Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Generalized Periodic Vehicle Routing and Maritime Surveillance

Maria Fleischer Fauske, Carlo Mannino, Paolo Ventura

To cite this article:
Maria Fleischer Fauske, Carlo Mannino, Paolo Ventura (2020) Generalized Periodic Vehicle Routing and Maritime Surveillance. Transportation Science 54(1):164-183. https://doi.org/10.1287/trsc.2019.0899

http://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Vol. 54, No. 1, January–February 2020, pp. 164–183
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Generalized Periodic Vehicle Routing and Maritime Surveillance

Maria Fleischer Fauske,[a] Carlo Mannino,[b,c] Paolo Ventura[d]

[a] Norwegian Defence Research Establishment, NO-2027 Kjeller, Norway; [b] SINTEF ICT Applied Mathematics, NO-0314 Oslo, Norway; [c] University of Oslo, NO-0316 Oslo, Norway; [d] Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti" (IASI) del CNR, 00185 Rome, Italy

Contact: maria.fauske@ffi.no (MFF); carlo.mannino@sintef.no, http://orcid.org/0000-0003-2974-9165 (CM); paolo.ventura@iasi.cnr.it, http://orcid.org/0000-0002-6923-3799 (PV)

Received: November 29, 2017
Revised: August 14, 2018
Accepted: December 13, 2018
Published Online in Articles in Advance: November 18, 2019

https://doi.org/10.1287/trsc.2019.0899

Abstract. Planning maritime surveillance activities in military operations and long-term defense planning is a huge task that is done manually today. Because 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. We defined and implemented a branch-and-price procedure to solve this formulation to exact optimality. Moreover, to tackle instances of practical size, we defined and applied efficient and effective heuristic techniques for solving the pricing problem. We show how our approach can plan up to 72-hour realistic missions with routing ships.

Funding: This work was supported by Norges Forskningsråd [Grant Opstra–267554].

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. Norway's economic and fishery zone consists of almost 2,000 000 km^2 of waters, which is about 80% the size of the Mediterranean Sea. Surveillance of this area is a huge task that 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 (FEs) in maritime surveillance and 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 defense planning (LTP). LTP at FFI is conducted as a direct support to the Norwegian Ministry of Defence, which makes the decisions 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 maximize the ability to react to a detection to obtain more information. The model that we describe in this paper is used to study the movement of force elements 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, for example, 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, whereas other, less important areas only have to be observed, for example, 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 throughout the territory (see Mingozzi 2005, Pirkwieser and Raidl 2012, Cacchiani, Hemmelmayr, and Tricoire 2014). 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, etc.). 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, PVRP corresponds to the special MSP case when every cell p can be observed only from the same cell p, and therefore, MSP generalizes PVRP. Second, another aspect is that, in PVRP, the planning horizon is typically subdivided into days, and 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. Therefore, some cells must be observed once a day, whereas others must be observed every hour or even continuously. Also, routes may last several periods or less than one period.

To cope with all of 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) 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 (Caprara, Fischetti, and Toth 2002, Harrod 2011), airplane routing and scheduling (Kjenstad et al. 2013, Avella et al. 2017), and classical traveling salesman (such as in Dash et al. 2012, 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. 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 the distinction between visiting a cell while traveling and observing a cell. In particular, other than 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 (CG) approach, which allowed us to solve to optimality small instances and 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 of the aspects that must be considered when solving (our version of) the MSP. For instance, Marlow, Kilby, and Mercer (2007) describe 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 Traveling Salesman Problem is used to model the problem; then, it is solved by classical heuristic approaches (such as genetic algorithms and 2-opt). Dridi, Krichen, and Guitouni (2012) describe a multiobjective optimization approach for solving a maritime surveillance problem where a set of resources is 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, because 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 that 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

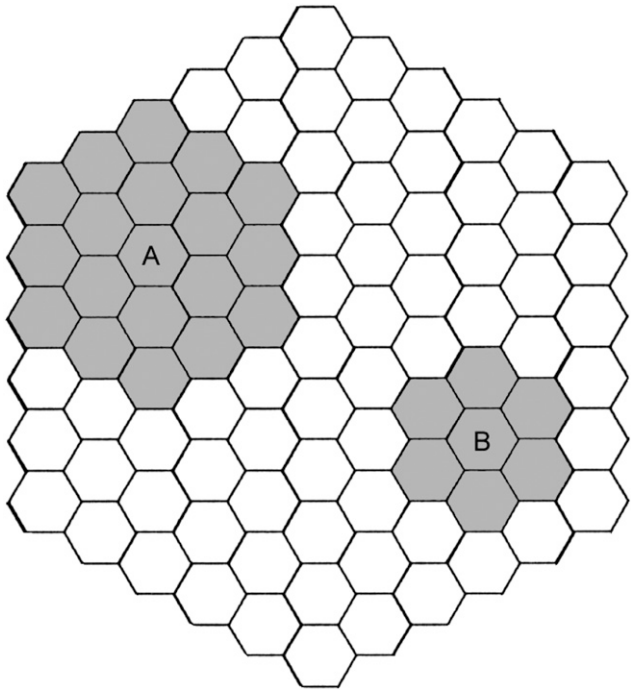
Roughly speaking, the MSP consists of finding an operational plan for a given force structure doing maritime surveillance in a given area during a specific planning horizon H. In particular, we want a plan that maximizes the ability to keep a recognized maritime picture sufficiently updated. In the model, 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.

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 FEs, such as vessels, aircraft, etc. Each force element $f \in F$ is characterized by a number of parameters, such as the following.

- σ_f: speed; given as the 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 a 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 that 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, whereas 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, whereas 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, defining a partition $\{F_1, \ldots, F_k\}$ of F. Briefly, the MSP can be stated as follows.

Figure 1. A Indicates Sensor Range of Two, and B Indicates Sensor Range of One.



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.

3. The Model

In this section, we introduce a pure binary linear programming model for the MSP. As mentioned in Section 1, the MSP can be considered as a variant of the PVRP, which in turn, can be viewed as a generalization of the classic Vehicle Routing Problem (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 them with the demand of periodic observations of cells.

3.1. Force Element Trajectories: Time-Expanded Network

We start by representing the movements of a single force element 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 that 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 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 $a = ((p, s), (q, s + 1))$ 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, because 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 $w_1, w_2, \dots, w_n$ such that $(w_i, w_j) \in A$ implies that $j > i$. 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 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. Because 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 dipath $(p_0 0, p_1 1, \dots, p_l l)$ with $p_0 = \phi_f$ has a straightforward interpretation: at any step $0 \le s \le l$, 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 $t = 1, \dots, \lfloor \frac{l}{\sigma_f} \rfloor$, 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 $p_{(t-1)\cdot\sigma_f}, p_{(t-1)\cdot\sigma_f+1}, p_{(t-1)\cdot\sigma_f+2}, \dots, p_{t\cdot\sigma_f-1}$.

3.2. 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 $T_p \le |H|$). Now, let $I = \{t, \dots, t + T_p - 1\}$ 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* $\mathcal{I}(p) = \{\{1, \dots, T_p\}, \{2, \dots, T_p + 1\}, \dots, \{|H| - T_p + 1, \dots, |H|\}\}$.

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 $\mathbb{O} = \{(p, I) : p \in P, I \in \mathcal{I}(p)\}$ 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) and (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.

