step 6 the pricing problem is solved exactly;

ii. At step 14, the set $\mathcal{R}^+$ contains all the routes with reduced cost greater than $LB_{MSP} - rVal$;

iii. The binary problem IntMast($\overline{\mathcal{R}} \cup \mathcal{R}^+$) is solved to optimality in step 17.

If one of these requirements is relaxed (for example the pricing problem is solved heuristically and/or the set $\mathcal{R}^+$ does not contain all the routes with a *promising* reduced cost, or the branch & bound algorithm for solving IntMast($\mathcal{R}^*$) problem is halted before proving the optimality of the current best solution) the solution provided by the **SolveMSP** algorithm is not necessarily optimal.

We implemented **SolveMSP** as described above: results are presented in Section 7. The exact algorithm can solve the MSP in a reasonable amount of time for small instances. Unfortunately, for the realistic size instances of our test-bed, we could not prove the optimality of the solutions provided, even if we can show they are not too far. Indeed, for these large instances, the following happens:

i. The exact combinatorial pricing algorithm EXC, quite effective for small instances, cannot tackle the pricing problem of the large ones, where the time-expanded network contains dozens of thousand nodes;

ii. The number of columns that are eligible for be part of the optimal solution gets too huge;

iii. The binary problem IntMast($\mathcal{R}^*$) is very large (and very dense) and cannot be solved to proven optimality in the time limit we set.

Therefore, for these large instances, we: i) solve the pricing problem with a heuristic procedure; ii) leave the **Column Generation Loop** as soon as a tailing-off behavior shows; iii) do not apply the **Integer Gap Pricing**: iv) set an adequate time limit for the solution of the final master problem.

Although these compromises may appear to heavily weaken the overall approach, the computational results that we present in Section 7) give evidence that the procedure we propose can still represent a useful tool for planning maritime surveillance, as it provides pretty good solutions to realistic size instances in a very reasonable amount of time.

7 Computational experiments

In this section we first describe our test-bed and give some implementation details. Then we discuss the computational results comparing different implementation choices for solving the pricing problem as well as the overall performance of our approach to the Maritime Surveillance Problem.

7.1 Instances description.

The grid. Our computational experiments are carried out on a set of instances of size and characteristics similar to real-life instances. All these instances are available upon request to the authors. We consider an area about 1/16 the size of Norwegian maritime territory, divided into 835 cells, as illustrated in Figure 3. The observation period associated with each cell is indicated in the picture.

The Force Elements. We considered a set of vessels of different types. Each type of vessel is characterized by its sensor range (representing the maximum distance of a cell that can be observed from the vessel's current position - two adjacent cells are at distance 1), maximum speed, starting cell, and cardinality (number of vessels of that type). Table 1 shows the different vessel types we consider and their features. Since the planning horizons considered in the experiments are shorter than the maximum endurance of the vessels, we let missions terminate anywhere in the grid; in contrast, the starting cell of any mission is given, and may coincide with the ending cell of the previous mission.

Table 1: Input data for the different vehicle types. Speed is given as the maximum number of cells that can be traversed in a time step and sensor range is given as the maximum distance of an observable cell. The starting point of each vessel type is indicated in Figure 3.

Vehicle type	Number of vehicles	Speed	Sensor range
1	2	2	1
2	1	1	1
3	2	1	1
4	1	1	2
5	2	1	2

In order to diversify the experiments, for a fixed planning horizon and based on the default values indicated in Table 1, we generate 8 distinct scenarios as follows. For each parameter column we consider two different settings depending on the status of a control switch, which are denoted by $\mathbf{CS^N}$ for the number of vehicles, $\mathbf{CS^S}$ for speed, and $\mathbf{CS^R}$ for sensor range respectively. When a specific switch is 0, the values of the corresponding parameter for each vehicle type are precisely those in the corresponding parameter column of Table 1. When the switch is 1, then all entries in the corresponding column are set to 1. So, for instance, when $\mathbf{CS^N} = 1$, we have exactly one vehicle for each type. Similarly, when $\mathbf{CS^R} = 1$, the sensor range will be 1 for all types. Because we have three parameters, we have precisely 8 combinations of parameter switch values (i.e. 8 scenarios).

Planning horizon and observations. We consider the following planning horizons: 6, 12, 24, 36, 48, 60, 72. For each cell i, we randomly generated an observation period $t_i \in T = \{1, 6, 12, 18, 24\}$ in such a way that the expected number of cells with observation period $t \in T$ is proportional to t. Therefore, for each cell with observation period 1, we have (on average) 24 cells with observation period 24. Such values are depicted in Figure 3. For all instances and all observations pI, we let $V_{pI} = 1$.

Hence, we consider 7 planning horizons and 8 scenarios: in total 56 instances indexed from 1 to 56.

Implementation details. We implemented the algorithms using C language and the optimization libraries of the commercial solver CPLEX (included in CPLEX Optimization studio 12.5) and we run them on a 3.5 GHz Intel(R) i7-4960X CPU machine with 6 dual threads cores, running 64 bit Linux Ubuntu 16.04 operating system.

Before delving into the discussion on our computational experiments, we give some implementation details on algorithm **SolveMSP** described in Section 6. At each iteration of the **Column Generation Loop**, the optimal solution to the current RelMast($\mathcal{R}^i$) is obtained by calling the CPLEX dual simplex algorithm, while the pricing problem is solved, depending on the case, by one of the algorithms DP, GDP, MDP, EXI, or EXC, defined in Section 4.

In the block named **Integrality Gap Pricing**, we invoke CPLEX MIP solver to solve to integral optimality the reduced problem IntMast($\overline{\mathcal{R}}$), defined over the set $\overline{\mathcal{R}}$ of routes generated in the previous block. The optimal value of this problem is denoted as LB_{MSP}. CPLEX MIP solver is finally invoked again to solve IntMast($\mathcal{R}^*$) in the final block **Integer Problem Solving** of the SolveMSP algorithm.

7.2 Assessing the column generation

In the first set of experiments we want to identify the best pricing heuristic among the three described in Section 4. In Table 2 we present the computational results obtained by solving the linear relaxation of the original master problem via the column generation scheme described in Section 6 (the Column Generation Loop of algorithm solveMSP) within a total time limit of 30 minutes. In the following we will denote by *CG-GDP*, *CG-DP* and *CG-MDP* the column generation (CG) algorithm where the pricing problem is solved by procedure *GDP* (Greedy Dynamic Programming), *DP* (Dynamic Programming), and *MDP* (Modified Dynamic Programming) resp.). Because of the time limit of 30 minutes and since all the three procedures GDP, DP, and MDP are not exact, the value of the optimal solution of RelMast($\overline{\mathcal{R}}$) (denoted by **rVal**) is in general a lower bound on the optimal value of RelMast($\mathcal{R}$).

The i-th row of the table is associated with instance $i = 1, \dots, 56$. The first four columns of the table describe the instance: **H** is the planning horizon, while the triple $\mathbf{CS^N}$, $\mathbf{CS^R}$, $\mathbf{CS^S}$ defines the control switches of the vessels. Then, there are three blocks of columns, associated with the three algorithms used to solve the pricing problem: GDP, DP, and MDP. In each of these blocks, **rVal** represents the optimal value of RelMast($\overline{\mathcal{R}}$), **rT** the total time requested by the algorithm, **cgN** the total number of routes generated, **cgI** the total number of iterations in the CG procedure, and **cgT** is the total time requested to solve the pricing problems. Times are expressed in seconds and 0.0 stands for any value less than 0.1.

