Maskooki, Alaleh; Kallio, Markku

# A bi-criteria moving-target travelling salesman problem under uncertainty

*Published in:*
European Journal of Operational Research

*DOI:*
10.1016/j.ejor.2023.01.009

Published: 16/08/2023

*Document Version*
Publisher's PDF, also known as Version of record

European Journal of Operational Research 309 (2023) 271–285

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

# A bi-criteria moving-target travelling salesman problem under uncertainty

Alaleh Maskooki[a,*], Markku Kallio[b]

[a] University of Turku, Vesilinnantie 5, Turku FI-20014, Finland
[b] Aalto University School of Business, Ekonominaukio 1, Espoo FI-00076, Finland

ARTICLE INFO

*Article history:*
Received 2 May 2022
Accepted 5 January 2023
Available online 10 January 2023

*Keywords:*
Travelling salesman
Moving target
Stochastic programming
Dynamic programming
Integer programming

ABSTRACT

This article concerns a variant of moving target travelling salesman problem where the number and locations of targets vary with time and realizations of random trajectories. Managerial objectives are to maximize the number of visits to different targets and to minimize the total travel distance. Employing a linear value function for finding supported Pareto-efficient solutions, we develop a two-stage stochastic programming model. We propose an iterative randomized dynamic programming (*RDP*) algorithm which converges to a global optimum with probability one. Each iteration in *RDP* involves a randomized backward and forward recursion stage as well as options for improving any given schedule: swaps of targets and optimization of timing for visits. An integer linear programming (ILP) model is developed and solved by a standard ILP solver to evaluate the performance of *RDP* on instances of real data for scheduling an environmental surveillance boat to visit ships navigating in the Baltic Sea. Due to a huge number of binary variables, the ILP model in practice becomes intractable. For small to medium size data sets, the Pareto-efficiency of solutions found by *RDP* and ILP solver are equal within a reasonable tolerance; however, *RDP* is significantly faster and able to deal with large-scale problems in practice.



## 1. Introduction

This article addresses a case study problem arising from a real-world application of a surveillance boat measuring greenhouse gas emissions of ships (targets) navigating during a given time horizon in a specific sea area called *the work area*. The boat performs mobile measurements when it is in the vicinity of a ship. Generally we are interested in as many ships to be measured as possible in an itinerary; in practice, measuring all ships appearing in the work area during one working shift is not possible. On the other hand, we aim to find the minimum cost of operation which is in conflict with maximizing measurements. Therefore, we need to find the largest possible subset of ships to be visited in an optimal sequence, such that it leads to minimum possible cost (travel distance). The trajectories and the velocities of the ships during the day are predicted shortly before the measurement tour starts; however, predicted locations often have deviations from actual locations over the day. Major deviations in the predicted locations may make the predetermined itinerary impossible to implement especially when the schedule is tight. To deal with the uncertainty involved in this problem, we introduce a new stochastic moving target travelling salesman problem (MT-TSP) based stochastic programming (SP), and propose a method for finding an optimal routing of the surveillance boat which maximizes the number of measurement visits and minimizes the total travel distance via maximizing the expected value of a linear value function.

The deterministic bi-criteria integer linear programming (ILP) model underlying our case problem is introduced by Maskooki & Nikulin (2020) based on predicted trajectories of moving targets; it is an extension of the time dependent TSP (TD-TSP) by Picard & Queyranne (1978). The Pareto optimal frontier of the deterministic MT-TSP can be efficiently estimated by a customized genetic algorithm introduced by Maskooki, Deb, & Kallio (2022). Regarding the uncertainty of the deterministic optimization, Maskooki, Virjonen, & Kallio (2021) assess the prediction uncertainty by employing a risk measure in a mean-risk framework (Ruszczynski & Vanderbei (2003)). The risk measure helps the decision maker evaluate alternative a priori itineraries based on predictions, and make a balanced risk-adjusted decision. However, the plan is adopted for implementation independent of the realized locations observed afterwards. Instead, we extend the deterministic model by providing

* Corresponding author.
*E-mail address:* alamas@utu.fi (A. Maskooki).

https://doi.org/10.1016/j.ejor.2023.01.009

SP based decision support to the manager at the time of choosing an itinerary plan.

The deterministic version of our problem pertains to a specific type of MT-TSP introduced by Kryazhimskiy & Savinov (1995). Their work was followed, for instance, by Hammar & Nilsson (2002); Hassoun, Shoval, Simchon, & Yedidsion (2020); Helvig, Robins, & Zelikovsky (2003); Jiang, Sarker, & Abbass (2005), and Choubey (2013); all of these instances assume that targets move linearly with constant velocities. Although there is a vast body of literature on dynamic TSPs, MT-TSP variant is quite rarely discussed in literature. Bourjolly, Gurtuna, & Lyngvic (2006); Groba, Sartal, & Vázquez (2015), and Viel, Vaultier, Wan, & Jaulin (2019) consider some variants of the problem, where the target locations can be defined arbitrarily, and propose problem specific methods. MJHB (2006) reports a case study on maritime surveillance proposing online search heuristics adapted for a dynamic environment. However, the problem formulations in the above MT-TSPs differ from our deterministic model version.

The deterministic version of our problem also shares some of the elements of the orienteering problem (Golden, Levy, & Vohra, 1987), prize-collecting TSP (Balas, 1989), and time-dependent TSP (TD-TSP) (Abeledo, Fukasawa, Pessoa, & Uchoa, 2013; Donati, Montemanni, Casagrande, Rizzoli, & Gambardella, 2008; Furini, Persiani, & Toth, 2016; Malandraki & Daskin, 1992; Malandraki & Dial, 1996; Picard & Queyranne, 1978; Vu, Hewitt, Boland, & Savelsbergh, 2019). In the orienteering problem the goal is to visit as many nodes as possible subject to a given resource restriction on the tour. In a prize-collecting version the problem is to construct a tour which maximizes the sum of prizes collected (from visits to targets) minus the total travel cost. The time varying version of orienteering and prize-collecting TSPs, as well as standard TD-TSP formulation however cannot be used directly in our case. Since in those models target locations are fixed and the time to traverse a given arc varies, depending on the departure time from the origin node; in other words, the travel cost between two targets depends on the ordinality of targets in the sequence of visits and, unlike in our problem, real time is not considered. Furthermore, a commonly used objective function for TD-TSP models is to minimize the duration of the tour without permitting waiting at a node during its time window.[1] Therefore, as an assumption, a later departure cannot lead to an earlier arrival. However, in the case of moving targets, such assumption does not hold.

The literature includes a large number of probabilistic variations of TSP as well. The uncertainty and dynamic features in the underlying TSP can refer, for instance, to customer demand, customer locations, travel time, dynamically revealed random edge costs, subset of nodes to be visited, and delivery times of goods to a supplier's distribution system; see e.g., Bertsimas (1992); Bertsimas & Ryzin (1991); Jaillet (1988); Laporte, Louveaux, & Mercure (1992); Toriello, Haskell, & Poremba (2014), and Archetti, Feillet, Mor, & Speranza (2020).

There are two basic approaches to deal with uncertainty in optimization. First, in *a priori optimization* (Florio, Hartl, & Minner, 2020; Zhang, Ohlmann, & Thomas, 2014) the planner chooses the route based on probabilistic information and the plan is adopted independent of the realizations of the uncertainties observed afterwards. Second, in *adaptive optimization* a dynamic response follows observed realizations of random events. A priori approaches include chance-constrained programming (Blackmore, Ono, & Williams, 2011) and robust optimization (Ben-Tal, El Ghaoui, & Nemirovski, 2009; García & Peña, 2018; Zhang, Jia, Zhu, Adulyasak, & Ma, 2023). A large share of studies focuses on adaptive setting where random realizations are revealed over time while travelling along the chosen route. Such plan may be stated in terms of an optimal policy which defines state-dependent optimal choices for the problem; see e.g., Toriello et al. (2014).

Toriello et al. (2014) address a fairly similar stochastic problem to our case. They formulate their problem as a valid dynamic programming (DP) problem following Bellman (1962), and Held & Karp (1962). The state (node in the network) indicates the target, the set of remaining targets to be visited and the realized cost from the current target to all remaining targets. Approximate linear programming is proposed for solving their problem. Similarly, we have random costs (distances) associated to subsequent visits of targets, but unlike in Toriello et al. (2014), we cannot assume that at the time of arrival to a target, the distances to the remaining targets (not yet visited) are deterministic. Besides, the DP for an ordinary TSP in Bellman (1962); Held & Karp (1962) is intractable for all but the smallest instances, and the dynamics and uncertainty in our SP make the situation even worse.

A different *restricted DP algorithm* is proposed in Malandraki & Dial (1996) for solving TD-TSP to avoid the explosion of time and storage requirements by the exact DP. At each stage of the recursion, the method in Malandraki & Dial (1996) retains only the most promising partial tours, and the number of such partial tours is a user specified parameter. The heuristics of Malandraki & Dial (1996) does not guarantee optimality. In our randomized DP method, we use empirical probabilities to restrict the choice of nodes, but allow asymptotic convergence.

For modeling uncertainty a widely used method is stochastic programming (SP) proposed by Dantzig (1955); for subsequent developments, see e.g., Birge & Louveaux (2011); Shapiro, Dentcheva, & Ruszczyński (2009). In SP a limited number of scenarios with their occurrence probabilities are generated to fit given distributions. In multi-stage SP the scenarios form a tree where branches originating from nodes reveal accumulating information on realizations of uncertainty over time stages. Choices are contingent to such information and the expected value of the objective function is determined based on the scenarios. Input data for SPs may be continuous distributions or a large data set providing an empirical discrete distribution of realizations. For scenario generation methods, see e.g. Kaut & Wallace (2007). Most of the stochastic vehicle routing problems can be modeled as two-stage SPs Oyola, Arntzen, & Woodruff (2018), see for instance studies by Adasme, Andrade, Leung, & Lisser (2016) for TSP with both deterministic and uncertain edge weights, Beraldi, Ghiani, Musmanno, & Vocaturo (2010) on probabilistic multi-vehicle pickup and delivery problem, and Jabali, Rei, Gendreau, & Laporte (2014) on capacitated vehicle routing problem with stochastic demands.

As discussed above, the problem of our interest is a new stochastic MT-TSP which shares some of the elements of the variants mentioned above but differs fundamentally from these models. First, in our MT-TSP, the locations of nodes (targets) are non-stationary and random; second, each target is only accessible in a given time window which can also be random; third, all targets need not be visited. We adopt two-stage stochastic programming to model uncertainty in target trajectories of our MT-TSP. To deal with the two criteria, we employ a linear value function whose expected value is maximized.