3.3. 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 $O(R) \subseteq \mathbb{O}$ the set of observations covered by at least one 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 period is equivalent to finding a subset of feasible routes $R \subseteq \mathcal{R}$ such that $O(R) = \mathbb{O}$. In general, this request may be impossible to achieve, and therefore, 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 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 that is one if and only if route r is selected.

Also, with every $(p,I) \in \mathbb{O}$, we associate a binary variable x_{pI} that is one if and only if (p,I) is covered by some selected route. The routes in $\mathcal{R}$ are grouped in clusters $R_1, R_2, \ldots R_k$, 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 $q = 1, \ldots, k$.

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

$$\max \sum_{(p,I)\in\mathbb{O}} V_{pI} x_{pI}$$

s.t.

$$x_{pI} - \sum_{r\in R(p,I)} y_r \leq 0 \quad (p,I) \in \mathbb{O}, \tag{1.i}$$

$$\sum_{r\in R_q} y_r \leq n_q \quad q = 1, \ldots, k, \tag{1.ii}$$

$$y \in \{0,1\}^{\mathcal{R}}, \quad x \in \{0,1\}^{\mathbb{O}}.$$

By constraint (1.i), a variable x_{pI} can assume value one only if observation (p,I) is covered by some selected route. Constraint (1.ii) ensures 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. To tackle this difficulty, we apply delayed row and column generation. We assume that the reader is familiar with the approach (for details, see Alvras and Padberg 2001, Desaulniers, Desrosiers, and Solomon 2006). We start by solving a problem corresponding to a subset of columns relative to a subset of routes $\overline{\mathcal{R}} \subseteq \mathcal{R}$. Correspondingly, we have a set $\overline{O} = O(\overline{\mathcal{R}})$ of observations, namely the subset of pairs $(p,I) \in \mathbb{O}$, that can be covered by at least one route in $\overline{\mathcal{R}}$. 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):

$$\max \sum_{(p,i)\in\overline{O}} V_{pI} x_{pI}$$

s.t.

$$\text{(i)} \quad x_{pI} - \sum_{r\in R(p,I)\cap\overline{\mathcal{R}}} y_r \leq 0 \quad (p,I) \in \overline{O},$$

$$\text{(ii)} \quad \sum_{r\in R_q\cap\overline{\mathcal{R}}} y_r \leq n_q \quad q = 1, \ldots, k, \qquad \text{(RelMast)}$$

$$\text{(iii)} \quad x_{pI} \leq 1 \quad (p,I) \in \overline{O},$$

$$\text{(iv)} \quad y_r \leq 1 \quad r \in \overline{\mathcal{R}},$$

$$y \in \mathbb{R}_+^{\overline{\mathcal{R}}}, \qquad x \in \mathbb{R}_+^{\overline{O}}.$$

Observe that the restricted master depends (only) on the current set $\overline{\mathcal{R}}$ of columns, because the restricted observation set $\overline{O} = O(\overline{\mathcal{R}})$ also depends only on $\overline{\mathcal{R}}$. We make explicit this dependency by denoting the current master as RelMast($\overline{\mathcal{R}}$). 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 if the time limit is reached. More details will be given in Section 4.

The 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 nonpositive. To compute reduced costs, we need the optimal dual variables (with respect to the current restricted master). Therefore, after solving the relaxation of the current restricted master program, let $\overline{\pi}$ and $\overline{\lambda}$ be the optimal dual vectors associated with constraints RelMast.*i* and RelMast.*ii*, respectively.[1] Note that, at the next iteration, the newly introduced variables may extend existing constraints, or they may appear only in new constraints. In contrast, by construction, the new constraints only contain new variables.

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

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

If we include a new route $r \in \mathcal{R} \setminus \overline{\mathcal{R}}$, we also include the set of corresponding additional observations $O(r) \setminus \overline{O}$. Consequently, the new master program will also contain, other than all former variables and the new variable y_r, variables x_{pI} for $(p,I) \in O(r) \setminus \overline{O}$. Therefore, 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 is then

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

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

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

For $r \in \mathcal{R} \setminus \overline{\mathcal{R}}$, 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 $w(r) - \overline{\lambda}_q$.

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 (i) be given for any force element type, (ii) coincide with ϕ_f if the route must actually be a tour (a closed route), or (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 that we will discuss below.

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 $w : O \to \mathbb{R}_+$. With every arc $a \in A$, we associate a subset O_a of the ground set O. For any directed path $Q = (a_1, \ldots, a_q)$ in G, we let $O(Q) = \bigcup_{a\in Q} O_a$, 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 $\mathbb{O}$ of observations, whereas the weights correspond to the values $w_{p,I}$ as in (2). Moreover, for any $a = (u, v) \in A$, let $O_a \subseteq O$ be the set of observations covered by node u (and therefore, by arc a), and for each observation $(p, I) \in \mathbb{O}$, let $A(p, I) = \{a : (p, I) \in O_a\}$. 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 because 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_a for every $a \in A$, which is one if and only if arc a is taken in the chosen path r. Also, similar to what we did in problem (1), for each observation $(p, I) \in \mathbb{O}$, we introduce a binary variable x_{pI}, which will be one if and only if $(p, I) \in O_a$ for some $a \in r$. Also, for each $v \in N$, $\delta^+(v)$ and $\delta^-(v)$ denote the outgoing and incoming stars of v, respectively. The following is an ILP formulation for Problem 4.1:

$$\max \quad \sum_{p\in P} \sum_{I\in\mathcal{I}(p)} w_{pI} x_{pI}$$

s.t.

$$\sum_{a\in A(p,I)} z_a - x_{pI} \geq 0 \qquad (p, I) \in \mathbb{O}, \tag{3.i}$$

$$\sum_{a\in\delta^+(o)} z_a = 1 \tag{3.ii}$$

$$\sum_{a\in\delta^+(v)} z_a - \sum_{a\in\delta^-(v)} z_a = 0 \tag{3.iii}$$

$$v \in N \setminus (E \cup \{o\})$$

$$z \in \{0, 1\}^A, \quad x \in \{0, 1\}^{\mathbb{O}}.$$

Constraints (3.i) allow variable x_{pI} to be one 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, because G is acyclic (Property 3.1).

The pricing problem LSPP is hard to solve, which 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 $l \in \mathbb{R}^A$ associated with the arcs. Let $1, \ldots, n$ 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. $L_v = -\infty$, $v = 2, \ldots, n$. $L_1 = 0$
2. **for** $v = 2$ to n, **do**
3. $\quad L_v = \max_{u\in N^-(v)} \; L_u + l_{uv}$
4. $\quad pred[v] := \operatorname{argmax}_{u\in N^-(v)} \; L_u + l_{uv}$
5. **end for**

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

The first of our heuristic procedures is basically the straightforward application of Algorithm 1 DP to the expanded network $G = (N, A)$ with arc length $l_a = \sum_{(p,I)\in O_a} w_{pI}$ for $a = (u, v) \in A$. In other words, the length l_a of an arc a is the total (reduced) value of the observations that are carried out from a. The problem with this approach is that, when we consider a route

r of G, an observation $(p, I) \in \mathbb{O}$ can be covered by different arcs of the route, and therefore, 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. Therefore, 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 1 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 subroute from one to v (according to the vector $pred$). For a set of observations $O \subseteq \mathbb{O}$, let $\overline{w}(O) = \sum_{(p,I)\in O} w_{pI}$ be its value.