The computational results of Table 2 show that all the three algorithms for the pricing problem are very fast and indeed, for all instances, most of the computation time is required by the iterative calls to the re-optimization procedure. The best performances in terms of generated lower bounds are the ones produced by the CG-MDP algorithm, which consistently dominates the other two algorithms. Nevertheless, the values provided by CG-DP and CG-GDP are not much worse, as illustrated in Figure 4.

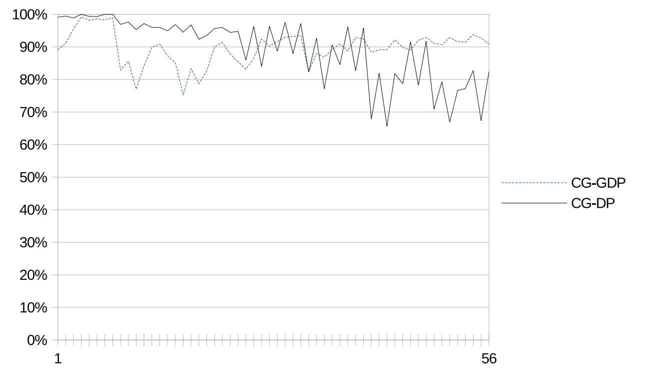
Here, for each instance $i \in 1, \dots, 56$ (in the x-axis), we fix the bound w_i computed by CG-MDP as reference value and reported in Table 2. Then we plot the ratios $\frac{y_i}{w_i}$ and $\frac{z_i}{w_i}$ of the bound y_i obtained by CG-DP and the bound z_i obtained by CG-GDP with the reference value w_i. The points are then joined by a line to better highlight the behavior.

For both ratios, the average is about 89%. In particular, CG-DP performs better than CG-GDP on the instances with a longer planning horizon.

Instance				CG-GDP					CG-DP					CG-MDP				
H	CS^N	CS^R	CS^S	rVal	rT	cgN	cgI	cgT	rVal	rT	cgN	cgI	cgT	rVal	rT	cgN	cgI	cgT
6	0	0	0	204.0	0.0	9	5	0.0	227.0	0.3	67	25	0.3	229.0	1.4	73	36	1.4
6	0	0	1	174.0	0.0	13	6	0.0	190.0	0.1	39	11	0.1	191.0	0.5	49	14	0.5
6	0	1	0	169.0	0.0	15	7	0.0	175.0	0.4	61	31	0.3	177.0	0.5	30	11	0.5
6	0	1	1	138.0	0.0	23	9	0.0	139.0	0.2	42	13	0.2	139.0	0.5	34	13	0.5
6	1	0	0	160.0	0.0	23	12	0.0	162.0	0.3	35	21	0.3	163.0	0.3	16	6	0.2
6	1	0	1	143.0	0.0	14	7	0.0	144.0	0.1	23	7	0.1	145.0	0.2	17	6	0.2
6	1	1	0	113.0	0.0	15	5	0.0	115.0	0.4	43	30	0.4	115.0	0.3	17	6	0.2
6	1	1	1	96.0	0.0	15	7	0.0	97.0	0.1	17	7	0.1	97.0	0.3	18	7	0.3
12	0	0	0	525.0	0.1	28	16	0.0	614.0	5.1	396	178	4.2	633.5	3.6	186	52	3.3
12	0	0	1	451.0	0.0	18	7	0.0	514.5	1.9	275	80	1.4	527.0	1.0	121	27	0.9
12	0	1	0	416.0	0.0	13	6	0.0	514.7	3.4	377	134	2.7	540.0	2.2	183	50	2.0
12	0	1	1	366.0	0.0	26	9	0.0	422.3	1.5	291	73	1.1	434.6	1.0	97	25	0.9
12	1	0	0	413.0	0.0	14	5	0.0	441.0	1.9	217	74	1.7	459.5	1.4	61	19	1.4
12	1	0	1	373.0	0.0	18	7	0.0	394.0	0.7	129	34	0.6	410.5	0.7	42	12	0.7
12	1	1	0	312.7	0.0	18	6	0.0	340.0	1.3	197	56	1.2	358.0	1.1	54	15	1.1
12	1	1	1	259.5	0.0	19	8	0.0	295.3	1.1	121	61	1.0	305.0	0.8	42	14	0.8
24	0	0	0	2647.8	0.3	27	8	0.0	3325.4	104.5	1358	526	51.6	3517.5	145.3	576	192	57.9
24	0	0	1	2367.9	0.5	39	13	0.0	2752.0	49.1	944	255	20.0	2843.4	31.1	346	78	12.8
24	0	1	0	2413.7	0.6	47	16	0.0	2833.6	106.0	1520	538	47.9	3068.4	106.5	469	133	38.7
24	0	1	1	2004.0	0.2	30	10	0.0	2268.7	50.8	1184	279	15.4	2425.8	49.1	488	126	18.1
24	1	0	0	2256.9	0.5	41	13	0.0	2397.9	25.7	635	147	15.5	2506.1	23.2	176	45	15.5
24	1	0	1	1978.5	0.2	29	10	0.0	2078.2	19.9	628	150	10.5	2166.1	23.6	209	60	12.9
24	1	1	0	1763.7	0.3	45	15	0.1	1897.0	13.2	602	147	10.1	2008.3	27.0	223	63	19.2
24	1	1	1	1436.9	0.4	53	17	0.0	1595.0	6.4	472	124	4.5	1683.3	16.7	201	51	9.2
36	0	0	0	9262.7	22.4	54	17	0.2	9578.0	463.0	1785	464	86.8	11 141.5	1833.3	555	111	103.8
36	0	0	1	8040.9	36.7	94	33	0.4	8942.3	708.1	2423	651	94.9	9284.5	1265.2	1145	258	121.7
36	0	1	0	8804.8	70.6	104	35	0.4	8005.5	419.2	2179	485	67.1	9529.3	1816.8	515	103	82.1
36	0	1	1	6910.3	26.5	71	20	0.2	7373.9	1273.5	2574	683	102.5	7658.7	1661.7	1320	301	123.7
36	1	0	0	7275.2	21.0	48	16	0.2	7037.6	500.1	1297	306	88.7	7937.2	991.3	759	192	200.6
36	1	0	1	6458.4	21.3	58	19	0.1	6775.9	348.7	1212	286	62.7	6947.9	750.8	688	169	127.9
36	1	1	0	5644.5	37.6	113	34	0.4	5327.5	409.1	1462	349	84.0	6054.7	525.8	959	229	173.7
36	1	1	1	4745.9	19.2	72	22	0.2	4931.0	151.4	1019	226	30.4	5074.0	168.2	615	146	46.1
48	0	0	0	15 803.6	152.2	92	32	1.3	15 768.0	1803.0	2193	481	175.1	19 169.5	1843.9	250	50	99.7
48	0	0	1	14 488.5	53.3	88	25	0.3	15 252.8	1816.0	2408	493	159.5	16 469.3	1832.6	250	50	92.7
48	0	1	0	14 162.2	57.7	87	27	0.4	12 568.0	1751.3	2914	625	198.9	16 302.2	1929.0	305	61	101.0
48	0	1	1	11 849.8	83.9	148	45	0.6	12 038.5	1821.5	2616	541	125.6	13 282.0	1829.3	310	62	83.7
48	1	0	0	12 470.8	43.3	77	24	0.3	11 606.0	1802.7	1580	356	209.9	13 718.5	1805.6	380	76	148.7
48	1	0	1	10 758.9	11.1	50	17	0.2	11 654.9	1110.6	1930	426	136.8	12 119.2	1816.6	385	77	140.3
48	1	1	0	9682.2	30.1	83	26	0.4	8624.1	1250.0	1783	440	203.3	10 429.9	1198.5	893	198	262.6
48	1	1	1	8121.5	13.5	58	17	0.2	8413.3	1185.4	2018	429	152.9	8781.9	1205.6	798	180	179.0
60	0	0	0	23 884.6	440.4	80	24	2.7	18 347.2	1812.4	1230	246	216.9	27 046.2	2047.1	150	30	90.1
60	0	0	1	21 111.7	737.9	107	35	3.7	19 404.9	1810.8	1908	403	210.0	23 683.4	1829.5	185	37	108.7
60	0	1	0	20 422.4	304.9	79	22	2.4	15 044.6	1803.1	1712	346	221.2	22 908.8	1856.6	160	32	90.5
60	0	1	1	17 502.8	476.0	107	26	3.5	15 530.8	1805.8	2644	533	205.8	18 978.9	1797.9	230	46	103.0
60	1	0	0	17 525.9	409.8	65	19	2.0	15 359.0	1807.4	1117	241	244.6	19 512.0	1829.8	275	55	145.1
60	1	0	1	15 415.0	71.9	52	14	0.9	15 837.7	1648.1	2105	475	282.2	17 310.0	1813.0	345	69	185.3
60	1	1	0	13 654.5	44.3	70	19	0.5	11 604.9	1800.5	2025	507	332.1	14 834.1	1806.2	390	78	199.7
60	1	1	1	11 642.1	42.8	96	27	0.6	11 512.7	1158.0	2298	529	197.6	12 540.2	1807.5	834	169	249.8
72	0	0	0	31 445.0	731.5	154	46	2.3	24 487.4	1800.4	1351	276	257.8	34 517.6	1990.0	95	19	99.4
72	0	0	1	27 701.9	569.3	142	48	2.0	24 220.5	1807.3	1877	397	250.5	30 539.6	1819.3	85	17	102.4
72	0	1	0	27 480.2	300.6	107	27	1.4	19 810.8	1802.5	2800	732	384.9	29 602.7	1877.8	155	31	105.5
72	0	1	1	22 674.1	355.2	166	49	2.0	18 945.9	1809.9	2593	578	244.3	24 739.0	1857.9	260	52	107.1
72	1	0	0	23 111.3	145.6	91	27	1.4	19 518.5	1804.6	1778	376	278.9	25 279.8	1824.3	185	37	186.3
72	1	0	1	21 134.8	293.5	138	42	1.9	18 648.1	1809.3	2128	461	290.5	22 535.5	1906.2	225	45	144.9
72	1	1	0	17 854.1	151.6	122	35	1.8	12 973.1	1801.4	2852	586	254.6	19 244.0	1812.6	315	63	266.9
72	1	1	1	14 814.3	37.4	62	15	0.6	13 428.3	747.3	2173	499	224.9	16 302.0	1835.4	375	75	190.7