Our two-stage SP is in harmony with conventional multi-stage SP where new information is revealed at the end of the first stage and subsequent choices in the model account for such information. While using multi-stage SP, the decision maker is primarily interested in finding best choices in the first stage; at the end of the first stage period, the model is revised based on updated information. Likewise, our two-stage SP is proposed for use in a rolling horizon basis where the second stage only serves for choosing a

[1] Waiting occurs only if the arrival time at a location is before the beginning of its time window.

good solution for the first stage to be implemented; at the end of the first stage, the subsequent model with refreshed information is considered.

In the state space formulation by Bellman (1962); Held & Karp (1962) and Toriello et al. (2014), the role of the set of remaining targets is to ensure that each target will be visited (at most) once. We propose to deal with this condition differently. As suggested by Stieber & Fügenschuh (2022) we use a discrete time formulation. We let the state (node in the network) indicate the location of the target in a (discrete) time slot and use recursions similar to DP for finding a tour; however, to avoid duplicate visits to targets, we need to keep track of the set of targets recursively chosen for visits. Thereby, standard stochastic DP (SDP) recursion can be used to produce a feasible solution for our SP; however, the conditions required for DP (Bellman, 1957) to yield an optimal solution are violated, and consequently, such SDP solution can be far from optimal. To overcome this drawback, unlike in SDP, we propose an iterative randomized dynamic programming (*RDP*) algorithm with alternating and interacting backward and forward recursions where subsets of nodes (states) are randomly drawn at each stage of the recursions. In backward recursions, such random draws employ the statistics of best tours produced by forward recursion (and vice versa). We provide a proof that the best solution found over iterations converges with probability one to an optimal solution in theory. We also show that a near optimal solution can be expected in early iterations of *RDP*.

We formulate an ILP model for our two-stage SP and solve it by an ILP solver, in order to evaluate *RDP* solutions in terms of the quality with respect to the run time *RDP* takes to return such solution. It also helps estimate the average rate of convergence to the optimal solution in general. For generating scenarios of target trajectories, we introduce an antithetic simulation based on a stochastic model of deviations of realized trajectories from predicted ones. We employ instances of real data for scheduling a surveillance boat to visit ships navigating in the Baltic Sea. For large-scale test instances, due to a large number of binary variables, the ILP model could not be solved to be used for quality evaluation. However For small to medium size data sets, the Pareto-efficiency of solutions found by *RDP* and ILP solver are equal within a reasonable tolerance; Furthermore, *RDP* is significantly fast and able to deal with large-scale problems in practice. In addition, based on the experimental results, we show that an efficient solution (an optimal or a near optimal solution sufficient for practical purposes) can be expected in early iterations of *RDP*.

An important and general aspect of the proposed *RDP* approach is that it shows how dynamic programming can be modified to solve large-size network optimization problem formulations, which violate the necessary conditions for standard DP to yield an optimal solution. In particular, our numerous (over 200) test instances with *RDP* on real-world problems show a highly promising performance and usefulness in practical applications.

The rest of the paper is organized as follows. Section 2 defines the deterministic routing problem, two-stage stochastic programming for the routing problem, and ILP model for the two-stage stochastic programming. Sections 3 and 4 introduce the solution method (*RDP*). It begins with the concept and formulation used for the deterministic case, followed by discussion on how it is extended for solving the two-stage stochastic version. Section 5 presents numerical results using real-life instances.Finally, a conclusion briefly summarizing the main contributions and anticipated future work is given in Section 6. An Appendix in the supplementary material includes the convergence proof for *RDP*, optional improvement steps for RDP solutions, a discussion on a trajectory prediction approach, and a stochastic model for scenario generation.

## 2. Optimal routing in a dynamic network

As mentioned, the problem of our case study arises from a real-world application of an emission control boat measuring greenhouse gas emissions of ships in the work area. We introduce a new version of MT-TSP with the following features: (i) The number of targets change over time. (ii) Targets have time windows during which they can be visited; the window starts when the ship enters the work area and ends when it leaves the area. (iii) The trajectories of the moving targets and their varying velocities can be defined freely. (iv) The total number of targets to be visited is endogenous, and in general, visiting all targets is not possible given the time windows of targets, the time horizon, limited work area and the speed of the surveillance boat. (v) The problem is bi-criteria: maximize the number of measurement visits and minimize the total travel distance of the boat. (vi) Due to prediction errors, trajectories of the targets involve uncertainty.

Given a set of moving targets passing through the work area in a given day we consider a time horizon $T = [t_0, t_2]$. For two-stage SP, we further subdivide the horizon into the first stage $[t_0, t_1]$ and second stage $[t_1, t_2]$. The predicted locations in early hours of the planning horizon are more precise than in later hours. Therefore, we assume known target locations over the first stage $[t_0.t_1]$, and use a finite number of scenarios for trajectories over the second stage $[t_1, t_2]$. To deal with the two criteria, we employ a linear value function[2] and maximize its expected value given the set of scenarios. Even though we use the value function as a single criterion, we solve a bi-criteria problem. Consequently, all targets are not necessarily visited.

The problem can be cast as an ILP model; however, even the deterministic case may well be intractable, and thus impractical for large-size real-world data sets; for examples, see Table B.3 in Maskooki et al. (2022). To avoid the curse of dimensionality in the ILP problem, we develop a randomized dynamic programming approach (*RDP*), employing iteratively interacting backward and forward recursions. Our case study in Section 5 shows that such interactive exchange of information is essential for *RDP* to work at best for our SP.

Although the focus of our article is on stochastic programming for optimal routing, it is convenient to begin by introducing *RDP* for the deterministic case. Therefore, Section 2.1 states the deterministic routing problem, then Section 2.2 presents the two-stage SP problem and Section 2.3 its ILP formulation. Thereafter, *RDP* is introduced in Sections 3 and 4.

### 2.1. The deterministic routing problem

Consider the time span $T$ with a set $N = \{1, 2, \ldots, n\}$ of $n$ targets $i$ present in the work area during this time span. Let $i = 0$ refer to the depot. In order to formulate and solve the problem, we discretize the time horizon into $m$ time slots $k = 1 \ldots m$ of equal length $w$, where $mw = t_2 - t_0$ and time slot $k$ refers to the interval $[t_0 + (k-1)w, t_0 + kw)$. Let the time slot $k = 0$ denote the initial time $t_0$. Thus, the time $s_k$ at the beginning of time slot $k$ is $s_k = t_0 + (k-1)w$, for $k > 0$, and $s_0 = t_0$.

Since the length $w$ is chosen relatively small compared to $T$ and the speeds of the targets (ships) are modest, for the sake of simplicity in the model, we assume the location of targets remain unchanged during each time slot.

[2] Only supported Pareto points can be generated using a linear value function; however, for our scheduling problems in practice, the efficient frontier is expected to be almost convex; for deterministic examples, see Maskooki et al. (2022) where only some least interesting Pareto points associated with a small number of visits are missed by a linear value function.

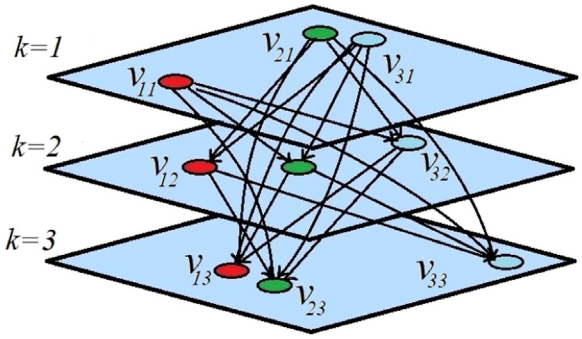


**Fig. 1.** Part of a directed network with time slots $k = 1, 2, 3$ and nodes $v_{ik}$ for $i = 1, 2, 3$. Edges from and to the depot are not shown.

We define a network flow model over a layered graph, where each layer corresponds to a given time slot $k$, $k = 0, 1, 2, \ldots, m$, and consists of *nodes* $v_{ik} \in R^2$, for $i \in N \cup \{0\}$ present at time slot $k$. Each node $v_{ik}$ is a coordinate vector stating the location of target $i$ in the work area at time slot $k$; for an illustration, see Fig. 1. Let $S_i$ be the set of time slots when target $i$ is present in the work area. The depot node is present in all layers $k$; hence $S_0 = \{0, 1, \ldots, m\}$ and the traveler can initiate from and return to the depot at any time to complete the tour. However, for notational convenience and without loss of generality, we assume the tour starts at time slot $k = 0$ and ends at time slot $k = m$. Each *arc* $(ik, jl)$ from node $v_{ik}$ to $v_{jl}$ connects nodes in distinct layers $k$ and $l$, $k < l$. The length of time slots is chosen short enough not to include more than one processing (measurement in our case). Hence, the nodes of the graph in the same layer are not connected.

For the deterministic model, we assume the locations of $n$ targets are based on accurate predictions within the time horizon. The length of each arc $(ik, jl)$ is defined by the Euclidean distance $d_{ik}^{jl} = \| v_{ik} - v_{jl} \|$. The speed of the boat is assumed to be limited by a fixed value $c$. Therefore, corresponding to each distance $d_{ik}^{jl}$ there is a minimum travel time with a speed $c$ which is denoted by $t_{ik}^{jl} = d_{ik}^{jl}/c$. Visiting target $i$ needs a processing time $p_i$ before leaving and visiting the next target; let $p_i = p$, for all $i \in N$ and let $p_0 = 0$, for the depot.

The arc $(ik, jl)$ indicating travel from node $v_{ik}$ to $v_{jl}$ is defined only if $i \neq j$ and both nodes $v_{ik}$ and $v_{jl}$ are in the work area: $k \in S_i$ and $l \in S_j$. Furthermore, the arc $(ik, jl)$ is not feasible[3] if $s_k + p_i + t_{ik}^{jl} \geq t_0 + wl$. In such instances, the arc $(ik, jl)$ is omitted from the network. Formally, we define the set $\Xi$ of *admissible arcs* in the network as follows:

$$\Xi = \{(ik, jl) \mid i, j \in N \cup \{0\},\ i \neq j,\ k \in S_i,\ l \in S_j,\ k < l,\ s_k + p_i + t_{ik}^{jl} < t_0 + wl\}. \quad (1)$$

For the two criteria, we denote the number of nodes (targets) visited during the time horizon $T$ by $\alpha$, and the total travel distance from the depot to the targets to be visited and back to the depot by $z$. One may solve the problem by determining the efficient frontier first and letting the user choose the most preferred solution thereafter; see Maskooki & Nikulin (2020) and Maskooki et al. (2022). Instead, for finding supported Pareto-optimal solutions, we employ a linear value function $\lambda\alpha - z$ to be maximized; it is a linear combination of the two objectives with a weighting parameter $\lambda > 0$ defining preferences in terms of trade-off among the objectives. Hence, we employ the standard weighted sum method for our bi-criteria problem.