Algorithm 2 (Modified Dynamic Programming (MDP))

1. $Obs[v] = \emptyset$ for all $v = 1, \ldots, n$
2. **for** $v = 2$ to n, **do**
3. $\quad u^* = \text{argmax}_{u\in N^-(v)} \ \overline{w}(Obs[u] \cup O_{(u,v)})$
4. $\quad pred[v] := u^*$
5. $\quad Obs[v] = Obs[u^*] \cup O_{(u^*,v)}$
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 with 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 Algorithm 1 DP and Algorithm 2 MDP require visiting every node and every arc of the graph.

To speed up the procedure and reduce memory usage, we introduce a simplified "greedy" version of Algorithm 2 MDP. The idea is to visit only a small subset of nodes and arcs of G. 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 $P = (v_1, v_2, \ldots)$ necessarily visits nodes according to the sequence of layers, and therefore, v_i belongs to layer i. In the following algorithm, at iteration s, we select the sth 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. $\quad u^* = \text{argmax}_{u\in N^+(v)} \ \overline{w}(Obs \cup O_{(v,u)})$
4. $\quad pred[u^*] := v$
5. $\quad Obs = Obs \cup O_{(v,u^*)}$
6. $\quad v := u^*$
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-and-bound procedure of a commercial ILP solver. In the following, we will refer to this method as the Exact ILP (EXI) algorithm; we also present an exact combinatorial algorithm (EXC) to solve the pricing problem. Such an algorithm follows a branch-and-bound scheme based on a recursive enumeration of the paths of the graph G.

For each path P, we denote by $O(P)$ the set of observations collected by such a path (i.e., $O(P) = \cup_{a\in P} O_a$). Moreover, for each $O \subseteq \mathbb{O}$, we denote by P^*_u the path that maximizes $w(O(P_u) \cap O)$ among all paths from the origin o to node u, and we let $maxPath(u, O) := w(O(P^*_u) \cap O)$. The following recursive expression holds:

$$maxPath(v, O) = \max_{u\in N^-(v)} \{maxPath(u, O \setminus O_{(u,v)}) + w(O_{(u,v)} \cap O)\}. \quad (4)$$

Note that, if the maximum is attained for $u = t$ and P^*_t is the path associated with $maxPath(t, O \setminus O_{(t,v)})$, then $P^*_v = P^*_t \bullet (t, v)$. Then, if z is the destination node, our problem is equivalent to computing $maxPath(z, \mathbb{O})$, with the optimal route being P^*_z.

We implemented the recursive Equation (4) by Algorithm 4 EXC.

We remark that, different from most models in the VRP literature, here we cannot apply some adaptation of the "standard" labeling pricing algorithm. Basically, all of the pricing algorithms presented for different VRP variants from the seminal paper of Christofides, Mingozzi, and Toth (1981) up to the more recent contributions of Fukasawa et al. (2006), Baldacci, Mingozzi, and Roberti (2011), and Martinelli, Pecin, and Poggi (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.

By contrast, 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 instances of time), and therefore, a subpath of a largest set path is not necessarily a largest set path to a previous node. Moreover, standard dominance techniques based on capacity or 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 subpath of P_z from o to v, and let P_{vz} be the remaining subpath (i.e., $P_z = P_v \bullet P_{vz}$). 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 $UB(v)$ be an upper bound on the value of all paths from v to any destination node $z \in E$. Then, we have

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

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 v. If we have that

$$w(O(P_v^*)) + UB(v) \leq LB,$$

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

We apply this idea in Algorithm 4 EXC, where $UB(v)$ is precomputed by Algorithm 1 DP for each pair v, z.