Table 2: Comparing the three column generation approaches CG-GDP, CG-DP, and CG-MDP when solving the linear relaxation of the master problem within a time limit of 30 minutes.

Figure 4 Ratios between the bound obtained by CG-DP and CG-GDP with the bound obtained by CG-MDP, for each instance $1, \ldots, 56$. The instance index is represented on the x-axis.



The fact that MDP outperforms GDP and DP is also emphasized by the results reported in Table 3. In these experiments we first select, for each instance i, the best performing algorithm among CG-DP and CG-GDP, according to the returned lower bound **rVal(i)** of Table 2 (the corresponding data are reported in the columns labeled **CG-GDP / CG-DP**). Then, we solve the master relaxation of i with CG-MDP, stopping the procedure as soon as the lower bound **rVal(i)** is reached (or exceeded). The results of the procedure are presented in the columns labeled as **CG-MDP**. As one can see, with the exception of only three instances with $H = 6$, CG-MDP is definitively the fastest algorithm.

Next, in Table 4 we present the results obtained by the Column Generation Loop when the pricing problem is solved exactly. In particular, at each iteration of the Column Generation Loop, we first invoke our best heuristic approach MDP and then, in case MDP does not identify any route with positive reduced cost, we invoke the exact algorithm: namely, either the CPLEX MIP solver applied to the linear integer model (3) (EXI); or the exact combinatorial algorithm (EXC) presented in Section 4.

Because of the computational difficulty of the pricing problem, we can provide the optimal solution only for a subset of the 56 instances in our test-bed. In particular, we could solve to optimality all the instances with a 6-hour and 12-hour planning horizon. The values of such optimal solutions are presented in column **rVal/Exact**, while column **rVal/Heu** reports the values provided by the CG-MDP procedure. For each of the two exact methods we implemented, the ILP model and EXC, columns **tT**, $\mathbf{cgN_H}$, $\mathbf{cgN_E}$, **cgI** present the total time requested, the number of heuristic routes generated, the number of exact routes generated, and number of iterations in the column generation loop respectively. The symbol (-) is used to identify the instances that could not be solved by the CPLEX algorithm within a time limit of one hour for a single call.

We stress that, as the pricing problem is solved to optimality, the values reported in the column **rVal/Exact** are the optimal values of the linear relaxation RelMast($\mathcal{R}$) of the overall master problem and therefore valid upper bounds for the optimal value of the MSP.