[3] Even if the processing of target $i$ starts at time $s_k$, at the beginning of time slot $k$, the arrival time in the location $v_{jl}$ is beyond the time slot $l$, given the maximum speed of the boat.

**Definition 1** (Feasible deterministic tours)**.** A deterministic tour $\pi = \{v_{ik}\}$ is an ordered set (a sequence) of nodes defining an itinerary plan starting with node $v_{00}$ at the depot, visiting a number of targets $i \in N$ in time slots $k$ in locations $v_{ik}$ and returning to the depot at $v_{0m}$. A tour $\pi$ is feasible, if each target is visited at most once, and the arcs along the tour $\pi$ are in the admissible set $\Xi$. Given a feasible tour $\pi$, $\sigma(\pi)$ denotes the ordered set defining the sequence of targets $i \neq 0$ visited in tour $\pi$.

For a feasible tour $\pi$, let $\alpha(\pi)$ be the number of targets visited, $z(\pi)$ the total travel distance and $V(\pi)$ the value function. Given the weight $\lambda$, the problem is to find a tour $\pi$ in the set $\Pi$ of feasible tours to maximize the linear value function:

$$\max_{\pi \in \Pi} V(\pi) = \lambda\alpha(\pi) - z(\pi). \quad (2)$$

We assume that the weight $\lambda$ reflects the user's trade-off preferences. It can be estimated by a simple trade-off question: Given $\alpha$ and $z$, how many extra kilometres $\delta$ is accepted by the user at most in return to an extra ship visited. The parameter $\lambda$ is then determined by indifference equation $\lambda(\alpha + 1) - (z + \delta) = \lambda\alpha - z$ and thus $\lambda = \delta$ as a reward in kilometer per one extra visit. To tackle the problem (2), an ILP formulation is possible (see Section 2.3) but in practice it suffers from a huge number of binary variables. Instead, we adopt concepts of DP to develop an alternative solution method applicable in practical situations.

### 2.2. Two-stage stochastic programming for the routing problem

Next, we introduce the two-stage SP problem for optimal routing. For the locational uncertainties, we note that target locations during early hours in $T_0 = [t_0, t_1]$, $t_0 < t_1 < t_2$, of the planning horizon are known with a higher precision than during later hours in $T_1 = [t_1, t_2]$. Thus, we assume accurate predictions for target locations over early hours $t_1 - t_0$. For the rest of the day we use a set of $M$ equally likely scenarios $s$, $s \in C = \{1, \ldots, M\}$, of trajectories for targets. As in Section 3, we assume there is at least one visit in the deterministic period $T_0$ at an optimum.

For two-stage SP, we use our previous discretization of the time horizon into $m$ time slots $k$ of length $w$ and let time slot $k = 0$ refer to time $t_0$. The deterministic time span $[t_0, t_1]$, is such that $t_1 = t_0 + wk^*$, for some positive integer $k^* < m$. Hence, $k^*$ is the last time slot in $T_0$. For the deterministic time period $T_0$, let $N_0$ be the set of targets expected to appear in the work area and the coordinate vector $v_{ik}$, the location of target $i \in N_0$ at time slot $k$, is given by the prediction.

Again, we define a network flow model over a layered graph, where each layer corresponds to a given time slot $k$, $k = 0, 1, 2, \ldots, m$. For the first stage $T_0$, layer $k$ includes nodes $v_{ik}$, for targets $i \in N_0 \cup \{0\}$ present in the work area at time slot $k$. For the second stage $T_1$, let $N_s$ be the set of targets appearing in the work area in scenario $s$. The graph involves a sub-layer for each scenario $s$ consisting of nodes $v(s)_{ik}$, for $i \in N_s$ present at time slot $k > k^*$. Additionally, the layer at $k^*$ at the end of $T_0$ may contain of dummy nodes $v$ (without a target) serving as possible transition nodes from $T_0$ to the scenarios in $T_1$; it may be desirable to travel to a location $v$ even though there is no target but that may be close to many potential target sites by the time the realized scenario is observed. Then the location $v$ is subject to optimization as well. However, we tested real-world problems with such a transition formulation and conclude that instead it is justified to use a relaxation to be defined and justified below.

As before, let $S_i$ be the set of time slots when target $i \in N_0$ is present in the work area during $T_0$. For each scenario $s$, let $S_{si}$ be the set of time slots when target $i \in N_s$ is present in the work area during $T_1$. The depot node is present in all layers $k$; hence $S_0 = S_{s0} = \{0, 1, \ldots, m\}$ and the boat can initiate from and return

to the depot at any time to complete the tour. However, as in the deterministic case, for notational convenience and without loss of generality, we assume the tour starts at time slot $k = 0$ and ends at time slot $k = m$.

For $T_0$, let an arc $(ik, jl)$, where $i, j \in N_0 \cup \{0\}$ with $i \neq j$, $j \neq 0$ and $0 \leq k < l \leq k^*$, indicate travelling from node $v_{ik}$ to $v_{jl}$. The set $\Xi$ of admissible arcs accounting for timing requirements is given by

$$\Xi = \{(ik, jl) \mid i \in N_0 \cup \{0\},\ j \in N_0,\ i \neq j,\ k \in S_i,\ l \in S_j,\ k < l,\ s_k + p_i + t_{ik}^{jl} < t_0 + wl\}. \quad (3)$$

A similar definition holds for the second stage $T_1$ during each scenario $s$, where $v(s)_{ik}$ indicates the location of target $i$ at time slot $k$ in scenario $s$. Given scenario $s \in C$, let arc $(ik, jl)$ with $i, j \in N_s \cup \{0\}$, $i \neq j$, $i \neq 0$ and $l > k > k^*$ corresponds to travelling from node $v(s)_{ik}$ to node $v(s)_{jl}$. Furthermore, definitions of travel distance and travel time between two consecutive targets become scenario dependent accordingly, i.e. $d(s)_{ik}^{jl} = \|v(s)_{ik} - v(s)_{jl}\|$ and $t(s)_{ik}^{jl} = d(s)_{ik}^{jl}/c$ are the travel distance and time, respectively, between the two locations (nodes) $v(s)_{ik}$ and $v(s)_{jl}$. For scenario $s$, the set $\Xi(s)$ of admissible arcs in $T_1$ is

$$\Xi(s) = \{(ik, jl) \mid i \in N_s,\ j \in N_s \cup \{0\},\ i \neq j,\ k \in S_{si},\ l \in S_{sj},\ k < l,\ s_k + p_i + t(s)_{ik}^{jl} < t_0 + wl\}. \quad (4)$$

For modeling the transitions from the first stage $T_0$ to scenario $s \in C$ of the second stage $T_1$ we use a relaxation to be explained shortly. We let the transition occur from time slot $k \leq k^*$ to time slots $l > k^*$ of scenario $s$, and define a distinct arc $(ik, jl)$ where $i \in N_0$, $j \in N_s \cup \{0\}$ and $i \neq j$ to indicate travelling from coordinate $v_{ik}$ to $v(s)_{jl}$; the distance of the transition is $d'(s)_{ik}^{jl} = \|v_{ik} - v(s)_{jl}\|$, the travel time is $t'(s)_{ik}^{jl} = d'(s)_{ik}^{jl}/c$ and the set $\Xi'(s)$ of admissible arcs is given by

$$\Xi'(s) = \{(ik, jl) \mid i \in N_0,\ j \in N_s \cup \{0\},\ i \neq j,\ k \in S_i,\ l \in S_{sj},\ s_k + p_i + t(s)_{ik}^{jl} < t_0 + wl\}. \quad (5)$$

A tour $\pi$ in our SP formulation is a tree, connecting the root node $v_{00}$ to the terminal nodes $v(s)_{0m}$ for all scenarios $s \in C$; $\pi$ defines a sequence of nodes (visits) in $T_0$ as well as in each scenario $s \in C$ of period $T_1$. Arcs along the tour $\pi$ are in the set $\Xi$ in $T_0$ and in $\Xi(s)$ in $T_1$ for each scenario $s \in C$. At the time of transition from a node in $T_0$ to a node in scenario $s \in C$, the arc is in $\Xi'(s)$.

**Definition 2** (Feasible tours for SP)**.** Tour $\pi$ is feasible, if the arcs along the tour are admissible (defined by (3)–(5)), each target is visited at most once during the entire time horizon $T$, and there is a *unique transition node* $v_{\hat{i}\hat{k}}$ in $T_0$ from which transition occurs to some node $v(s)_{jl}$ of scenario $s$, for all $s \in C$. The node $v_{\hat{i}\hat{k}}$ is the last node to be visited in $T_0$ with $\hat{i} \in N_0$ and $\hat{k} \leq k^*$.

The value function in (2) is replaced by the expected value $V(\pi)$ of the linear value function with equal probabilities over $M$ scenarios. Given a tour $\pi$, let $\alpha_0$ and $\alpha_s$ denote the number of visits in $T_0$ and in scenario $s \in C$, respectively; similarly, let $z_0$ and $z_s$ denote the travel distance in $T_0$ and in scenario $s$ (starting from the transition node), respectively. Given the set $\Pi$ of feasible tours $\pi$ and a weight $\lambda > 0$ for the value function, the problem of maximizing the expected value of the linear value function is as follows

$$\max_{\pi \in \Pi} V(\pi) = \lambda\alpha_0(\pi) - z_0(\pi) + (1/M)\sum_s(\lambda\alpha_s(\pi) - z_s(\pi)). \quad (6)$$

As mentioned, our formulation for transition from $T_0$ to $T_1$ involves a relaxation. In principle, the transition occurs at the end of $T_0$ from some node $v$ where the scenario realization is observed. Such a dummy node $v$, may reside anywhere in the work area and the coordinate vector $v$ is subject to optimization along with the rest of the tour. To see how and why we use a relaxation, consider a tour $\pi$ which may or may not include a dummy transition node $v$. Suppressing targets and time slots, let $v_0$ in $\pi$ denote the last node (with a target) in $T_0$ and let node $v_s$ in $\pi$ be the first node in scenario $s$, for all $s$. Suppose $\pi = \bar{\pi}$ is an optimal tour with an optimal dummy node $v$ and let $\bar{V} = V(\bar{\pi})$ be the optimal value. In this tour, the travel from location $v_0$ to $v_s$ proceeds first from $v_0$ to $v$ and then from $v$ to $v_s$. To avoid excessive computations related to optimal choice of the dummy node $v$, we use a relaxation of travelling directly from $v_0$ to $v_s$, for all $s$. The shortcuts from $v_0$ to $v_s$ in $\bar{\pi}$ lead to a feasible tour $\pi'$ for our relaxed problem with a value $V(\pi') \geq \bar{V}$. If $V^*$ is the optimal value for the relaxed problem, then $V^* \geq V(\pi') \geq \bar{V}$ and $\Delta V = V^* - \bar{V} \geq 0$ is the relaxation error.