Algorithm 4 (EXC)

```
1. maxPath((o, ∅))
2. function maxPath((u, O))
3.     for v ∈ N⁺(u), do
4.         if (v ∈ E), then
5.             if (w(O ∪ O_(u,v)) > LB), then
6.                 LB = w(O ∪ O_(u,v))
7.             end if
8.         else
9.             if (w(O ∪ O_(u,v)) + UB(v) > LB), then
10.                maxPath((v, O ∪ 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 4.1) is NP hard by reduction from the (unweighted) *maximum coverage problem* (MCP), which is 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 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 the following theorem.

Theorem 5.1. *The largest set path problem* (Problem 2.1) *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 that $p \leq m$.

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

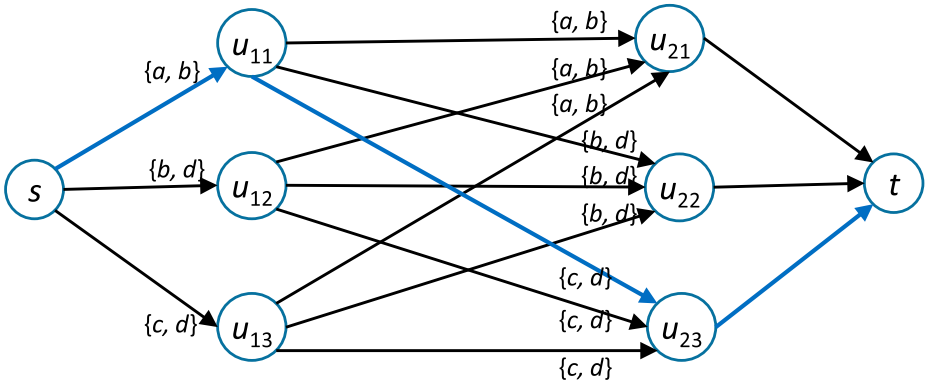
- $A_s = (s, u_{1i}), \forall i = 1, \ldots, m$;
- $A_t = (u_{pi}, t), \forall i = 1, \ldots, m$; and
- $A_u = (u_{ki}, u_{k+1j}), \forall k = 1, \ldots, p-1, \forall i, j = 1, \ldots, m$.

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 $\mathcal{S}$: therefore, node u_{ki} can be interpreted as the representative of set S_i in layer k.

Next, we let $O_a = S_i$ for each arc entering a node u_{ki}, with $k = 1, \ldots, p$ and $i = 1, \ldots, m$, and we let $O_a = \emptyset$ for each arc entering t (see the example in Figure 2). Therefore, 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 $\mathcal{S}^* = \{S_{i_1}, \ldots, S_{i_p}\}$ to MCP, with $i_1, i_2, \ldots, i_p \in \{1, \ldots, m\}$ 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 $P^* = \{(s, u_{1i_1}), (u_{1i_1}, u_{2i_2}), \cdots, (u_{pi_p}, t)\}$. We have that $w(P^*) = |\bigcup_{e \in P^*} O_e| = |O_{(s,u_{1i_1})} \cup O_{(u_{1i_1},u_{2i_2})} \cup \cdots \cup O_{(u_{pi_p},t)}| = |S_{i_1} \cup S_{i_2} \cdots \cup S_{i_p}|$: therefore, 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 $P^* = \{(s, u_{1i_1}), (u_{1i_1}, u_{2i_2}), \ldots, (u_{pi_p}, t)\}$ be an optimal solution to LSPP; then, we can construct an equivalent solution to MCP by

Figure 2. (Color online) 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 $p = 2$



Note. The blue arcs show an optimal solution of the corresponding LSPP instance.

letting $\mathcal{S}^* = \{S_{i_1}, \ldots, S_{i_p}\}$, 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, Mingozzi, and Roberti (2011). In particular, in the first loop (Algorithm 5 SolveMSP, 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), and (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), 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: that is, those appearing in RelMast($\mathcal{R}^z$). Then, we can restipulate 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 nonoptimal for the original problem (1). This is because any optimal solution may contain at least one route that is not in $\mathcal{R}^z$.

To find the optimal solution to problem (1), we can proceed as follows. Let $\overline{\mathcal{R}} = \mathcal{R}^z$ be the set of routes at the termination of the column generation loop. We build a set $\mathcal{R}^* \supseteq \overline{\mathcal{R}}$ that contains all of the routes appearing in (at least) one optimal solution. To this end, assume that 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 the generation of 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 if any.
7. **end for**
8. $\mathcal{R}^{i+1} = \mathcal{R}^i \cup_{q=1,\ldots,k} \{r_i^q\}$.
9. **until** no route with positive reduced cost is found
10. $\overline{\mathcal{R}} := \mathcal{R}^i$ and $\overline{w} := w^i$

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 of the routes with reduced cost greater than $LB_{MSP} - rVal$}.

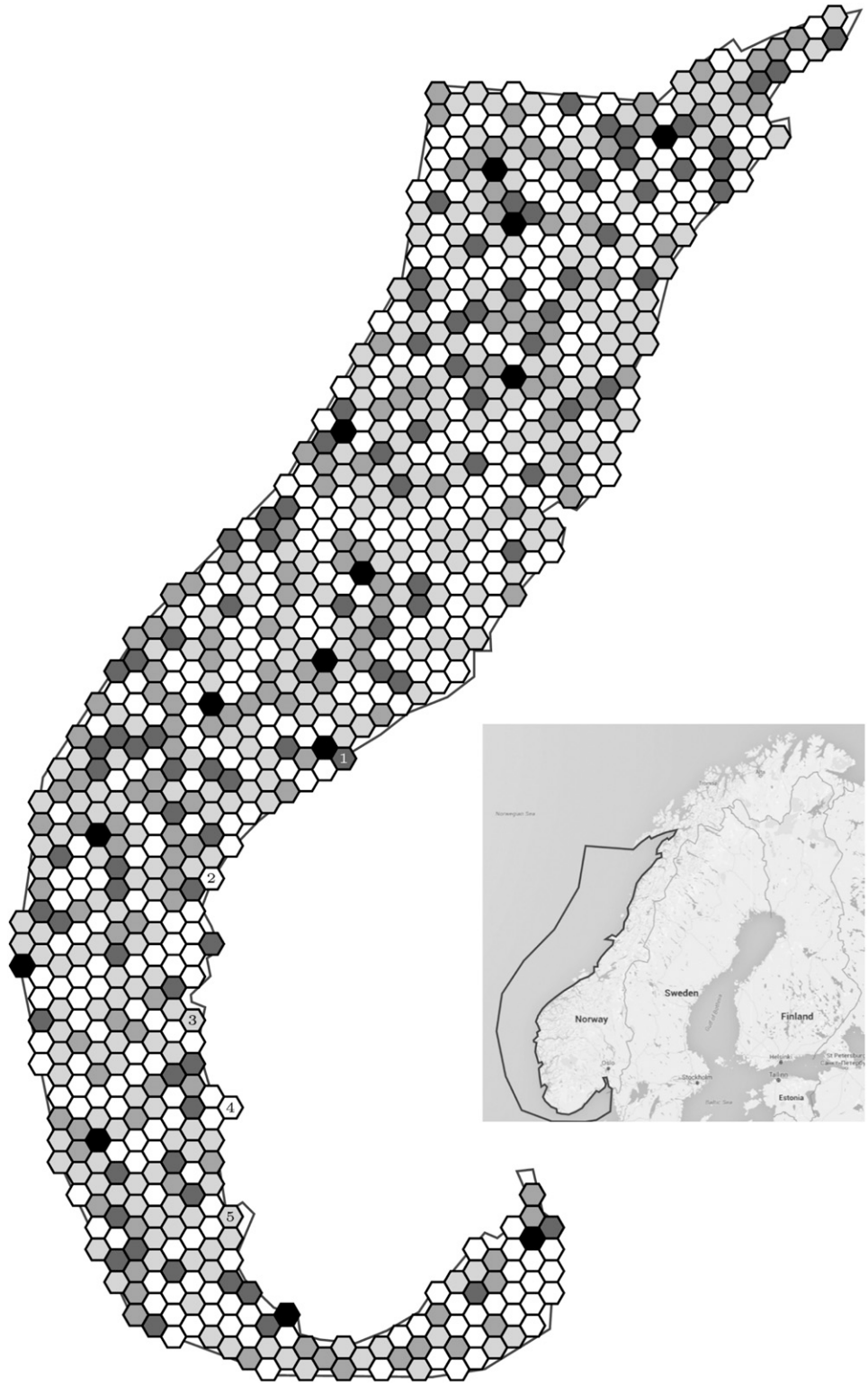
Integer Problem Solution

15. SOLVE(IntMast($\overline{\mathcal{R}} \cup \mathcal{R}^+$)).

Remark 1. Here, we want to stress that Algorithm 5 SolveMSP returns a certified optimal solution to MSP if its implementation fulfills the following requirements that are critical from a computational standpoint.

(i) At step 6, the pricing problem is solved exactly.

Figure 3. The Picture Shows the Grid That We Considered for the Computational Experiments



Notes. The colors of the hexagonal cells represent the corresponding observation periods: white is 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 in Table 1.

(ii) At step 14, the set $\mathcal{R}^+$ contains all of the routes with reduced cost greater than $LB_{MSP} - rVal$. Such routes can be produced by slightly amending algorithm EXC. First, the lower bound LB is initialized to $LB_{MSP} - rVal$ and never updated during the search. Moreover, we initially set $\mathcal{R}^+ = \emptyset$. Second, when the test at step 5 is

satisfied (i.e., a new "good" route is identified), we add it to $\mathcal{R}^+$.

(iii) The binary problem IntMast($\overline{\mathcal{R}} \cup \mathcal{R}^+$) is solved to optimality in step 15.

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 of the routes with a *promising* reduced cost or the branch-and-bound algorithm for solving the 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 that they are not too far. Indeed, for these large instances, the following happens.

(i) The exact combinatorial pricing Algorithm 4 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 being part of the optimal solution gets too large.

(iii) The binary problem IntMast($\mathcal{R}^*$) is very large (and very dense) and cannot be solved to proven optimality in the time limit that 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, and (iv) set an adequate time limit for the solution of the final master problem.

Although these compromises may seem to heavily weaken the overall approach, the computational results that we present in Section 7 give evidence that the procedure that we propose can still represent a useful tool for planning maritime surveillance, because 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