Instance				CG-DP/CG-GDP						CG-MDP				
H	**CS^{N}**	**CS^{R}**	**CS^{S}**	**rVal**	**rT**	**cgN**	**cgI**	**cgT**	**Algo**	**rVal**	**rT**	**cgN**	**cgI**	**cgT**
6	0	0	0	227.0	0.3	67	25	0.3	$CG - DP$	228.0	0.0	35	8	0.2
6	0	0	1	190.0	0.1	39	11	0.1	$CG - DP$	190.0	0.2	42	9	0.2
6	0	1	0	175.0	0.4	61	31	0.3	$CG - DP$	176.0	0.1	19	4	0.1
6	0	1	1	139.0	0.2	42	13	0.2	$CG - DP$	139.0	0.3	34	13	0.3
6	1	0	0	162.0	0.3	35	21	0.3	$CG - DP$	163.0	0.1	13	3	0.1
6	1	0	1	144.0	0.1	23	7	0.1	$CG - DP$	145.0	0.1	14	3	0.1
6	1	1	0	115.0	0.4	43	30	0.4	$CG - DP$	115.0	0.1	17	6	0.1
6	1	1	1	97.0	0.1	17	7	0.1	$CG - DP$	97.0	0.2	18	7	0.2
12	0	0	0	614.0	5.1	396	178	4.2	$CG - DP$	616.0	0.3	35	7	0.2
12	0	0	1	514.5	1.9	275	80	1.4	$CG - DP$	516.6	0.3	40	8	0.3
12	0	1	0	514.7	3.4	377	134	2.7	$CG - DP$	524.0	0.2	19	4	0.2
12	0	1	1	422.3	1.5	291	73	1.1	$CG - DP$	426.0	0.2	30	6	0.2
12	1	0	0	441.0	1.9	217	74	1.7	$CG - DP$	456.0	0.2	20	4	0.2
12	1	0	1	394.0	0.7	129	34	0.6	$CG - DP$	397.0	0.1	15	3	0.1
12	1	1	0	340.0	1.3	197	56	1.2	$CG - DP$	341.0	0.1	9	2	0.1
12	1	1	1	295.3	1.1	121	61	1.0	$CG - DP$	297.0	0.1	14	3	0.1
24	0	0	0	3325.4	104.5	1358	526	51.6	$CG - DP$	3349.8	0.9	20	4	0.7
24	0	0	1	2752.0	49.1	944	255	20.0	$CG - DP$	2762.4	1.6	40	8	0.9
24	0	1	0	2833.6	106.0	1520	538	47.9	$CG - DP$	2844.0	0.6	15	3	0.4
24	0	1	1	2268.7	50.8	1184	279	15.4	$CG - DP$	2275.2	0.6	20	4	0.4
24	1	0	0	2397.9	25.7	635	147	15.5	$CG - DP$	2421.0	0.7	15	3	0.5
24	1	0	1	2078.2	19.9	628	150	10.5	$CG - DP$	2089.5	0.7	20	4	0.5
24	1	1	0	1897.0	13.2	602	147	10.1	$CG - DP$	1933.5	0.6	15	3	0.5
24	1	1	1	1595.0	6.4	472	124	4.5	$CG - DP$	1608.0	0.6	20	4	0.4
36	0	0	0	9578.0	463.0	1785	464	86.8	$CG - DP$	9712.9	2.8	15	3	1.9
36	0	0	1	8942.3	708.1	2423	651	94.9	$CG - DP$	8947.5	9.7	40	8	3.0
36	0	1	0	8804.8	70.6	104	35	0.4	$CG - GDP$	8842.8	5.0	20	4	2.1
36	0	1	1	7373.9	1273.5	2574	683	102.5	$CG - DP$	7387.8	8.2	40	8	2.3
36	1	0	0	7275.2	21.0	48	16	0.2	$CG - GDP$	7298.0	3.2	15	3	1.8
36	1	0	1	6775.9	348.7	1212	286	62.7	$CG - DP$	6802.2	8.8	40	8	3.1
36	1	1	0	5644.5	37.6	113	34	0.4	$CG - GDP$	5736.5	1.2	10	2	1.0
36	1	1	1	4931.0	151.4	1019	226	30.4	$CG - DP$	4969.8	3.6	30	6	1.7
48	0	0	0	15 803.6	152.2	92	32	1.3	$CG - GDP$	16 591.7	7.1	15	3	4.0
48	0	0	1	15 252.8	1816.0	2408	493	159.5	$CG - DP$	15 391.2	15.5	25	5	4.7
48	0	1	0	14 162.2	57.7	87	27	0.4	$CG - GDP$	14 758.1	20.4	20	4	4.5
48	0	1	1	12 038.5	1821.5	2616	541	125.6	$CG - DP$	12 166.9	8.2	20	4	2.6
48	1	0	0	12 470.8	43.3	77	24	0.3	$CG - GDP$	12 534.7	9.4	15	3	4.2
48	1	0	1	11 654.9	1110.6	1930	426	136.8	$CG - DP$	11 776.1	17.4	30	6	5.4
48	1	1	0	9682.2	30.1	83	26	0.4	$CG - GDP$	9904.7	5.0	15	3	3.5
48	1	1	1	8413.3	1185.4	2018	429	152.9	$CG - DP$	8434.0	6.6	20	4	2.7
60	0	0	0	23 884.6	440.4	80	24	2.7	$CG - GDP$	24 531.0	37.2	20	4	9.7
60	0	0	1	21 111.7	737.9	107	35	3.7	$CG - GDP$	21 679.9	24.7	20	4	6.8
60	0	1	0	20 422.4	304.9	79	22	2.4	$CG - GDP$	20 886.0	29.5	20	4	7.9
60	0	1	1	17 502.8	476.0	107	26	3.5	$CG - GDP$	17 873.9	22.3	25	5	6.1
60	1	0	0	17 525.9	409.8	65	19	2.0	$CG - GDP$	17 812.4	13.2	15	3	7.5
60	1	0	1	15 837.7	1648.1	2105	475	282.2	$CG - DP$	16 262.7	23.3	20	4	7.0
60	1	1	0	13 654.5	44.3	70	19	0.5	$CG - GDP$	14 058.0	9.2	15	3	6.0
60	1	1	1	11 642.1	42.8	96	27	0.6	$CG - GDP$	11 816.5	6.0	15	3	3.8
72	0	0	0	31 445.0	731.5	154	46	2.3	$CG - GDP$	31 639.6	80.6	20	4	15.0
72	0	0	1	27 701.9	569.3	142	48	2.0	$CG - GDP$	28 398.2	67.7	20	4	11.3
72	0	1	0	27 480.2	300.6	107	27	1.4	$CG - GDP$	27 662.7	48.2	25	5	15.2
72	0	1	1	22 674.1	355.2	166	49	2.0	$CG - GDP$	22 729.1	29.1	25	5	9.8
72	1	0	0	23 111.3	145.6	91	27	1.4	$CG - GDP$	23 237.2	31.7	15	3	11.7
72	1	0	1	21 134.8	293.5	138	42	1.9	$CG - GDP$	21 135.1	32	20	4	11.4
72	1	1	0	17 854.1	151.6	122	35	1.8	$CG - GDP$	18 026.2	17.7	15	3	9.7
72	1	1	1	14 814.3	37.4	62	15	0.6	$CG - GDP$	15 298.9	16.7	20	4	8.2

Table 3: Comparing computational effort and results of the best performing algorithm between CG-GDP and CG-DP against CG-MDP, when returning (almost) the same lower bound value.

Remarkably, such values coincide almost always with the ones provided by the heuristic algorithm CG-MDP, and the largest gap is less than 1%.

Moreover, observe that the EXC pricing algorithm significantly outperforms the MILP (+ CPLEX) approach and it is very effective for all the 6-hours planning horizon instances and the 12-hours ones with no ships with speed 2. Indeed, in the latter case, the number of possible routes[3] grows to 7^{24} and the enumeration tree (even if fathomed by bounding the sub-paths) grows too large.