Our relaxation is justified as follows. (i) In an optimal tour $\pi$ of the relaxed SP the last node $v_0$ in $T_0$ often is at the end of $T_0$, and $v = v_0$ is optimal for the non-relaxed problem as well. (ii) Similarly, if the node $v_0$ in the optimal tour $\pi$ of the relaxed SP is such that the surveillance boat can wait at $v_0$ until the end of $T_0$, and thereafter there is sufficient time to reach each node $v_s$, then $v = v_0$ is optimal for the non-relaxed problem too. (iii) In other cases the relaxation error is expected to be small because the time slot of visiting $v_0$ in real-world problems with many targets is likely to be close to the end of $T_0$. (iv) At the end of Section 5 we estimate the relaxation error in 153 case study problems; on average the error is found negligible. (v) We also explore the extra computational effort due to optimizing the location of the dummy node. Using a grid of 1320 alternative dummy nodes covering the work area, the ILP approach is intractable for all but the smallest problem instances with time horizon $|T| = 4$ hours. For *RDP* employing the same grid for dummy nodes, the computing time for $|T| = 4, \ldots, 12$ hours and $M = 6, \ldots, 100$ increases by a factor ranging from 4 to 12 depending on $T$ and $M$.

### 2.3. ILP model for two-stage stochastic programming

For the formulation of the two-stage problem of Section 2.2 as an ILP problem, we define the set of binary variables by admissible arcs defined by (3)–(5). Hence, in stage $T_0$, for all $(ik, jl) \in \Xi$ we have a binary variable $x_{ik}^{jl}$, in stage $T_1$ for scenario $s \in C$ and for all $(ik, jl) \in \Xi(s)$, we have a binary variable $x(s)_{ik}^{jl}$, and for transition from $T_0$ to scenario $s \in C$, for all $(ik, jl) \in \Xi'(s)$ we have a binary variable $x'(s)_{ik}^{jl}$. The value of a binary variable is 1 if and only if the corresponding arc is included in the tour. For summation over binary variables it is convenient to use the dot notation: a dot replacing an index means summation over the index it replaces. For example, for binary variables $x_{ik}^{jl}$ for stage $T_0$, $x_{ik}^{\bullet\bullet} = \sum_{jl} x_{ik}^{jl}$ where the set of pairs of indices $jl$ is defined by the requirement $(ik, jl) \in \Xi$. Similarly, dot notation applies to binary variables $x(s)_{ik}^{jl}$ and $x'(s)_{ik}^{jl}$.

As mentioned above, we assume that the optimal plan over $T = [t_0, t_2]$ is such that the number of visits during $T_0 = [t_0, t_1]$ is positive. The following constraint ensures exactly one departure from depot $i = 0$:

$$x_{0\bullet}^{\bullet\bullet} = 1 \quad (7)$$

Given that exactly one tour is chosen for each scenario $s$, there is exactly one arc $(ik, 0l)$ chosen to enter the depot for each scenario. This is achieved by the following terminal constraint:

$$x'(s)_{\bullet\bullet}^{0m} + x(s)_{\bullet\bullet}^{0m} = 1 \quad \forall\ s \in C. \quad (8)$$

If the first component in the left side of Eq. (8) is equal to 1 for a scenario $s$, then there are no visits to targets during $T_1$ in that scenario.

We define the intermediate flow constraints in $T_1 = [t_1, t_2]$ for each scenario $s$ and nodes $v(s)_{ik}$ as follows:

$$x'(s)^{ik}_{\bullet\bullet} + x(s)^{ik}_{\bullet\bullet} = x(s)^{\bullet\bullet}_{ik} \quad \forall\ i \in N_s,\ k \in S_{si},\ s \in C. \tag{9}$$

Here the right side is always 0 or 1 by (8) and (9). In case it is 1, there are two possibilities for each scenario $s$; either (i) node $v(s)_{ik}$ is the first node visited when entering scenario $s$ in period $T_1$ (then the first component on the left side is equal to 1 so the second component must be 0) or (ii) node $v(s)_{ik}$ is not the first node visited during the period $T_1$ (then the second component on the left side is 1 and the first is 0).

In $T_0 = [t_0, t_1]$, the intermediate flow constraints for node $v_{ik},\ i \in N_0,\ k \in S_i\ (0 < k \le k^*)$ is as follows: the immediate predecessor target $j \in N_0 \cup \{0\}$ is visited at time slot $l < k$ with $l \ge 0$ and $j \ne i$, and for the immediate successor target, either (i) there is a $j \in N_0$ visited at time slot $l > k$ with $l \in S_j$ and $j \ne i$, or (ii) for all $s \in C$, there is $j \in N_s \cup \{0\}$ visited at time slot $l \in S_{sj}$ and $j \ne i$. This is ensured by the following flow conservation constraints:

$$x^{ik}_{\bullet\bullet} = x^{\bullet\bullet}_{ik} + x'(s)^{\bullet\bullet}_{ik} \qquad \forall\ i \in N_0,\ k \in S_i,\ s \in C. \tag{10}$$

Here the left side is always equal to 0 or 1 by (7) and (10). If the left side is 1, then either (i) node $v_{ik}$ is not the last node visited during $T_0$, the first component on the right side is 1 and the second component on the right side must be 0 or (ii) node $v_{ik}$ is the last node visited during $T_0$, the first component on the right is 0 and the second component $x'(s)^{\bullet\bullet}_{ik}$ must be 1 for all scenarios $s \in C$.

We also guarantee that each target $i$ is visited at most once by the following inequality which should hold for each $i \in N$ and for each scenario $s \in C$:

$$x^{i\bullet}_{\bullet\bullet} + x(s)^{\bullet\bullet}_{i\bullet} \le 1 \quad \forall i \in N,\ s \in C. \tag{11}$$

Here the first component counts arrivals to target $i \in N_0$ (0 if $i \notin N_0$) the second component counts departures from target $i$ in scenario $s$ (0 if $i \notin N_s$). Note that if the first component on the left side of (11) is 1, then target $i$ is only visited during $T_0$; otherwise either target $i$ is visited later during $T_1$ in the scenario $s$, so that the second component is equal to 1, or target $i$ is left non-visited, in which case both components are zero.

The objective is to maximize the expected value of the linear value function with equal scenario probabilities as follows:

$$\text{Maximize } V = \lambda\alpha_0 - z_0 + \frac{1}{M}\sum_{s\in C}(\lambda\alpha_s - z_s) \tag{12}$$

where $\lambda$ is a weighting parameter reflecting the decision maker's trade-off preferences. The component $\alpha_0$ is the number of targets visited during $T_0$. It counts arrivals to all targets $i \in N_0$ as follows

$$\alpha_0 = x^{\bullet\bullet}_{\bullet\bullet} \tag{13}$$

The component $\alpha_s$ is the number of targets visited in scenario $s$ during $T_1$,

$$\alpha_s = x(s)^{\bullet\bullet}_{\bullet\bullet} \quad \forall\ s \in C \tag{14}$$

counting departures from all targets $i \in N_s$. The component $z_0$ is the travel distance in the deterministic period $T_0$,

$$z_0 = \sum_{(ik,jl)\in\Xi} d^{jl}_{ik} x^{jl}_{ik} \tag{15}$$

and finally the travel distance $z_s$ for scenario $s$ from the transition node to depot node during $T_1$ is as follows:

$$z_s = \sum_{(ik,jl)\in\Xi'(s)} d'(s)^{jl}_{ik} x'(s)^{jl}_{ik} + \sum_{(ik,jl)\in\Xi(s)} d(s)^{jl}_{ik} x(s)^{jl}_{ik}. \tag{16}$$

## 3. RDP for the deterministic case

In this section, we consider the deterministic problem of Section 2.1. We begin by discussing some DP-based concepts, and thereafter, we present the basic steps of the *RDP* algorithm for which the supplementary Appendix B introduces optional improvement steps. Notations and concepts used in *RDP* for the deterministic case are subsequently adopted in Section 4 to the two-stage SP model as well.

### 3.1. Some DP considerations

Based on the classical DP formulations for TSP (Bellman, 1962; Held & Karp, 1962), our problem is in theory solvable using DP in a network specified as follows. Let $(ik, I)$ be a node (state) where ship $i$ in time slot $k$ is in the location given by coordinate vector $v_{ik}$, and $I$ is the set of ships left for possible later visits. Initially, at the root node $ik = 00$ and $I = N$. At the end of $T$, $ik = 0m$ and $I$ is an empty set or a set of ships not to be visited at all during $T$. Consider an arc, a transition, from node $(ik, I)$ to node $(jl, J)$. If travelling from node $v_{ik}$ to node $v_{jl}$ is admissible then we have $(ij, kl) \in \Xi$ in (1) and $J = I\backslash\{i\}$. Using DP backward recursion is straightforward. However, it is well known that such DP for an ordinary TSP (Applegate, Bixby, Chvâtal, & Cook, 2006) is intractable for all but the smallest instances, and alternative time slots $k$ of the nodes of each ship $i$ makes the situation even worse.

In the DP formulation above, the role of the set $I$ at node $(ik, I)$ is to ensure that each ship $i$ can be visited at most once. We deal with this condition differently. Our network is defined by nodes $v_{ik}$, and a transition from node $v_{ik}$ to node $v_{jl}$ is admissible if the arc $(ik, jl) \in \Xi$. The initial node is $v_{00}$ and the terminal node is $v_{0m}$. In this network, we may carry out a standard DP backward recursion steps employing the linear value function as follows. Let $V_{ik}$ denote the value at node $v_{ik}$ based on the chosen sub-tour from $v_{ik}$ to $v_{0m}$. At the terminal node, define $V_{0m} = 0$. Working backwards for $k = m-1, m-2, \ldots, 0$, at time stage $k < m$ we assume the values $V_{jl}$ have been evaluated for all $j$ and $l > k$ (such that node $v_{lj}$ is in the work area), and additionally, we have recorded the set of ships $I_{jl}$ which have been chosen for a visit in a sub-tour starting from node $v_{jl}$ and ending at $v_{0m}$. When a transition is chosen at node $v_{ik}$ to an admissible node $v_{jl}$, we require $i \notin I_{jl}$. The successor node chosen by the standard DP rule maximizes the value $V_{ik}$. At the root node $V_{00}$ is the value of the feasible tour produced by standard DP.