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

Table 1. Input Data for the Different Vehicle Types

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

Notes. 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.

7.1.2. The Force Elements. We consider 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 that we consider and their features. Because 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.

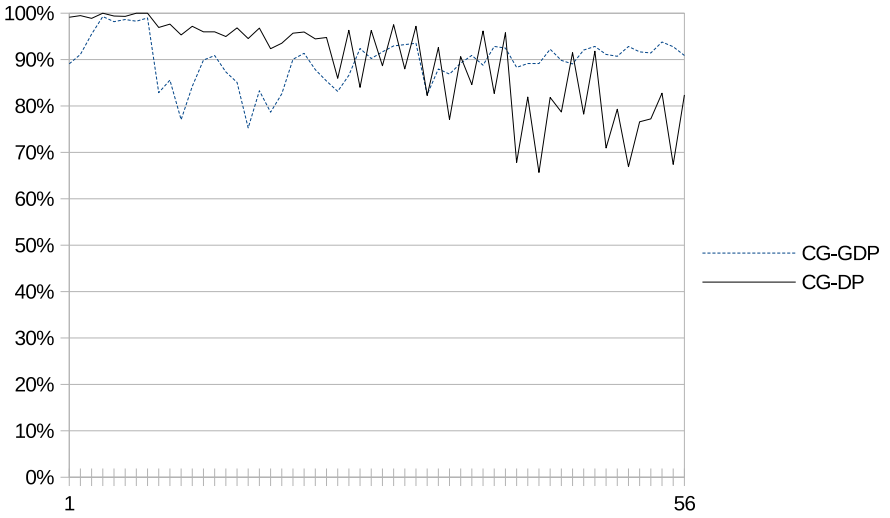
To diversify the experiments, for a fixed planning horizon and based on the default values indicated in Table 1, we generate eight distinct scenarios as follows. For each parameter column, we consider two different settings depending on the status of a control switch. Such switches are denoted by $\mathbf{CS^N}$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

7.1.3. Planning Horizon and Observations. We consider the following planning horizons: 6, 12, 24, 36, 48, 60, and 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)

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

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	2,647.8	0.3	27	8	0.0	3,325.4	104.5	1,358	526	51.6	3,517.5	145.3	576	192	57.9
24	0	0	1	2,367.9	0.5	39	13	0.0	2,752.0	49.1	944	255	20.0	2,843.4	31.1	346	78	12.8
24	0	1	0	2,413.7	0.6	47	16	0.0	2,833.6	106.0	1,520	538	47.9	3,068.4	106.5	469	133	38.7
24	0	1	1	2,004.0	0.2	30	10	0.0	2,268.7	50.8	1,184	279	15.4	2,425.8	49.1	488	126	18.1
24	1	0	0	2,256.9	0.5	41	13	0.0	2,397.9	25.7	635	147	15.5	2,506.1	23.2	176	45	15.5
24	1	0	1	1,978.5	0.2	29	10	0.0	2,078.2	19.9	628	150	10.5	2,166.1	23.6	209	60	12.9
24	1	1	0	1,763.7	0.3	45	15	0.1	1,897.0	13.2	602	147	10.1	2,008.3	27.0	223	63	19.2
24	1	1	1	1,436.9	0.4	53	17	0.0	1,595.0	6.4	472	124	4.5	1,683.3	16.7	201	51	9.2
36	0	0	0	9,262.7	22.4	54	17	0.2	9,578.0	463.0	1,785	464	86.8	11,141.5	1,833.3	555	111	103.8
36	0	0	1	8,040.9	36.7	94	33	0.4	8,942.3	708.1	2,423	651	94.9	9,284.5	1,265.2	1,145	258	121.7
36	0	1	0	8,804.8	70.6	104	35	0.4	8,005.5	419.2	2,179	485	67.1	9,529.3	1,816.8	515	103	82.1
36	0	1	1	6,910.3	26.5	71	20	0.2	7,373.9	1,273.5	2,574	683	102.5	7,658.7	1,661.7	1,320	301	123.7
36	1	0	0	7,275.2	21.0	48	16	0.2	7,037.6	500.1	1,297	306	88.7	7,937.2	991.3	759	192	200.6
36	1	0	1	6,458.4	21.3	58	19	0.1	6,775.9	348.7	1,212	286	62.7	6,947.9	750.8	688	169	127.9
36	1	1	0	5,644.5	37.6	113	34	0.4	5,327.5	409.1	1,462	349	84.0	6,054.7	525.8	959	229	173.7
36	1	1	1	4,745.9	19.2	72	22	0.2	4,931.0	151.4	1,019	226	30.4	5,074.0	168.2	615	146	46.1
48	0	0	0	15,803.6	152.2	92	32	1.3	15,768.0	1,803.0	2,193	481	175.1	19,169.5	1,843.9	250	50	99.7
48	0	0	1	14,488.5	53.3	88	25	0.3	15,252.8	1,816.0	2,408	493	159.5	16,469.3	1,832.6	250	50	92.7
48	0	1	0	14,162.2	57.7	87	27	0.4	12,568.0	1,751.3	2,914	625	198.9	16,302.2	1,929.0	305	61	101.0
48	0	1	1	11,849.8	83.9	148	45	0.6	12,038.5	1,821.5	2,616	541	125.6	13,282.0	1,829.3	310	62	83.7
48	1	0	0	12,470.8	43.3	77	24	0.3	11,606.0	1,802.7	1,580	356	209.9	13,718.5	1,805.6	380	76	148.7
48	1	0	1	10,758.9	11.1	50	17	0.2	11,654.9	1,110.6	1,930	426	136.8	12,119.2	1,816.6	385	77	140.3
48	1	1	0	9,682.2	30.1	83	26	0.4	8,624.1	1,250.0	1,783	440	203.3	10,429.9	1,198.5	893	198	262.6
48	1	1	1	8,121.5	13.5	58	17	0.2	8,413.3	1,185.4	2,018	429	152.9	8,781.9	1,,205.6	798	180	179.0
60	0	0	0	23,884.6	440.4	80	24	2.7	18,347.2	1,812.4	1,230	246	216.9	27,046.2	2,047.1	150	30	90.1
60	0	0	1	21,111.7	737.9	107	35	3.7	19,404.9	1,810.8	1,908	403	210.0	23,683.4	1,829.5	185	37	108.7
60	0	1	0	20,422.4	304.9	79	22	2.4	15,044.6	1,803.1	1,712	346	221.2	22,908.8	1,856.6	160	32	90.5
60	0	1	1	17,502.8	476.0	107	26	3.5	15,530.8	1,805.8	2,644	533	205.8	18,978.9	1,797.9	230	46	103.0
60	1	0	0	17,525.9	409.8	65	19	2.0	15,359.0	1,807.4	1,117	241	244.6	19,512.0	1,829.8	275	55	145.1
60	1	0	1	15,415.0	71.9	52	14	0.9	15,837.7	1,648.1	2,105	475	282.2	17,310.0	1,813.0	345	69	185.3
60	1	1	0	13,654.5	44.3	70	19	0.5	11,604.9	1,800.5	2,025	507	332.1	14,834.1	1,806.2	390	78	199.7
60	1	1	1	11,642.1	42.8	96	27	0.6	11,512.7	1,158.0	2,298	529	197.6	12,540.2	1,807.5	834	169	249.8
72	0	0	0	31,445.0	731.5	154	46	2.3	24,487.4	1,800.4	1,351	276	257.8	34,517.6	1,990.0	95	19	99.4
72	0	0	1	27,701.9	569.3	142	48	2.0	24,220.5	1,807.3	1,877	397	250.5	30,539.6	1,819.3	85	17	102.4
72	0	1	0	27,480.2	300.6	107	27	1.4	19,810.8	1,802.5	2,800	732	384.9	29,602.7	1,877.8	155	31	105.5
72	0	1	1	22,674.1	355.2	166	49	2.0	18,945.9	1,809.9	2,593	578	244.3	24,739.0	1,857.9	260	52	107.1
72	1	0	0	23,111.3	145.6	91	27	1.4	19,518.5	1,804.6	1,778	376	278.9	25,279.8	1,824.3	185	37	186.3
72	1	0	1	21,134.8	293.5	138	42	1.9	18,648.1	1,809.3	2,128	461	290.5	22,535.5	1,906.2	225	45	144.9
72	1	1	0	17,854.1	151.6	122	35	1.8	12,973.1	1,801.4	2,852	586	254.6	19,244.0	1,812.6	315	63	266.9
72	1	1	1	14,814.3	37.4	62	15	0.6	13,428.3	7,47.3	2,173	499	224.9	16,302.0	1,835.4	375	75	190.7

Figure 4. (Color online) Ratios Between the Bound Obtained by CG-DP and CG-GDP and the Bound Obtained by CG-MDP for Each Instance $1, \ldots, 56$



Note. The instance index is represented on the x axis.

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 seven planning horizons and eight scenarios; in total, 56 instances that are indexed from 1 to 56.

7.1.4. 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 ran them on a 3.5-GHz Intel(R) i7-4960X CPU machine with six dual-threads cores running on a 64-bit Linux Ubuntu 16.04 operating system.