Instance				**rVal**		**EXI**				**EXC**			
H	**CS^N**	**CS^R**	**CS^S**	**Heu**	**Exact**	**tT**	**cgN_H**	**cgN_E**	**cgI**	**tT**	**cgN_H**	**cgN_E**	**cgI**
6	0	0	0	229.0	229.0	1440.0	73	0	36	1.0	73	0	36
6	0	0	1	191.0	191.0	7.1	49	0	14	0.6	49	0	14
6	0	1	0	177.0	177.0	1293.4	30	0	11	0.6	30	0	11
6	0	1	1	139.0	139.0	32.3	34	0	13	0.5	34	0	13
6	1	0	0	163.0	163.0	1823.7	16	0	6	0.6	16	0	6
6	1	0	1	145.0	145.0	17.6	17	1	7	0.9	17	1	7
6	1	1	0	115.0	115.0	1932.0	17	0	6	0.6	17	0	6
6	1	1	1	97.0	97.0	6.5	18	0	7	0.4	18	0	7
12	0	0	1	527.0	528.0	96736.1	159	80	101	8.0	124	7	32
12	0	1	1	434.6	435.7	481147.9	153	241	168	19.2	108	22	48
12	1	0	1	410.5	411.0	361806.4	66	224	173	6.8	45	4	18
12	1	1	1	305.0	305.0	92795.6	54	76	65	6.9	46	9	21
12	0	0	0	633.5	633.5	-	-	-	-	10733.0	200	18	71
12	0	1	0	540.0	545.0	-	-	-	-	17828.0	308	58	128
12	1	0	0	459.5	463.0	-	-	-	-	3232.2	65	7	27
12	1	1	0	358.0	359.0	-	-	-	-	4482.6	66	17	29

Table 4: Solving the Master Linear Relaxation to optimality.

In principle, in column generation schemes the process terminates when no columns with strictly positive reduced cost exist. However, it is well known that the approach suffers the so called *tailing off*, that is the last generated columns tend to give little or no contribution to improve the bound. One may wonder whether putting a cap on the time spent in generating columns (without waiting for the natural termination) would have significant impact on the quality of the final solution. Another natural question regards the number of columns to generate before solving the new master problem relaxation. In our algorithm, we return at most one column (for each vehicle type). In some cases we may benefit from trying to identify and generate multiple columns.

We try to answer these questions with the next experiments. Table 5 reports the computational results obtained by the CG-MDP algorithm when adding one column (CG-MDP-300s-1c) or five columns (CG-MDP-300s-5c) for each vessel type at each iteration of the Column Generation Loop. The time limit is set to five minutes for all experiments. The figures are summarized in Figure 5 where the lower bounds obtained by CG-MDP-300s-1c and CG-MDP-300s-5c are compared with the bounds of the original CG-MDP algorithm (with 30 minutes time limit), reported in Table 2. As previously, in Figure 5 we use the bound computed by CG-MDP as reference value, and we plot, for each instance, the ratios of the bound obtained by CG-MDP-300s-1c and CG-MDP-300s-5c with this reference value. One can immediately notice that CG-MDP-300s-1c performs slightly better than CG-MDP-300s-5c. Adding more columns for each vessel type at any iteration of the Column Generation Loop does not seem to have any positive impact.

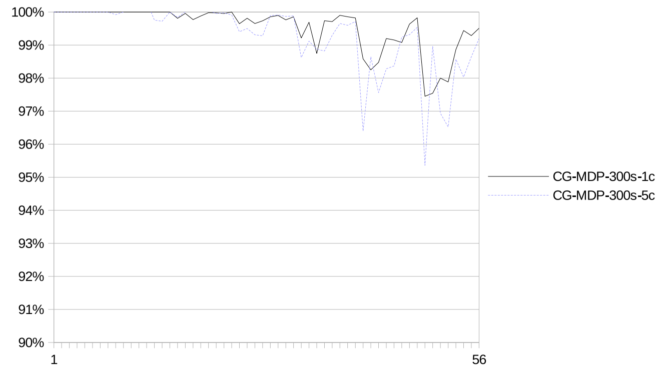
[3]In each cell vessels can make 7 different choices, either moving to one of the 6 neighbors or staying still

Instance				CG-MDP-300s-1c	CG-MDP-300s-5c
H	CS^{N}	CS^{R}	CS^{S}	rVal	rVal
6	0	0	0	229.0	229.0
6	0	0	1	191.0	191.0
6	0	1	0	177.0	177.0
6	0	1	1	139.0	139.0
6	1	0	0	163.0	163.0
6	1	0	1	145.0	145.0
6	1	1	0	115.0	115.0
6	1	1	1	97.0	97.0
12	0	0	0	633.5	633
12	0	0	1	527.0	527.0
12	0	1	0	540.0	540.4
12	0	1	1	434.6	434.9
12	1	0	0	459.5	461.0
12	1	0	1	410.5	409.5
12	1	1	0	358.0	357.0
12	1	1	1	305.0	305.0
24	0	0	0	3510.8	3511.8
24	0	0	1	2842.2	2843.6
24	0	1	0	3061.3	3069.0
24	0	1	1	2422.9	2427.4
24	1	0	0	2505.6	2507.2
24	1	0	1	2165.6	2165.2
24	1	1	0	2007.4	2007.8
24	1	1	1	1683.3	1681.8
36	0	0	0	11 102.1	11 075.1
36	0	0	1	9267.1	9238.2
36	0	1	0	9496.1	9463.6
36	0	1	1	7638.3	7603.7
36	1	0	0	7925.7	7928.5
36	1	0	1	6940.6	6941.8
36	1	1	0	6040.4	6046.7
36	1	1	1	5066.7	5068.1
48	0	0	0	19 019.4	18 905.6
48	0	0	1	16 418.3	16 324.6
48	0	1	0	16 098.4	16 118.3
48	0	1	1	13 246.9	13 125.5
48	1	0	0	13 678.8	13 621.7
48	1	0	1	12 107.1	12 077.0
48	1	1	0	10 415.1	10 388.0
48	1	1	1	8766.2	8756.6
60	0	0	0	26 662.7	26 071.8
60	0	0	1	23 268.8	23 362.8
60	0	1	0	22 559.6	22 350.2
60	0	1	1	18 826.7	18 652.5
60	1	0	0	19 347.0	19 193.5
60	1	0	1	17 150.3	17 178.5
60	1	1	0	14 779.9	14 732.5
60	1	1	1	12 518.2	12 482.0
72	0	0	0	33 638.0	32 915.8
72	0	0	1	29 788.9	30 223.0
72	0	1	0	29 010.7	28 695.9
72	0	1	1	24 215.5	23 878.9
72	1	0	0	24 992.8	24 919.1
72	1	0	1	22 409.3	22 091.6
72	1	1	0	19 106.6	18 983.3
72	1	1	1	16 222.0	16 171.3

Table 5: Bounds obtained by the CG-MDP algorithm within a time limit of 5 minutes. At each iteration of Column Generation Loop and for each vessel type, we add up to one (**CG-MDP-300s-1c**) or five (**CG-MDP-300s-5c**) new routes.

Moreover, the average ratio of the lower bound obtained by CG-MDP-300s-1c is always greater than 97.5%. Hence, reducing the time limit to 5 minutes (from 30 minutes) does not deteriorate significantly the solution quality.

Figure 5 Ratios between the bound obtained by CG-MDP-300-1c and CG-MDP-300-5c with the bound obtained by CG-MDP, for each instance $1, \ldots, 56$.



7.3 Solving the Maritime Surveillance Problem

With the next experiments we test the ability of the overall approach to generate good and possibly optimal solutions to the Maritime Surveillance Problem.