The latter DP formulation shows how a feasible tour for (2) can be found. However, standard DP generally does not produce a global optimum for our routing problem (2). The reason is that the conditions for applying DP are violated: while choosing a successor node at a node $v_{ik}$, if node $v_{jl}$ is selected, it is possible that ship $j$ in the global optimum is scheduled for a visit before time slot $k$ (not after as DP would suggest). Thus, the violation of DP conditions is that the best choice at a node $v_{ik}$ can depend on choices preceding time slot $k$. This applies to both of our deterministic and stochastic cases. In the case study of Section 4.3 we show the significant average loss of optimality while applying the standard DP backward recursion for producing feasible solutions.

### 3.2. Randomized dynamic programming (RDP)

To avoid the drawback with standard DP, in RDP we use randomization and iterations to achieve asymptotic convergence to some optimal solution. In each iteration $\tau$ of RDP we perform (i) a backward recursion *Br*, (ii) a forward recursion *Fr* and (iii) optional improvement steps (discussed in Appendix B).Recursion *Br* (*Fr*) is executed similarly as in standard DP; however, in *RDP* at node $v_{ik}$ the choice of a successor (predecessor) node is based on

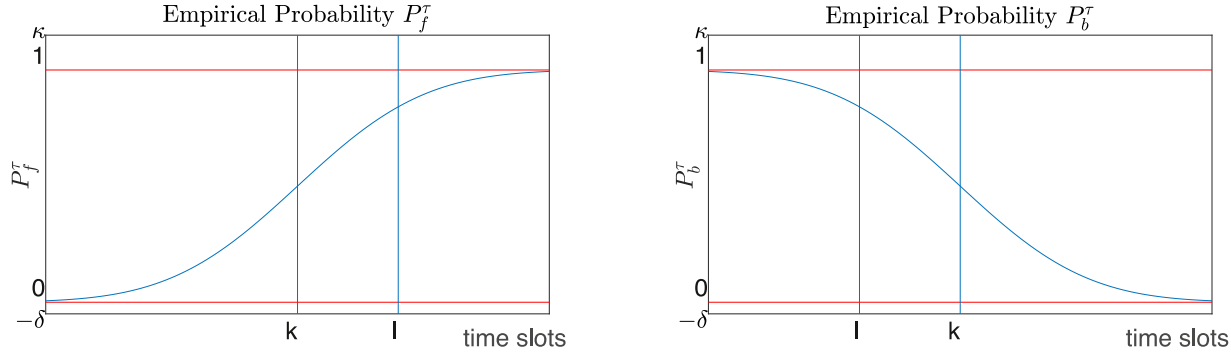


**Fig. 2.** Empirical probabilities $P_f^\tau(j,k)$ (left) and $P_b^\tau(j,k)$ (right) for a ship $j$ are represented schematically over time horizon $k = 0,\ldots,m$ (after a number of iterations). At node $v_{ik}$ in *Br* (*Fr*), a feasible node $v_{jl}$ with $l > k$ ($l < k$) is accepted as a candidate if $r_j > P_f(j,k)$ ($r_j > P_b(j,k)$) for a random draw $r_j$ from U(-$\delta$, $\kappa$).

maximization of the value function over a *set of candidate nodes* which is a *randomly drawn subset of feasible nodes* used in the standard DP.

At the end of iteration $\tau$, $\pi^\tau$ denotes the best tour produced in iteration $\tau$ resulting from *Br*, *Fr* or from the improvement steps; $\hat{\pi}^\tau$ is the best tour found in iterations $1,2,\ldots,\tau$ and $\hat{V}^\tau = V(\hat{\pi}^\tau)$ is its respective value. In the sequel, for other notation we frequently suppress $\tau$. For instance, $V_{ik}$ and $I_{ik}$ in iteration $\tau$ stand for the value and the set of ships, respectively, in the sub-tour chosen in *Br* from node $v_{ik}$ to $v_{0m}$; similarly in *Fr*, $V'_{ik}$ is the value and $I'_{ik}$ is the set of ships in the sub-tour from the root node $v_{00}$ to $v_{ik}$. Due to randomization, these values and sets vary over iterations $\tau$.

**i) Backward recursion:** For *Br* in iteration $\tau$, while choosing a successor node at $v_{ik}$, for $1 < k < m$ and $k \in S_i$, the depot node $v_{0m}$ is always accepted as a candidate provided it is admissible. For ships $j \neq 0$, we use randomization employing parameter $P_f(j,k)$ which is *an empirical probability for ship $j$ to be visited at time slot $k$ or earlier*. Let $\phi_{ik} = \{v_{jl} |\ (ik, jl) \in \Xi,\ i \notin I_{jl}\}$ denote the *set of feasible successor nodes* at node $v_{ik}$. The *set of feasible targets* is $J_{ik} = \{j | v_{jl} \in \phi_{ik}$ for some $l\}$. At node $v_{ik}$, for each $j \in J_{ik}$, nodes $v_{jl} \in \phi_{ik}$ are accepted as *candidates* if $r_j > P_f(j,k)$, where $r_j$ is an independent random draw from the uniform distribution $U(-\delta, \kappa)$ and $\delta > 0$, $\kappa > 1$ are pre-specified parameters;[4] otherwise nodes $v_{jl}$ of ship $j$ are rejected. Intuitively, a high probability $P_f(j,k)$ means a small chance for ship $j$ appearing in an optimal sequence after time slot $k$ and nodes $v_{jl}$ with $l > k$ are likely to be rejected; for an illustration, see the left curve in Fig. 2. At node $v_{ik}$, randomization further restricts the choice of a successor node $v_{jl}$, in addition to the requirements $(ik, jl) \in \Xi$ and $i \notin I_{jl}$. For all $i$, the distribution $P_f$ is updated in each iteration $\tau$ based on tours from *Fr*; we discuss such updating shortly.

Formally, the backward recursion *Br* begins at terminal node $v_{0m}$, and ends at the initial node $v_{00}$. For node $v_{0m}$, the value is $V_{0m} = 0$ and the set $I_{0m}$ is empty. For time slots $k = m-1,\ldots,1$ and $i \in N$ such that $k \in S_i$, we carry out the random selection of the candidate nodes to succeed node $v_{ik}$. From the set of candidates, we choose the one which yields the maximum value $V_{ik}$. If a node $v_{jl}$ is chosen, then the value $V_{ik}$ is obtained by adding an immediate reward to $V_{jl}$ as follows

$$V_{ik} = \lambda - d_{ik}^{jl} + V_{jl} \quad \text{and} \quad I_{ik} = I_{jl} \cup \{i\}. \tag{17}$$

If no candidate is found, then $V_{ik} = -\infty$ and $I_{ik} = \{i\}$. At the starting node $v_{00}$, all admissible nodes $v_{jl}$ are eligible candidates. Given the best admissible node $v_{jl}$, we obtain

$$V_{00} = -d_{00}^{jl} + V_{jl} \quad \text{and} \quad I_{00} = I_{jl}. \tag{18}$$

[4] For most empirical tests in Section 5, we use $\delta = 0.01$ and $\kappa = 1.2$.

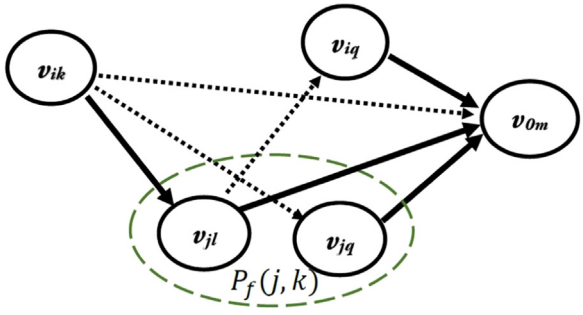


**Fig. 3.** An illustration of a *Br* recursion. Consider successive time slots $k$, $l$, $q$ and $m$, s.t. $0 < k < l < q < m$, and ships $i$ and $j$. Thick edges are chosen by *Br* and dashed edges are other feasible edges. For the choice at node $v_{ik}$, there were *three* feasible successor nodes $v_{jl}$, $v_{jq}$ and $v_{0m}$. Nodes $v_{jl}$ and $v_{jq}$ belong to a cluster of ship $j$ shown in the dashed circle. While the depot $v_{0m}$ is automatically a candidate node, the cluster is subject to randomization. Based on the probability $P_f(j,k)$, a single random draw determines whether the nodes in the cluster are accepted as candidates. In this case, the draw was favorable and the node $v_{jl}$ was the best choice at node $v_{ik}$. - If the dashed edge joining $v_{jl}$ and $v_{iq}$ had been chosen at $v_{jl}$, then $v_{jl}$ would not be a feasible choice at $v_{ik}$, because ship $i$ would be in the sub-tour starting at $v_{jl}$.

In *Br* of iteration $\tau$, for each node $v_{ik}$, we record the value $V_{ik}$, the chosen successor node $v_{jl}$ as well as the set of targets $I_{ik}$. The best tour from *Br* in iteration $\tau$ is denoted by $\pi_b$ for which the value $V(\pi_b)$ is given by (18). An illustration of *Br* is shown in Fig. 3.

**ii) Forward recursion:** The forward recursion *Fr* of *RDP* is a mirror image of *Br*; replacing in Fig. 3 the terminal node $v_{0m}$ by the root node $v_{00}$ and $P_f(j,k)$ by $P_b(j,k)$, the figure illustrates *Fr* with $0 < q < l < k < m$. Random choices in *Fr* employ parameters $P_b(j,k)$ which yield *an empirical probability for targets $j$ appearing in the itinerary at time slot $k$ or later*. The probabilities $P_b$ are obtained from tours produced by *Br*. For each node $v_{ik}$, the value $V'_{ik}$ in *Fr* is obtained from the sub-tour from the root node $v_{00}$ to $v_{ik}$ and $I'_{ik}$ is the set of ships in this sub-tour. At the root node $v_{00}$, $V'_{00} = 0$ and $I'_{00}$ is empty. For $k = 1,2,\ldots,m-1$ and $i$ such that $k \in S_i$, the set of admissible arcs entering node $v_{ik}$ are $(jl, ik) \in \Xi$. At node $v_{ik}$, the depot node $v_{00}$ is always accepted as a candidate provided it is admissible. For ships $j \neq 0$, candidate nodes are obtained by random draws using empirical distributions $P_b$ and employing the same principles used in *Br*. Hence, the larger is $P_b(j,k)$ the smaller is the chance of node $v_{jl}$ being chosen as a predecessor node at $v_{ik}$; see the right curve in Fig. 2. The best candidate maximizes the value $V'_{ik}$ at node $v_{ik}$. For $k = m$, if $(jl, 0m) \in \Xi$ then $v_{jl}$ is a candidate node at $v_{0m}$. The best tour from *Fr* in iteration $\tau$ is denoted by $\pi_f$. If $V(\pi_f) > V(\pi_b)$, then replace $\pi^\tau$ by $\pi_f$.