Before delving into the discussion on our computational experiments, we give some implementation details on Algorithm 5 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, whereas 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 the 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}. The 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 5 SolveMSP) within a total time limit of 30 minutes. In the following, we will denote by CG-GDP, CG-DP, and CG-MDP the CG algorithms where the pricing problem is solved by procedure GDP, DP, or MDP, respectively. Because of the time limit of 30 minutes and because all of 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 ith row of the table is associated with instance $i = 1, \ldots, 56$. The first four columns of the table describe the following: H is the planning horizon, whereas $\mathbf{CS^N}$, $\mathbf{CS^R}$, and $\mathbf{CS^S}$ define 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 is the total time requested by the algorithm, cgN is the total number of routes generated, cgI is 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 of 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 reoptimization procedure. The best performances in terms of generated lower bounds are the ones produced by the CG-MDP algorithm, which consistently

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

Instance				CG-DP/CG-GDP						CG-MDP				
H	CS^N	CS^R	CS^S	rVal	rT	cgN	cgI	cgT	Algorithm	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	3,325.4	104.5	1,358	526	51.6	*CG – DP*	3,349.8	0.9	20	4	0.7
24	0	0	1	2,752.0	49.1	944	255	20.0	*CG – DP*	2,762.4	1.6	40	8	0.9
24	0	1	0	2,833.6	106.0	1,520	538	47.9	*CG – DP*	2,844.0	0.6	15	3	0.4
24	0	1	1	2,268.7	50.8	1,184	279	15.4	*CG – DP*	2,275.2	0.6	20	4	0.4
24	1	0	0	2,397.9	25.7	635	147	15.5	*CG – DP*	2,421.0	0.7	15	3	0.5
24	1	0	1	2,078.2	19.9	628	150	10.5	*CG – DP*	2,089.5	0.7	20	4	0.5
24	1	1	0	1,897.0	13.2	602	147	10.1	*CG – DP*	1,933.5	0.6	15	3	0.5
24	1	1	1	1,595.0	6.4	472	124	4.5	*CG – DP*	1,608.0	0.6	20	4	0.4
36	0	0	0	9,578.0	463.0	1,785	464	86.8	*CG – DP*	9,712.9	2.8	15	3	1.9
36	0	0	1	8,942.3	708.1	2,423	651	94.9	*CG – DP*	8,947.5	9.7	40	8	3.0
36	0	1	0	8,804.8	70.6	104	35	0.4	*CG – GDP*	8,842.8	5.0	20	4	2.1
36	0	1	1	7,373.9	1,273.5	2,574	683	102.5	*CG – DP*	7,387.8	8.2	40	8	2.3
36	1	0	0	7,275.2	21.0	48	16	0.2	*CG – GDP*	7,298.0	3.2	15	3	1.8
36	1	0	1	6,775.9	348.7	1,212	286	62.7	*CG – DP*	6,802.2	8.8	40	8	3.1
36	1	1	0	5,644.5	37.6	113	34	0.4	*CG – GDP*	5,736.5	1.2	10	2	1.0
36	1	1	1	4,931.0	151.4	1,019	226	30.4	*CG – DP*	4,969.8	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	1,816.0	2,408	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	1,821.5	2,616	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	1,110.6	1,930	426	136.8	*CG – DP*	11,776.1	17.4	30	6	5.4
48	1	1	0	9,682.2	30.1	83	26	0.4	*CG – GDP*	9,904.7	5.0	15	3	3.5
48	1	1	1	8,413.3	1,185.4	2,018	429	152.9	*CG – DP*	8,434.0	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	1,648.1	2,105	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	32.0	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

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 $i \in 1, \ldots, 56$ (on 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.

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) in Table 2 (the corresponding data are reported in the columns labeled CG-GDP and 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 from 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 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 of the instances with a 6- or 12-hour planning horizon. The values of such optimal solutions are presented in column rVal/Exact, whereas column rVal/Heu reports the values provided by the CG-MDP procedure.

For each of the two exact methods that we implemented, the ILP model and EXC, columns tT, $\mathbf{cgN_H}$, $\mathbf{cgN_E}$, and **cgI** present the total time requested, the number of heuristic routes generated, the number of exact routes generated, and the 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, because 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.

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 of the 6-hour planning horizon instances and the 12-hour ones with no ships with speed 2. Indeed, in the latter case, the number of possible routes[2] grows to 7^{24}, and the enumeration tree (even if fathomed by bounding the subpaths) grows too large.

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 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

Table 4. Solving the Master Linear Relaxation to Optimality

Instance				rVal			EXI				EXC		
H	$\mathbf{CS^N}$	$\mathbf{CS^R}$	$\mathbf{CS^S}$	Heu	Exact	tT	$\mathbf{cgN_H}$	$\mathbf{cgN_E}$	cgI	tT	$\mathbf{cgN_H}$	$\mathbf{cgN_E}$	cgI
6	0	0	0	229.0	229.0	1,440.0	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	1,293.4	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	1,823.7	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	1,932.0	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	96,736.1	159	80	101	8.0	124	7	32
12	0	1	1	434.6	435.7	481,147.9	153	241	168	19.2	108	22	48
12	1	0	1	410.5	411.0	361,806.4	66	224	173	6.8	45	4	18
12	1	1	1	305.0	305.0	92,795.6	54	76	65	6.9	46	9	21
12	0	0	0	633.5	633.5	—	—	—	—	10,733.0	200	18	71
12	0	1	0	540.0	545.0	—	—	—	—	17,828.0	308	58	128
12	1	0	0	459.5	463.0	—	—	—	—	3,232.2	65	7	27
12	1	1	0	358.0	359.0	—	—	—	—	4,482.6	66	17	29

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 results 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 a 30-minute time limit) reported in Table 2. As previously discussed, in Figure 5, we use the bound computed by CG-MDP as the 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. This is possibly because the increased computational time required to solve the current linear program relaxation (LP) at each iteration overcompensates for the stabilization effect of adding several columns.

Moreover, the 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.

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 5 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 a subset of the 12- and 15-hour planning horizon instances. Therefore, our experiments with the exact version of Algorithm 5 SolveMSP will be limited to this set of instances. The corresponding results are shown in Table 6.

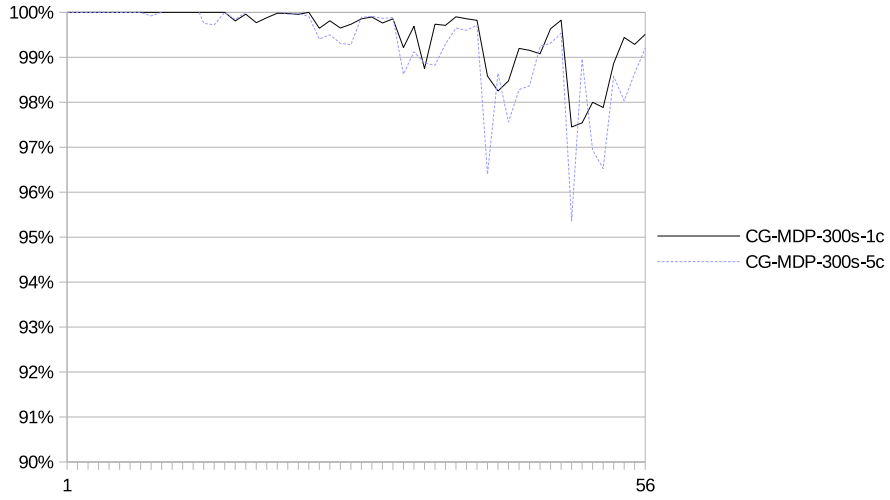