We first focus on the exact version of Algorithm SolveMSP, where the pricing problem is solved by the purely combinatorial approach EXC. We allow a one-hour time limit for the Column Generation Loop block. Within this time limit we were able to solve RelMast($\mathcal{R}$) to optimality for all 6-hour planning horizon instances and for a subset of the 12-hour and 15-hour planning horizon instances. Therefore, our experiments with the exact version of Algorithm SolveMSP will be limited to this set of instances. The corresponding results are shown in Table 6.

Instance				**SolveMSP**						
H	**CS^N**	**CS^R**	**CS^S**	**rVal**	**OPT$_{MSP}$**	**LB$_{MSP}$**	**igN**	**cgT**	**igT**	**intT**
6	0	0	0	229.0	229.0	229.0	0	1.0	0.5	0.0
6	0	0	1	191.0	191.0	191.0	0	0.5	0.4	0.0
6	0	1	0	177.0	177.0	177.0	0	0.6	0.5	0.0
6	0	1	1	139.0	139.0	139.0	0	0.4	0.3	0.0
6	1	0	0	163.0	163.0	163.0	0	0.5	0.5	0.0
6	1	0	1	145.0	145.0	145.0	0	0.7	0.3	0.0
6	1	1	0	115.0	115.0	115.0	0	0.5	0.4	0.0
6	1	1	1	97.0	97.0	97.0	0	0.4	0.3	0.0
12	0	0	1	528.0	528.0	528.0	0	7.1	1.6	0.6
12	0	1	1	435.7	435.0	435.0	0	18.4	6.2	2.6
12	1	0	1	411.0	411.0	411.0	0	6.8	1.4	0.0
12	1	1	1	305.0	305.0	305.0	0	6.5	1.3	0.0
12	1	0	0	463.0	463.0	463.0	0	3228.6	386.3	0.1
15	0	0	1	869.3	868.0	868.0	2665	94.0	19.1	17.7
15	0	1	1	724.9	-	718.0	$>$ 100000	85.5	-	-
15	1	0	1	671.5	671.0	670.0	376	40.4	7.2	8.4
15	1	1	1	506.0	506.0	506.0	0	29.7	2.6	0.3

Table 6: Solving MSP to optimality

As in the previous tables, in column **rVal** we report the optimal values of RelMast($\overline{\mathcal{R}}$)

(since the pricing problem is solved to optimality, we have RelMast($\overline{\mathcal{R}}$ = RelMast($\mathcal{R}$)). Column $\mathbf{OPT_{MSP}}$ indicates the optimal value of IntMast($\mathcal{R}$) (i.e. the optimal value of MSP), while $\mathbf{LB_{MSP}}$ is the lower bound on $\mathbf{OPT_{MSP}}$, used in the Integrality Gap Pricing block of Algorithm 5 to produce the **igN** routes of set $\mathcal{R}^+$. Further, columns **cgT**, **igT**, and **intT** are the running times for the Column Generation Loop, the Integrality Gap Pricing, and the Integer Problem Solution respectively.

The results in the table show that for all instances with planning horizon up to 12 hours, the optimal solution to MSP can be found by using only the routes in $\overline{\mathcal{R}}$ generated in the Column Generation Loop. Indeed, in all these cases, $LB_{MSP} = \lfloor rVal \rfloor$ and, since the values V_{pI} are all integer, there is no need to add routes in the Integer Gap Pricing step. The same occurs also for the 15-hour instance in the last row of the table. For the remaining three 15-hour instances, the integrality gap $rVal - LB_{MSP}$ is greater than 1 and, consequently, the exact pricing algorithm has to be called in order to construct the set $\mathcal{R}^+$ of routes, needed to guarantee that the optimal solution to IntMast($\overline{\mathcal{R}} \cup \mathcal{R}^+$) is an optimal solution to the overall problem IntMast($\mathcal{R}$). For the two cases with $CS^R = 0$, $CS^S = 1$, and $CS^N \in \{0, 1\}$ (where such a gap is less than 1), the number of routes of $\mathcal{R}^+$ stays reasonably small and the final ILP problem IntMast($\overline{\mathcal{R}} \cup \mathcal{R}^+$) can be solved by CPLEX in a few seconds. On the other hand, for the case with $CS^R = 1$, the integrality gap is higher (i.e. $= 6.9$) and the Integrality Gap Pricing block of the algorithm rapidly identifies more than 100000 routes in $\mathcal{R}^+$. Therefore, the size of IntMast($\mathcal{R}^*$) is so large that the corresponding ILP program cannot be tackled by CPLEX.

We now evaluate the performance of the heuristic variant of Algorithm SolveMSP. Indeed, our heuristic algorithms for the pricing problem behave rather well in practice. In particular, whenever it is possible to check, we verified that CG-MDP returns almost always the optimal columns, i.e. those with maximum reduced costs. Consequently, for most of these instances, at termination of the Column Generation Loop, all columns in $\mathcal{R}$ have non-positive reduced costs. Recall that, when this is the case, the current restricted master program RelMast($\overline{\mathcal{R}}$) actually suffices to solve the linear relaxation RelMast($\mathcal{R}$) of the (complete) master program (1), and its value is an upper bound on the optimal (integer) solution value of the MSP (i.e. IntMast($\mathcal{R}$)).

The next Table 7 and Figure 6 describe the computational results obtained by the following heuristic version of Algorithm SolveMSP, where the Integrality Gap Pricing block is skipped:

1. *Greedy solution.* Define an initial solution SS to the MSP problem by iteratively applying, in a greedy fashion, algorithm MDP so to find a route for every available vessel. The value of this solution is reported in column **SS val** of the table;

2. *Column Generation Loop*: apply the CG-MDP algorithm with 300-second time limit;

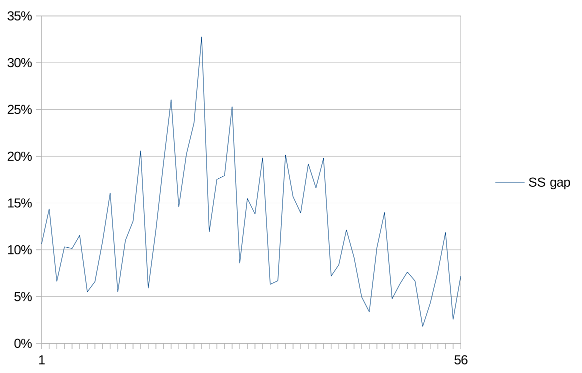
3. *Integer Problem Solution*: solve to integer optimality the reduced master problem IntMast($\overline{\mathcal{R}}$) by the branch & bound algorithm of CPLEX MIP solver, with 1-hour time limit. The initial solution SS provides the first incumbent. We also set CPX_PARAM_PROB = 3 and we gave branching priority 100 to variables y and branching priority 0 to variables x.

In the table, column **BS val** reports the values of the best solutions to IntMast($\overline{\mathcal{R}}$) found within the time limit, while, in column **SS Gap**, we report the percentage increment of **BS val** with respect to **SS val**. Such a value, depicted also in Figure 6, is more than 12% on average and, in a few cases, even larger than 25%. This somehow gives evidence of the difficulty of the Maritime Surveillance Problem and confirms the idea that a more sophisticated approach, like the one we propose here, is needed in order to obtain solutions of guaranteed quality. The results of Table 7 also show that such solutions can be obtained within a time limit of 65 minutes for instances of size up to 72-hours planning horizon.