**iii) Empirical probability distributions:** The empirical probability distributions $P_b$ and $P_f$ are based on best tours produced by

*Br* and *Fr*, respectively, in iterations $1, 2, \ldots, \tau$. Thus, *Br* borrows data from *Fr* and vice versa. In Section 5.3, we test empirically that such interaction of *Br* and *Fr* indeed pays off. Next, we give the formal definition of the empirical probabilities.

For $P_f$, initially we set $P_f(i,k) = 0$ for $k < m$ and $P_f(i,m) = 1$, for all $i \in N$. In each iteration $\tau$, if tour $\pi_f$ is the best one found by *Fr*, then another tour $\pi$ from *Fr* in iteration $\tau$ is $\eta$-optimal if $V(\pi_f) - V(\pi) \le \eta$, where $\eta > 0$ is a given tolerance. Distributions $P_f$ are updated based on the set of $\eta$-optimal tours $\Pi_f^\tau$ generated by *Fr*. The cumulative statistics of $\nu_f(i,k)$ (suppressing $\tau$) counts the number of times ship $i$ is visited in time slot $k$ taking into account all tours in $\Pi_f^{\tau'}$ for all $\tau' \le \tau$. At the end of iteration $\tau$, if $\sum_l \nu_f(i,l) \neq 0$ the updated empirical probabilities for $i$ and for all $k$ are given by $P_f(i,k) = \sum_{l \le k} \nu_f(i,l) / \sum_l \nu_f(i,l)$; otherwise, $P_f(i,k) = 0$ for all $k < m$ and $P_f(i,m) = 1$. For $P_b$ similarly, if $\nu_b(i,k)$ counts the visits to ship $i$ in time slot $k$ taking into account $\eta$-optimal tours produced by *Br*, then $P_b(i,k) = \sum_{l \ge k} \nu_b(i,l) / \sum_l \nu_b(i,l)$. For all $i$, we have $P_f(i,k)$ increasing and $P_b(i,k)$ decreasing in $k$ over iterations.

**iv) Optional improvements:** Given $\pi^\tau$, the best of the tours produced by *Br* and *Fr* in iteration $\tau$, we may carry out improvement updates on $\pi^\tau$; see Appendix B. At the end of iteration $\tau$, we observe $\hat{\pi}^\tau$, the best tour found over all iterations $1, 2, \ldots, \tau$.

To summarize, the schematic picture of *RDP* for solving the deterministic problem (2) is given by the following steps:

- *Initialization*. Let $\hat{V}^0 = -\infty$; for all $i$, $P_f(i,k) = 0$ for $k < m$ and $P_f(i,m) = 1$, $P_b(i,k) = 0$ for $k > 0$ and $P_b(i,0) = 1$; choose parameters $\delta > 0$, $\kappa > 1$, tolerance $\eta > 0$, iterations limit $\bar{\tau}$, and set $\tau = 1$.
- *Br: Backward recursion*.
  - For all $k = m-1, \ldots, 0$ and $i \in N$ such that $k \in S_i$,
    - At node $v_{ik}$, choose the best successor node based on the randomly drawn candidates.
    - Record the backward value $V_{ik}$, the chosen successor node and the set $I_{ik}$.
- ◦ Let $\pi_b$ denote the best tour from *Br* in iteration $\tau$; let $\pi^\tau = \pi_b$.
- ◦ Using $\eta$-optimal tours from *Br* update the distribution $P_b(i,\cdot)$ for all $i$.
- *Fr: Forward recursion*.
  - For all $k = 1, \ldots, m$ and $i \in N$ such that $k \in S_i$,
    - At node $v_{ik}$, choose the best predecessor node based on the randomly drawn candidates.
    - Record the forward value $V'_{ik}$, the chosen predecessor node and the set $I'_{ik}$.
- ◦ Let $\pi_f$ denote the best tour from *Fr* in iteration $\tau$; if $V(\pi_f) > V(\pi_b)$, then $\pi^\tau = \pi_f$.
- ◦ Using $\eta$-optimal tours from *Fr* update the distribution $P_f(i,\cdot)$ for all $i$.
- *Improvements*. (For optional steps, see Appendix B).
  - Optimal timing. Revise $\pi^\tau$ by finding optimal timing for the sequence $\sigma(\pi^\tau)$.
  - *Swaps*. For some pairs of targets $i, j$ in $\sigma(\pi^\tau)$,
    - Update $\pi^\tau$ if the swap $i, j$ improves the tour.
- If $V(\pi^\tau) > \hat{V}^{\tau-1}$, record $\hat{\pi}^\tau = \pi^\tau$ and $\hat{V}^\tau = V(\pi^\tau)$, the best tour found so far.
- *Termination*. If $\tau < \bar{\tau}$, increment $\tau$ by 1 and return to step *Br*; otherwise, stop.

Next, letting the iterations limit $\bar{\tau}$ increase without limit, we state the asymptotic convergence result for *RDP* in the deterministic case;for the two-stage SP problem in Section 4 the result is omitted for brevity, since the same convergence arguments apply in both cases. The proof is in the supplementary Appendix A.

**Theorem 1.** *Assume an optimal tour $\pi^* = \{v^*_{ik}\}$ exists for problem (2) with $\alpha^* = \alpha(\pi^*)$ targets to be visited and with an optimal value of $V^* = V(\pi^*)$. Given $\kappa > 1$ and $\delta > 0$, let $\hat{\pi}^\tau = \{\hat{v}^\tau_{ik}\}$ with $\hat{V}^\tau = V(\hat{\pi}^\tau)$ denote the best tour found by the end of iteration $\tau$ of RDP. Then $\hat{\pi}^\tau$ converges with probability one to some optimal tour $\hat{\pi}$ with value $V(\hat{\pi}) = V^*$.*

## 4. RDP for two-stage stochastic programming

Next, we explain how *RDP* presented in Section 3 is adopted for solving the two-stage SP of the optimal routing in Section 2.2. The implementation of *RDP* for *SP* is as follows:

Again, each iteration $\tau$ of *RDP* involves *Br* and *Fr* recursions as well as optional improvements. Similarly, $\hat{\pi}^\tau$ denotes the best tour found by the end of iteration $\tau$. We avoid repetition of details of *RDP* which are fully explained in Sections 3, and describe the extensions thematically. For randomization, for instance, we only discuss the definition of the empirical probabilities; random choice of the candidate nodes is done as explained in Section 3.

- *Backward recursion Br:*
  - In the second stage, for each $s \in C$, $k = m, \ldots, k^* + 1$, $i \in N_s$ with $k \in S_{si}$, and nodes $v(s)_{ik}$, the *Br* recursion is as defined in Section 3. The value is $V(s)_{ik}$ and the set of targets is $I(s)_{ik}$ in the chosen sub-tour from node $v(s)_{ik}$ to $v(s)_{0m}$.
  - In the first stage, for $k = k^*, \ldots, 1$, $i \in N_0$ with $k \in S_i$, and nodes $v_{ik}$, we consider two cases. First, we find the best successor candidate node $v_{jl}$ in $T_0$ (as in Section 3) and denote the resulting value at $v_{ik}$ by $V^0_{ik}$. Second, for each $s \in C$, we find the best successor candidate $v(s)_{jl}$ in scenario $s$ (as in Section 3), denote the resulting value at $v_{ik}$ by $V(s)_{ik}$, and obtain the expected value $V^1_{ik} = (1/M)\sum_s V(s)_{ik}$. If $V^0_{ik} < V^1_{ik}$, then the tour transits from node $v_{ik}$ to the second stage $T_1$, the value at $v_{ik}$ is $V_{ik} = V^1_{ik}$ and $I_{ik} = (\cup_{s \in C} I(s)_{jl}) \cup \{i\}$ where the pairs $jl$ refer to best choices in each scenario $s$. If $V^0_{ik} \ge V^1_{ik}$, then the value at $v_{ik}$ is $V_{ik} = V^0_{ik}$ and $I_{ik} = I_{jl} \cup \{i\}$.
  - For the starting node $v_{00}$, assuming there is at least one visit in $T_0$, we only consider the successor nodes $v_{jl}$ in $T_0$ (without randomization). Thereafter, we are ready to recover the tour $\pi_b$ from *Br* and its value $V(\pi_b) = V_{00}$.
- *Forward recursion Fr:*
  - In the first stage, for $k = 1, \ldots, k^*$, $i \in N_0$ with $k \in S_i$, and node $v_{ik}$ we proceed as in *Fr* recursion of Section 3 to obtain value $V'_{ik}$ and the set $I'_{ik}$ of targets in the sub-tour from node $v_{ik}$ backwards to $v_{00}$.
  - In the second stage, for all $s \in C$, $k = k^* + 1, \ldots, m$, $i \in N_s$ with $k \in S_{si}$, and node $v(s)_{ik}$, we also proceed as in *Fr* of Section 3. As predecessor node candidates, we consider both nodes $v(s)_{jl}$ in scenario $s$ and nodes $v_{jl}$ in the first stage $T_0$. The best choice determines the value $V'(s)_{ik}$ and the set $I'(s)_{ik}$ of targets in the sub-tour from the node $v(s)_{ik}$ in scenario $s$ backwards to the root node $v_{00}$.
  - If the transition node in $T_0$ is not uniquely determined (but depends on the scenario) in a tour $\pi$, then $\pi$ is not feasible. In case of infeasibility, we choose a single transition node $v_{\tilde{i}\tilde{k}}$ for all scenarios based on the best value $V'_{ik}$ among the nodes $v_{ik}$ which served as transition nodes to different scenarios. Thereafter, the steps of *Fr* are repeated for all $s \in C$ but considering $v_{\tilde{i}\tilde{k}}$ as the only predecessor candidate in $T_0$. In case no feasible solution arises for scenario $s$, we define the value $V'(s)_{0m} = -\infty$.
  - For a tour $\pi_f$ from *Fr*, the expected value is $V(\pi_f) = (1/M)\sum_s V'(s)_{0m}$.