As in the previous tables, in column **rVal**, we report the optimal values of RelMast($\overline{\mathcal{R}}$) (because the pricing problem is solved to optimality, we have RelMast($\overline{\mathcal{R}}$ = RelMast($\mathcal{R}$)). Column $\mathbf{OPT_{MSP}}$ indicates the optimal value of IntMast($\mathcal{R}$) (i.e., the optimal value of MSP), whereas $\mathbf{LB_{MSP}}$ is the lower bound on $\mathbf{OPT_{MSP}}$ used in the Integrality Gap Pricing block of Algorithm 5 to

Table 5. Bounds Obtained by the CG-MDP Algorithm Within a Time Limit of 5 Minutes

Instance					
H	$\mathbf{CS^N}$	$\mathbf{CS^R}$	$\mathbf{CS^S}$	CG-MDP-300s-1crVal	CG-MDP-300s-5crVal
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	633.0
12	0	0	1	527.0	527.0
12	0	1	0	540.0	540.4
12	0	1	1	434.56	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	3,510.79	3,511.8
24	0	0	1	2,842.24	2,843.6
24	0	1	0	3,061.3	3,069.0
24	0	1	1	2,422.88	2,427.4
24	1	0	0	2,505.56	2,507.2
24	1	0	1	2,165.55	2,165.2
24	1	1	0	2,007.39	2,007.8
24	1	1	1	1,683.27	1,681.8
36	0	0	0	11,102.08	11,075.1
36	0	0	1	9,267.09	9,238.2
36	0	1	0	9,496.07	9,463.6
36	0	1	1	7,638.32	7,603.7
36	1	0	0	7,925.66	7,928.5
36	1	0	1	6,940.63	6,941.8
36	1	1	0	6,040.37	6,046.7
36	1	1	1	5,066.67	5,068.1
48	0	0	0	19,019.38	18,905.6
48	0	0	1	16,418.34	16,324.6
48	0	1	0	16,098.36	16,118.3
48	0	1	1	13,246.91	13,125.5
48	1	0	0	13,678.75	13,621.7
48	1	0	1	12,107.08	12,077.0
48	1	1	0	10,415.07	10,388.0
48	1	1	1	8,766.18	8,756.6
60	0	0	0	26,662.66	26,071.8
60	0	0	1	23,268.82	23,362.8
60	0	1	0	22,559.62	22,350.2
60	0	1	1	18,826.69	18,652.5
60	1	0	0	19,346.98	19,193.5
60	1	0	1	17,150.31	17,178.5
60	1	1	0	14,779.89	14,732.5
60	1	1	1	12,518.15	12,482.0
72	0	0	0	33,637.96	32,915.8
72	0	0	1	29,788.93	30,223.0
72	0	1	0	29,010.66	28,695.9
72	0	1	1	24,215.52	23,878.9
72	1	0	0	24,992.82	24,919.1
72	1	0	1	22,409.32	22,091.6
72	1	1	0	19,106.64	18,983.3
72	1	1	1	162,21.99	16,171.3

Note. 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.

Figure 5. (Color online) Ratios Between the Bound Obtained by CG-MDP-300-1c and CG-MDP-300-5c and the Bound Obtained by CG-MDP for Each Instance $1, \ldots, 56$



produce the igN routes of set $\mathcal{R}^+$. Furthermore, columns cgT, igT, and intT are the running times for the Column Generation Loop, the Integrality Gap Pricing, and the Integer Problem Solution, respectively.

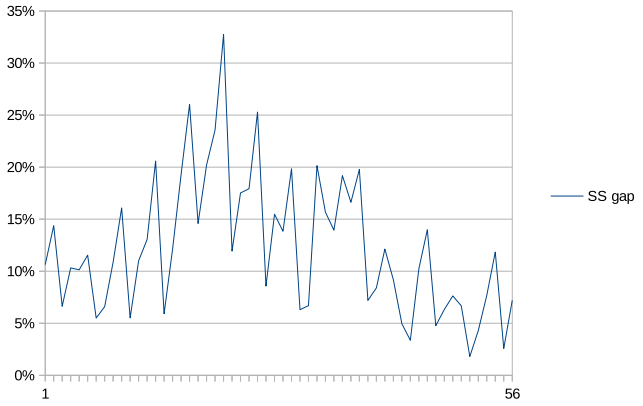
The results in Table 6 show that, for all instances with planning horizon up to 12 hours, the optimal solution to MSP can be found by using only the routes in $\overline{\mathcal{R}}$ generated in the Column Generation Loop. Indeed, in all of these cases, $LB_{MSP} = \lfloor rVal \rfloor$, and because the values V_{pI} are all integer, there is no need to add routes in the Integer Gap Pricing step. The same also occurs for the 15-hour instance in the last row in Table 6. For the remaining three 15-hour instances, the integrality gap $rVal - LB_{MSP}$ is greater than one, and, consequently, the exact pricing algorithm has to be called 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 2), 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. However, 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 100,000 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 5 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

Table 6. Solving MSP to Optimality

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	3,228.6	386.3	0.1
15	0	0	1	869.3	868.0	868.0	2,665	94.0	19.1	17.7
15	0	1	1	724.9	—	718.0	> 100,000	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

Figure 6. (Color online) The Picture Reports the Behavior of SS Gap in Table 7



these instances, at termination of the Column Generation Loop, all columns in $\mathcal{R}$ have nonpositive 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($\mathcal{R}$)).

Figure 6 and Table 7 describe the computational results obtained by the following heuristic version of Algorithm 5 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 2 MDP to find a route for every available vessel. The value of this solution is reported in column SS val of Table 7.

2. Column Generation Loop. Apply the CG-MDP algorithm with a 300-second time limit.

3. Integer Problem Solution. Solve to integer optimality the reduced master problem IntMast($\overline{\mathcal{R}}$) by the branch-and-bound algorithm of the CPLEX MIP solver with a one-hour time limit. The initial solution *SS* provides the first incumbent. We also set CPX_PARAM_PROB = 3 and use the Cplex function CPXcopyorder to give branching priority 100 to variables y and branching priority 0 to variables x.

In Table 7, column BS val reports the values of the best solutions to IntMast($\overline{\mathcal{R}}$) found within the time limit, whereas in column SS Gap, we report the percentage increment of BS val with respect to SS val. Such a value, also depicted in Figure 6, is more than 12% on average, and in a few cases, it is 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 that we propose here, is needed to obtain solutions of guaranteed quality. The results in Table 7 also show that such solutions can be obtained

Table 7. This Table Presents the Values of the Solutions for the MSP Obtained by Applying the CG-MDP Algorithm (with a 300-Second Time Limit) and Then Solving the Obtained Restricted Master Problem to Integral Optimality (Within a Time Limit of One Hour)

Instance				CG-MDP + branch and bound		
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	2,883.0	3,439.0	19.3
24	0	0	1	2,216.0	2,793.0	26.0
24	0	1	0	2,612.0	2,993.0	14.6
24	0	1	1	1,975.0	2,374.0	20.2
24	1	0	0	2,012.0	2,486.0	23.6
24	1	0	1	1,612.0	2,140.0	32.8
24	1	1	0	1,760.0	1,970.0	11.9
24	1	1	1	1,404.0	1,650.0	17.5
36	0	0	0	8,884.0	10,477.0	17.9
36	0	0	1	7,160.0	8,971.0	25.3
36	0	1	0	8,150.0	8,849.0	8.6
36	0	1	1	6,313.0	7,290.0	15.5
36	1	0	0	6,844.0	7,791.0	13.8
36	1	0	1	5,671.0	6,796.0	19.8
36	1	1	0	5,628.0	5,983.0	6.3