Instance				**CG-MDP + B&B**		
H	**CS^{N}**	**CS^{R}**	**CS^{S}**	**SS val**	**BS val**	**SS Gap (%)**
6	0	0	0	207.0	229.0	10.6
6	0	0	1	167.0	191.0	14.4
6	0	1	0	166.0	177.0	6.6
6	0	1	1	126.0	139.0	10.3
6	1	0	0	148.0	163.0	10.1
6	1	0	1	130.0	145.0	11.5
6	1	1	0	109.0	115.0	5.5
6	1	1	1	91.0	97.0	6.6
12	0	0	0	570.0	632.0	10.9
12	0	0	1	454.0	527.0	16.1
12	0	1	0	508.0	536.0	5.5
12	0	1	1	391.0	434.0	11.0
12	1	0	0	406.0	459.0	13.1
12	1	0	1	340.0	410.0	20.6
12	1	1	0	338.0	358.0	5.9
12	1	1	1	272.0	305.0	12.1
24	0	0	0	2883.0	3439.0	19.3
24	0	0	1	2216.0	2793.0	26.0
24	0	1	0	2612.0	2993.0	14.6
24	0	1	1	1975.0	2374.0	20.2
24	1	0	0	2012.0	2486.0	23.6
24	1	0	1	1612.0	2140.0	32.8
24	1	1	0	1760.0	1970.0	11.9
24	1	1	1	1404.0	1650.0	17.5
36	0	0	0	8884.0	10 477.0	17.9
36	0	0	1	7160.0	8971.0	25.3
36	0	1	0	8150.0	8849.0	8.6
36	0	1	1	6313.0	7290.0	15.5
36	1	0	0	6844.0	7791.0	13.8
36	1	0	1	5671.0	6796.0	19.8
36	1	1	0	5628.0	5983.0	6.3
36	1	1	1	4649.0	4960.0	6.7
48	0	0	0	14 395.0	17 294.0	20.1
48	0	0	1	13 295.0	15 379.0	15.7
48	0	1	0	12 874.0	14 669.0	13.9
48	0	1	1	10 469.0	12 477.0	19.2
48	1	0	0	11 298.0	13 176.0	16.6
48	1	0	1	9781.0	11 716.0	19.8
48	1	1	0	9494.0	10 177.0	7.2
48	1	1	1	7870.0	8531.0	8.4
60	0	0	0	21 617.0	24 239.0	12.1
60	0	0	1	19 629.0	21 430.0	9.2
60	0	1	0	19 096.0	20 043.0	5.0
60	0	1	1	16 754.0	17 318.0	3.4
60	1	0	0	16 672.0	18 377.0	10.2
60	1	0	1	14 453.0	16 474.0	14.0
60	1	1	0	13 570.0	14 217.0	4.8
60	1	1	1	11 399.0	12 120.0	6.3
72	0	0	0	28 593.0	30 774.0	7.6
72	0	0	1	25 677.0	27 391.0	6.7
72	0	1	0	25 592.0	26 054.0	1.8
72	0	1	1	21 500.0	22 424.0	4.3
72	1	0	0	21 669.0	23 332.0	7.7
72	1	0	1	19 057.0	21 314.0	11.8
72	1	1	0	17 722.0	18 179.0	2.6
72	1	1	1	14 668.0	15 725.0	7.2

Table 7: The table presents the values of the solutions for the MSP obtained by applying the CG-MDP algorithm (with 300 seconds time limit) and then solving the obtained restricted Master Problem to integral optimality (within a time limit of 1 hour).

Figure 6 The picture reports the behavior of **SS Gap** of Table 7.



8 Conclusions.

In cooperation with the Norwegian Defence Research Establishment, we tackled a crucial problem when planning maritime surveillance activities. In particular, in this paper we

- Introduce a new periodic vehicle routing problem, which is very relevant for public bodies and authorities involved in periodic surveillance activities;
- Develop a novel binary LP model for periodic vehicle routing, based on the new concept of ”interval formulation“, which could also be applied in other application contexts;
- Present a column generation approach to the problem;
- Show that the pricing problem is NP-hard;
- Develop exact and heuristic column generation pricing algorithms;
- Carry out extensive tests showing that medium to large sized realistic instances – which are indeed very large integer programs - can be tackled effectively by our approach.

Interesting future research directions and topics, include the use of heterogeneous fleets involving vehicles with very different features. For instance, one may consider aircraft, which have short endurance but high speed and wide sensor range. More in general, it would be interesting to extend the approach to long planning horizons. Clearly, the longer the horizon, the larger the instance to solve and this would make the problem even harder to tackle in practice. Still, the classic *rolling horizon* technique (e.g. see Pinedo (2016)) seems a natural decomposition approach for this kind of problem, certainly worth investigated.

References

D. Alvras and M. Padberg, *Linear Optimization and Extensions: Problems and Solutions.* Berlin, Germany: Springer-Verlag, 2001.

P. Avella, M. Boccia, C. Mannino, and I. Vasilyev, “Time-indexed formulations for the runway scheduling problem,” *Transportation Science*, 2017.

R. Baldacci, A. Mingozzi, and R. Roberti, “New route relaxation and pricing strategies for the vehicle routing problem,” *Operations Research*, vol. 59, pp. 1269–1283, 2011.

D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization.* Belmont, MA: Athena Scientific, 1997, vol. 6.

V. Cacchiani, V. Hemmelmayr, and F. Tricoire, “A set-covering based heuristic algorithm for the periodic vehicle routing problem,” *Discrete Applied Mathematics*, vol. 163, pp. 53–64, 2014.

A. Caprara, M. Fischetti, and P. Toth, “Modeling and solving the train timetabling problem,” *Operations research*, vol. 50, no. 5, pp. 851–861, 2002.

N. Christofides, A. Mingozzi, and P. Toth, “Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations,” *Mathematical Programming*, vol. 20, pp. 255–282, 1981.

S. Dash, O. Günlük, A. Lodi, and A. Tramontani, “A time bucket formulation for the traveling salesman problem with time windows,” *INFORMS Journal on Computing*, vol. 24, no. 1, pp. 132–147, 2012.

G. Desaulniers, J. Desrosiers, and M. Solomon, Eds., *Column Generation.* New York: Springer Science & Business Media, 2006, vol. 5.

M. Drexl and S. Irnich, “Solving elementary shortest-path problems as mixed-integer programs,” *OR Spectrum*, vol. 36, pp. 281–296, 2014.

O. Dridi, S. Krichen, and A. Guitouni, “A multi-objective optimization approach for resource assignment and task scheduling problem: Application to maritime domain awareness,” in *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1–8.

M. E. Dyer and L. A. Wolsey, “Formulating the single machine sequencing problem with release dates as a mixed integer program,” *Discrete Applied Mathematics*, vol. 26, no. 2-3, pp. 255–270, 1990.

R. Fukasawa, H. Longo, J. Lysgaard, M. P. d. Aragão, M. Reis, E. Uchoa, and R. F. Werneck, “Robust branch-and-cut-and-price for the capacitated vehicle routing problem,” *Mathematical Programming*, vol. 106, no. 3, pp. 491–511, 2006.