- In iteration $\tau$, for the tour $\pi^\tau$ (the better one among $\pi_b$ and $\pi_f$), the improvement steps in Appendix Bare adopted separately for the deterministic stage $T_0$ and for each scenario in the stage $T_1$. That is, in $T_0$ the starting and ending nodes $v_{00}$ and $v_{\hat{i}\hat{k}}$ are considered frozen (unchanged in location and time). Similarly for each scenario $s$ in $T_1$, the starting (transition) node $v_{\hat{i}\hat{k}}$ and ending nodes $v_{0m}(s)$ are frozen. Thus the transition node $(\hat{i},\hat{k})$ is not subject to changes in improvements. For numerical tests in Section 5, using a tolerance $\eta > 0$, we carry out the improvement steps in iteration $\tau$ only if $V(\pi^\tau) \geq \hat{\pi}^{\tau-1} - \eta$; i.e., if the best tour $\pi^\tau$ from *Br* and *Fr* appears promising.
- To define the empirical probabilities $P_b(i,k)$ (for $i \in N_0$ and $k \leq k^*$) and $P_b(s)(i,k)$ (for all $s \in C$, $i \in N_s$ and $k > k^*$) used by *Fr* for randomization, let $\Pi_b^\tau$ be the set $\eta$-optimal tours $\pi$ generated by *Br* in iteration $\tau$. For all $\pi \in \Pi_b^\tau$ and $s \in C$, let $\pi_s$ denote a *path* in $\pi$ connecting the terminal node $v(s)_{0m}$ of scenario $s$ to the root node $v_{00}$. The cumulative statistics of $v_b(s)(i,k)$ (suppressing $\tau$) counts the number of times ship $i$ is visited in time slot $k$ in path $\pi_s$ of $\pi$ taking into account all tours in $\pi \in \Pi_b^{\tau'}$ for all $\tau' \leq \tau$. At the end of iteration $\tau$, if $\sum_k v_b(s)(i,k) \neq 0$ then the empirical probabilities for $i \in N$, and for all $k \in \{0,1,\ldots,m\}$ and $s \in C$ are given by $P_b(s)(i,k) = \sum_{l \geq k} v_b(s)(i,l) / \sum_l v_b(s)(i,l)$. Thereafter, for the deterministic period $T_0$, the probabilities $P_b(i,k)$, for all $i \in N_0$ and $k \leq k^*$, are averages over scenarios; i.e., $P_b(i,k) = 1/M \sum_{s \in C} P_b(s)(i,k)$. - Similarly, we define the empirical probabilities $P_f(i,k)$ and $P_f(s)(i,k)$, for all $s$, $i$ and $k$ based on sets of $\eta$-optimal paths $\pi_s$ produced by *Fr* in iterations $\tau$.

## 5. Experimental results

In this section we discuss computational results using the *RDP* procedure of Section 4 on real-world multi-scenario data sets. First, in Section 5.1 we present test setting and data related to seven days of our case study. Then in Section 5.2 we evaluate the performance of *RDP* compared with ILP solutions based on model (7)–(16) in Section 2.3 for the stochastic optimal routing in a dynamic network.[5] All implementations are done in AMPL environment with Mosek version 9.2 as an IP solver in a standard HP-Z230 work station (4 threads, 3.4 GHz, 8 Gb RAM) operating under Linux. Finally, in Section 5.3 we discuss diverse topics, including the role of interaction of *Br* and *Fr* in *RDP*, the impact of increasing the number of scenarios, and the value of stochastic solution.

### 5.1. Data and instances for numerical test

The trajectory prediction model described in the supplementary Appendix C is trained using historical data gathered during May–July, 2018. The model is used to estimate trajectories of the ships present in the work area over a 12-hour time horizon 7 am–7 pm during one week of August 6–12, 2018. Each predicted trajectory is interpolated with $w = 5$ minutes time spacing. We use the predicted data separately for time horizons $T = 4, 6, 8, 10$ and 12 hours. We set the duration of deterministic stage $T_0$ to 3 hours and range the duration of stochastic stage $T_1$ over the set 1,3,5,7,9 hours.

To evaluate the performance of *RDP* against ILP solution on a range of medium to large size data sets, we made 6 batches of instances each containing 35 problems (totally 210 problems). The 35 instances relate to estimated trajectories of the 7 days with 5 values for the length of time horizon $T$. To create the instances we had 2 hyper-parameters to tweak, *number of scenarios* (M) and the weighting parameter $\lambda$ of the value function (6). Each batch of instances is solved for a pair of parameters in the ranges $M = 6, 10$ and $\lambda = 5, 10, 30$. Setting big $\lambda$ values leads to longer sequences of ships to be visited, and thus a more complex problem.

For scenario generation we employ the approach presented in the supplementary Appendix C. Model estimation is based on (C.2) using data of August, 2018. An even number of trajectories (scenarios) are generated independently for each ship using (C.6) with antithetic sampling. Each scenario of ship $i$ starts from the last node $v_{ik^*}$ of the predicted (or realized) trajectory in the first stage $T_0$. The parts of trajectories which fall outside the work area are omitted. Figure 5 illustrates two examples with predicted trajectories shown by solid red lines and four scenarios by dashed lines. In Fig. 5, ship 6 which is travelling toward northeast, is not in the work area during $T_0$. One of the four scenarios falls completely outside the work area and thus, it is omitted.[6] Ship 7 is travelling toward southwest, it is in the work area at the end of $T_0$ and parts of all four scenarios intersect the work area.

For time discretization, the length of a time slot is $w = 5/60$ hours and the number of time slots is $m = |T|/w = 48, 72, \ldots, 144$ for $|T| = 4, 6, \ldots, 12$ hours. The processing time by the surveillance boat is 3/60 hours and the speed of the boat is $c = 46.3$ kilometers per hour. For randomization in *RDP*, the values of parameters $\delta$ and $\kappa$ are determined using a grid search over the search space of $\{0, .02, .04, \ldots, 0.10\}$ for $\delta$ and $\{1.1, 1.2, 1.3\}$ for $\kappa$, and set to $\delta = 0.02$ and $\kappa = 1.20$. In each iteration of *RDP*, if the better tour resulting from *Br* and *Fr* is within a 10 kilometers tolerance as good as or better than the best value encountered by the current iteration, then the improvements of Appendix B are applied separately for the deterministic stage $T_0$ as well as for each scenario in $T_1$.

Using (7)–(16) we formulated 210 instances (6 batches of $7 \times 5$ problems related to $M = 6, 10$ scenarios and $\lambda = 5, 10, 30$) as ILP problems, a large share of which could be solved by a standard solver in a reasonable time. The dimension of ILP instances for $M = 10$ are summarized in Fig. 4 in terms of the number of binary variables and constraints of the respective ILP models. From Fig. 4 we can see even for the number of scenarios as few as 10, the ILP problem can be huge. The biggest size problems are related to $|T| \geq 10$ for August 9 and 12 data with more than one million binary variables and thousands of constraints. We will show in the next sub-section that *RDP* method can perform efficiently on such large dimensional problems in terms of accuracy and run time. For the number of scenarios 20 or more, the ILP solver is mostly incapable of finding a solution in a reasonable time; therefore, such instances are discussed below separately without ILP solutions.

### 5.2. Performance of RDP

We solved the 210 numerical examples by both ILP and *RDP* methods. All ILP problems are solved using Mosek solver with 1% relative gap tolerance and 2 hours time limit[7] along with other default settings. We use plain Mosek simply to enhance transparency in comparing *RDP* with an ILP solver. The iterations limit for *RDP*

[5] The comparison with an exact solver is solely for evaluating the quality of solutions obtained by *RDP* in a reasonable time for different size of instances. To the best of our knowledge, there were no alternative algorithm to be used for comparison. Using one of the many existing heuristic approaches poses the difficulty that problem formulations differ from our deterministic model version; see our discussion in the Introduction.

[6] In practice, it is possible that some ships are predicted to pass through the work area, but they do not appear, or their realized trajectories fall partly or completely outside the area. In this situations, the target cannot be visited at those locations, and thus the location points could be omitted.

[7] In a practical setting, the surveillance-boat operator is expected to have at most 2 hours to find a reasonably good schedule, before launching the trip from the harbor; therefore, we limit our solver time to two hours. For 1% of problem instances, we increase the time limit to 10 hours because no feasible solution was found in 2 hours.

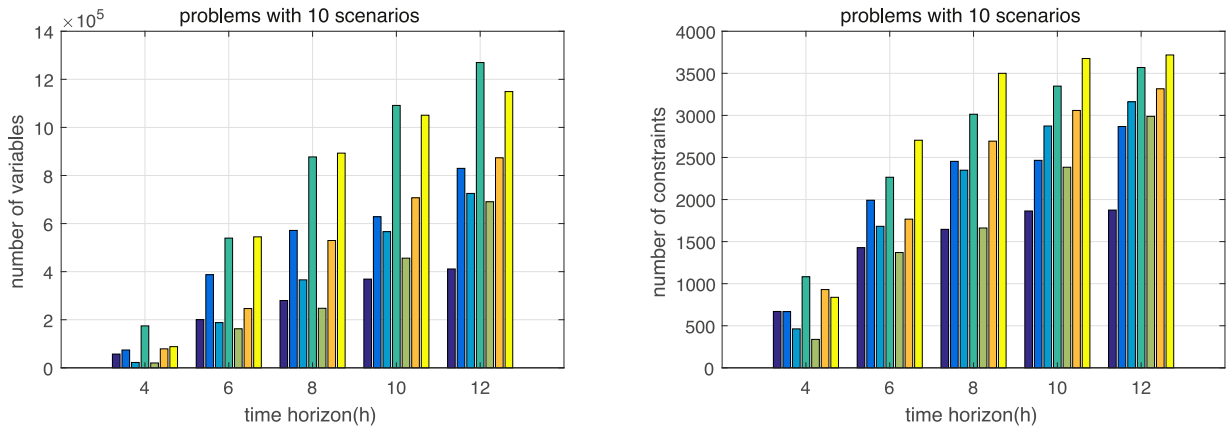


**Fig. 4.** Dimensions of the test problems for $M = 10$ in terms of number of binary variables and constraints, with 35 model configurations using each pair of hyper-parameters, horizon $T$ and the days: the five groups of problems relate to $|T| = 4, 6, 8, 10, 12$, each containing seven problems related to August 6–12, 2018.