36	1	1	1	4,649.0	4,960.0	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	9,781.0	11,716.0	19.8
48	1	1	0	9,494.0	10,177.0	7.2
48	1	1	1	7,870.0	8,531.0	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	2,2424.0	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

within a time limit of 65 minutes for instances of size up to a 72-hour planning horizon.

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

- introduced a new periodic vehicle routing problem, which is very relevant for public bodies and authorities involved in periodic surveillance activities;
- developed 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;
- presented a column generation approach to the problem;
- showed that the pricing problem is NP hard;
- developed exact and heuristic column generation pricing algorithms; and
- carried 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 aircrafts, 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 (see e.g., Pinedo 2016) seems a natural decomposition approach for this kind of problem, and it is certainly worth investigation.

Endnotes

[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 $\overline{y}, \overline{x}$ be the current solution. Then, a new column/variable y_r to generate corresponds to a nonbasic variable (implying $\overline{y}_r = 0$) with positive reduced cost. However, 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 $y_r \leq 1$. However, because $\overline{y}_r = 0$, then the associated dual variable is 0. A similar argument can be used for the x variables.

[2] In each cell, vessels can make seven different choices, either moving to one of the six neighbors or staying still.

References

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

Avella P, Boccia M, Mannino C, Vasilyev I (2017) Time-indexed formulations for the runway scheduling problem. *Transportation Sci.* 51(4):1196–1209.

Baldacci R, Mingozzi A, Roberti R (2011) New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.* 59(5):1269–1283.

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

Cacchiani V, Hemmelmayr V, Tricoire F (2014) A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Appl. Math.* 163(1):53–64.

Caprara A, Fischetti M, Toth P (2002) Modeling and solving the train timetabling problem. *Oper. Res.* 50(5):851–861.

Christofides N, Mingozzi A, Toth P (1981) Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Math. Programming* 20(1):255–282.

Dash S, Günlük O, Lodi A, Tramontani A (2012) A time bucket formulation for the traveling salesman problem with time windows. *INFORMS J. Comput.* 24(1):132–147.

Desaulniers G, Desrosiers J, Solomon M, eds. (2006) *Column Generation*, vol. 5 (Springer Science & Business Media, New York).

Drexl M, Irnich S (2014) Solving elementary shortest-path problems as mixed-integer programs. *OR Spectrum* 36(2):281–296.

Dridi O, Krichen S, Guitouni A (2012) A multi-objective optimization approach for resource assignment and task scheduling problem: Application to maritime domain awareness. *Proc. 2012 IEEE Congress Evolutionary Comput.* (IEEE, Piscataway, NJ), 1–8.

Dyer ME, Wolsey LA (1990) Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Appl. Math.* 26(2–3):255–270.

Fukasawa R, Longo H, Lysgaard J, Aragão MPd, Reis M, Uchoa E, Werneck RF (2006) Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Programming* 106(3): 491–511.

Grob MJHB (2006) Routing of platforms in a maritime surface surveillance operation. *Eur. J. Oper. Res.* 170(2):613–628.

Harrod S (2011) Modeling network transition constraints with hypergraphs. *Transportation Sci.* 45(1):81–97.

Hochbaum DS, ed. (1997) *Approximation Algorithms for NP-Hard Problems* (PWS Publishing Co., Boston).

Ilavarasi K, Joseph KS (2014) Variants of travelling salesman problem: A survey. *Proc. 2014 Internat. Conf. Inform. Comm. Embedded Systems (ICICES)* (IEEE, Piscataway, NJ), 1–7.

Kjenstad D, Mannino C, Schittekat P, Smedsrud M (2013) Integrated surface and departure management at airports by optimization. *Proc. 5th Internat. Conf. Modeling, Simulation and Appl. Optim. (ICMSAO)* (IEEE, Piscataway, NJ), 1–5.

Marlow D, Kilby P, Mercer G (2007) The travelling salesman problem in maritime surveillance-techniques, algorithms and analysis. *Proc. Internat. Congress Modelling Simulation*, 684–690.

Martinelli R, Pecin D, Poggi M (2014) Efficient elementary and restricted non-elementary route pricing. *Eur. J. Oper. Res.* 239(1): 102–111.

Mingozzi A (2005) *The Multi-depot Periodic Vehicle Routing Problem*, Lecture Notes in Computer Science, vol. 3607 (Springer, Berlin, Heidelberg), 347–350.

Pinedo ML (2016) *Scheduling: Theory, Algorithms, and Systems* (Springer, Cham, Switzerland).

Pirkwieser S, Raidl G (2012) A column generation approach for the periodic vehicle routing problem with time windows. Working paper, Vienna University of Technology, Vienna, Austria.

Queyranne M, Schulz AS (1994) Polyhedral approaches to machine scheduling. Technical report 480/1994, Technische Universitat, Berlin.

Quttineh N, Larsson T, Van den Bergh J, Beliën J (2015) A time-indexed generalized vehicle routing model and stabilized column generation for military aircraft mission planning. Migdalas A, Karakitsiou A, eds. *Optimization, Control, and Applications in the Information Age*, Springer Proceedings in Mathematics & Statistics (Springer International Publishing, Switzerland), 299–314.

Schrijver L (2003) *Combinatorial Optimization* (Springer-Verlag, Berlin).

Stumpt E, Michael N (2011) Multi-robot persistent surveillance planning as a vehicle routing problem. *Proc. IEEE Internat. Conf. Automation Sci. Engrg., Trieste, Italy*.

Vatne D, Gisnås H (2014) MOBY—a simulation tool for evaluating maritime surveillance. Unpublished manuscript, Norwegian Defence Research Establishment, Kjeller, Norway.

Wolsey LA (1998) *Integer Programming*, Wiley Series in Discrete Mathematics and Optimization (John Wiley & Sons, New York).