M. J. H. B. Grob, “Routing of platforms in a maritime surface surveillance operation,” *European Journal of Operational Research*, vol. 170, pp. 613–628, 2006.

S. Harrod, “Modeling network transition constraints with hypergraphs,” *Transportation Science*, vol. 45, no. 1, pp. 81–97, 2011.

D. S. Hochbaum, Ed., *Approximation Algorithms for NP-hard Problems.* Boston, MA, USA: PWS Publishing Co., 1997.

K. Ilavarasi and K. S. Joseph, “Variants of travelling salesman problem: a survey,” in *Information Communication and Embedded Systems (ICICES), 2014 International Conference on.* IEEE, 2014, pp. 1–7.

D. Kjenstad, C. Mannino, P. Schittekat, and M. Smedsrud, “Integrated surface and departure management at airports by optimization,” in *Modeling, Simulation and Applied Optimization (ICMSAO), 2013 5th International Conference on.* IEEE, 2013, pp. 1–5.

D. Marlow, P. Kilby, and G. Mercer, “The travelling salesman problem in maritime surveillance-techniques, algorithms and analysis,” in *International Congress on Modelling and Simulation*, 2007, pp. 684–690.

R. Martinelli, D. Pecin, and M. Poggi, “Efficient elementary and restricted non-elementary route pricing,” *European Journal of Operational Research*, vol. 239, no. 1, pp. 102–111, 2014.

A. Mingozzi, “The multi-depot periodic vehicle routing problem,” *Lecture notes in Computer Science*, vol. 3607, pp. 347–350, 2005.

M. L. Pinedo, *Scheduling: theory, algorithms, and systems.* Springer, 2016.

S. Pirkwieser and G. Raidl, “A column generation approach for the periodic vehicle routing problem with time windows,” https://www.researchgate.net/publication/, 2012, Accessed 3 June 2016.

M. Queyranne and A. S. Schulz, “Polyhedral approaches to machine scheduling,” Technische Universitat Berlin, Tech. Rep. 480/1994, 1994.

N. Quttineh, T. Larsson, J. Van den Bergh, and J. Beliën, “A Time-Indexed Generalized Vehicle Routing Model and Stabilized Column Generation for Military Aircraft Mission Planning,” in *Optimization, Control, and Applications in the Information Age.* Switzerland: Springer International Publishing, 2015, pp. 299–314.

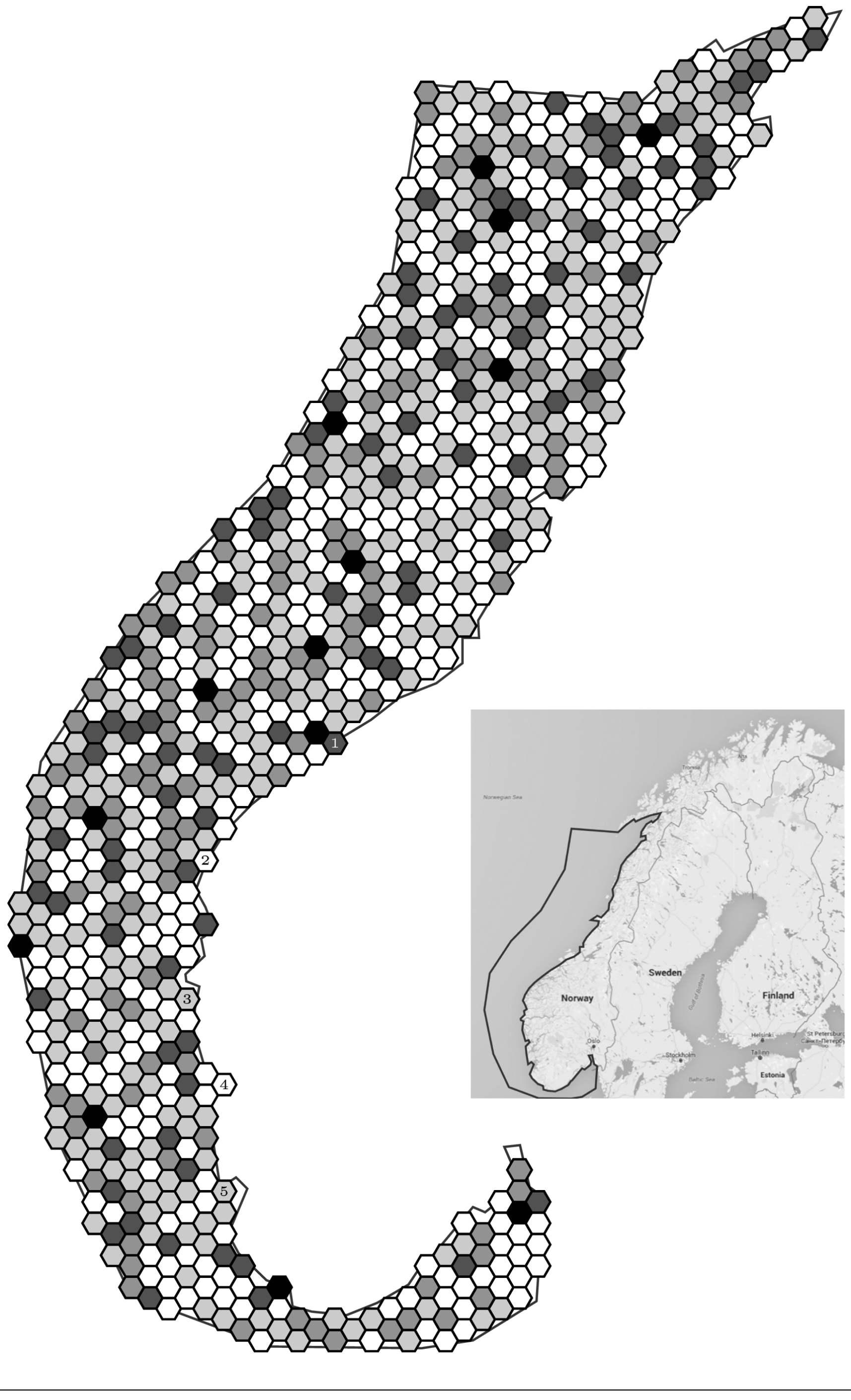
L. Schrijver, *Combinatorial Optimization.* Berlin Heidelberg: Springer-Verlag, 2003.

E. Stumpt and N. Michael, “Multi-robot persistent surveillance planning as a vehicle routing problem,” in *IEEE International Conference on Automation Science and Engineering*, Trieste, Italy, 24–27 August 2011.

D. Vatne and H. Gisnås, “MOBY – a simulation tool for evaluating maritime surveillance,” 2014, Unpublished manuscript. Available at the Norwegian Defence Research Establishment at request.

L. A. Wolsey, “Integer programming. series in discrete mathematics and optimization,” 1998.

Figure 3 The picture shows the grid we considered for the computational experiments. The colors of the hexagonal cells represent the corresponding observation periods: white is for 24 hours, light-gray is 18 hours, gray is 12 hours, dark-gray is 6 hours and black is 1 hour. We also indicate here (with indices in the corresponding cells) the route starting point for each vessel type of Table 1.


1
2
3
4
5
Norwegian Sea
Norway
Sweden
Finland
Oslo
Stockholm
Helsinki
Tallinn
Estonia
St Petersburg
Baltic Sea