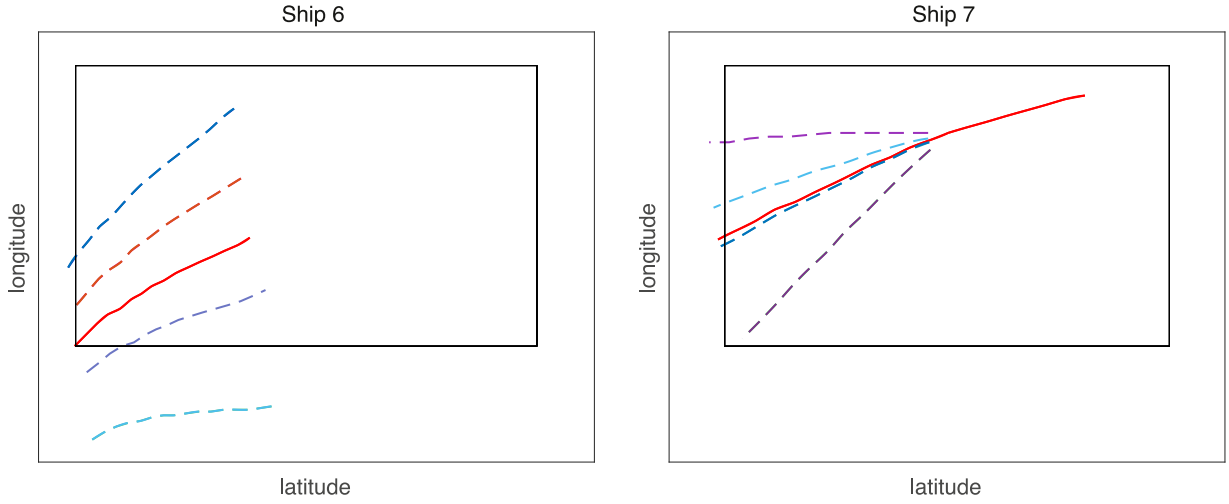


**Fig. 5.** Representation of trajectories related to August 7 for ships 6 and 7. Ship 6 (left figure) presents only during the second stage $T_1$. Ship 7 (right figure) presents during both stages $T_0$, $T_1$. Solid lines show the predicted route over part of the time horizon. In each figure, the four dashed lines show different scenarios in $T_1$. The rectangle depicts the work area. Those trajectory points which fall outside the work area are omitted from two-stage SP.

is set to 200 iterations. Table 1 reports the results. The numbers in the sub-tables show absolute errors $V_{ILP} - V_{RDP}$ (kilometer) of final *RDP* solution value compared with ILP solution, for six batches of instances with different model configurations each using a pair of parameters $\lambda, M$ from the range $\lambda = 5, 10, 30$ and $M = 6, 10$.[8] For small values $\lambda = 5$ and $M = 6$, both methods could solve all 35 instances to optimality. However, for $\lambda = 5$ and $M = 10$, ILP solver hit the 2 hours time limit in 20% of the cases (figures with '*' sign in Table 1) but provided a feasible solution. We notice that, even if the solver does not hit the time limit, the error for *RDP* can be negative (positive) meaning that the solution found by the *RDP* algorithm is more (less) efficient than the final optimal solution returned by the ILP solver. This is due to the fact that, the solver guarantees optimal value up to 1%, and *RDP* returns a higher (lower)objective value within 1% gap. By increasing the values of $\lambda$, the complexity increase because more ships will be visited. According to Table 1, for $\lambda = 10$ and $\lambda = 30$, there is an increasing number of cases with negative error values or runs hitting the time limit of ILP solution.

[8] Relative errors could be used as well. However, noting that the unit of the weighted objective $\lambda\alpha - z$ is kilometers, we are interested in the absolute difference, which is more intuitive in practical application than the relative errors. Besides, in cases where the differences are small, the relative difference is not appropriate for the actual gain or loss in the objective value.

Table 2 summarizes error statistics presented in Table 1 for the comparison of solutions found by *RDP* and ILP. The sample in *Case* 1 includes all 210 test problems. In *Case* 2 the 153 problems are those for which an optimal solution was found by ILP within 1% gap tolerance and 2 hours time limit. In *Case* 2, the mean error $-0.06$ is negative but the hypothesis of zero mean error is not rejected at a significance level 0.01; hence, setting aside the computing time taken by *RDP* and ILP, we conclude that *RDP* and ILP perform equally well for problems solvable within the 2 hours time limit. In *Case* 1 the mean error is negative with $t = 4.3e-5$ and significant at $p < 0.01$. - Results of average values of $(\alpha, z)$ over seven days obtained by *RDP* are reported for each case $\lambda, M$ and $T$ in Table 3.

In principle, we might apply some stopping criterion for *RDP* iterations, for instance, based on the improvement in a given number of iterations. However, in our tests we wanted to study the progress over a long sequence of 200 iterations. Therefore, regarding the execution time, we provide a *speed-up ratio* as a function of *iteration number* $\tau$ to measure the efficiency of *RDP* running time relative to the total execution time by the ILP solver. Given iteration $\tau$, the speed-up is the total run time of ILP solver $t_{ILP}$ divided by the run time $t_{RDP}(\tau)$ of *RDP* in $\tau$ iterations; where $t_{RDP}(\tau) = \tau \times$(time per iteration by *RDP*). The execution time (averaged over days) per *RDP* iteration for the instances with respect to each pair of parameters $T$, $M$ are shown in Table 4. Thereby,

**Table 1**
Errors for time horizon $|T| = 4, 6, 8, 10, 12$ (hour), $\lambda = 5, 10, 30$, and $M = 6, 10$ using data of August 6–12, 2018; error (kilometer) is the difference between the objective function value produced by ILP and *RDP*; $n$ = number of ships (average over seven days) in the work area during $[0, T]$; $^*$ = case where ILP solution time exceeds 2 hours limit; $\circ$ = case where ILP solution time exceeds 10 hours limit. In summary, problems are increasingly hard with large $M$ and $\lambda$ and thereby the relative merits of *RDP* increase.

| | $\|T\|=4$ | $\|T\|=6$ | $\|T\|=8$ | $\|T\|=10$ | $\|T\|=12$ | $\|T\|=4$ | $\|T\|=6$ | $\|T\|=8$ | $\|T\|=10$ | $\|T\|=12$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | 11.3 | 16.6 | 19.1 | 22.1 | 23.0 | 11.3 | 16.6 | 19.1 | 22.1 | 23.0 |
| | $M=6\lambda=5$ | | | | | $M=10\lambda=5$ | | | | |
| Aug 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | −0.01 | 0.00 | 0.00 | 0.00 |
| Aug 7 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | −0.01 | 0.00 | 0.00 |
| Aug 8 | 0.00 | −0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Aug 9 | 0.02 | 0.00 | 0.08 | 0.57 | 0.11 | 0.05 | 0.01* | 0.00* | 0.04 | 0.07* |
| Aug10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | −0.01 | 0.00 |
| Aug11 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Aug12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | −0.02* | −0.09* | −0.75* | −0.44* |
| | $M=6\lambda=10$ | | | | | $M=10\lambda=10$ | | | | |
| Aug 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.05 | 0.00* | −0.80* | −1.13* |
| Aug 7 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | −0.03 | −0.05* | −0.43* |
| Aug 8 | 0.00 | 0.00 | 0.00 | 0.00 | −0.02 | 0.00 | −0.11 | −0.22 | −0.68* | −0.59* |
| Aug 9 | 0.00 | −0.02 | −0.35 | −0.03 | 0.39* | −0.11 | 0.15 | 0.25 | −1.91* | −1.12* |
| Aug10 | 0.00 | 0.00 | 0.00 | 0.00 | −0.03* | 0.00 | −0.05 | −0.01 | 0.00* | −1.36* |
| Aug11 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | −0.19* | −0.48* | −1.61 |
| Aug12 | 0.00 | 0.00 | −2.72* | −2.12* | −5.84* | 0.00 | −3.42* | −7.54* | −1.00$\circ$ | −15.86 |
| | $M=6\lambda=30$ | | | | | $M=10\lambda=30$ | | | | |
| Aug 6 | −0.03 | −0.48 | −1.00 | −0.56 | −1.31 | 0.00 | −0.61 | −0.08* | −0.48 | −0.42 |
| Aug 7 | 0.00 | 0.40* | −3.04* | −0.55* | −0.32 | 0.00 | 1.03* | 0.58* | −0.75* | −2.64* |
| Aug 8 | 0.00 | −1.06 | −0.76 | −0.30 | −0.87 | 0.00 | −0.09 | −0.59 | −0.47 | −17.65* |
| Aug 9 | 0.15 | 1.41 | −0.69 | 0.45 | −0.50 | 0.55 | −2.11 | 0.31 | 1.21* | −3.74* |
| Aug10 | −0.65 | 0.00 | 0.51 | −0.61 | 0.87* | 0.00 | −0.58 | 0.48 | −1.77* | −7.47* |
| Aug11 | 0.00 | 0.15 | 0.03 | 1.06* | −2.14* | 0.35 | −0.01 | −0.46* | −3.63* | −1.88* |
| Aug12 | 0.00 | −0.57* | −1.63* | −6.75* | −7.70* | 0.00 | −5.56* | −9.32* | −10.38$\circ$ | −23.16* |

**Table 2**
Statistics of absolute errors in kilometers presented in Table 1. *Case* 1 includes all 210 problems; *Case* 2 includes problems with an optimal ILP solution under relative gap tolerance 1%.

| Sample | Size | Mean | std | $t$-value | $p$-value |
|---|---|---|---|---|---|
| *Case* 1 | 210 | −0.79 | 2.73 | −4.18 | 4.3e−5 |
| *Case* 2 | 153 | −0.06 | 0.33 | −2.39 | 0.018 |

**Table 4**
Average time per *RDP* iteration (in seconds) over seven days for problems with $\lambda = 30$; for $\lambda < 30$ the results are roughly the same. The time per iteration increases fast with increasing time horizon $|T|$ and and the number of scenarios $M$.

| $M$ | $\|T\|=4$ | $\|T\|=6$ | $\|T\|=8$ | $\|T\|=10$ | $\|T\|=12$ |
|---|---|---|---|---|---|
| 6 | 0.6 | 2.5 | 4.0 | 5.2 | 5.8 |
| 10 | 0.9 | 4.3 | 7.0 | 8.6 | 10.1 |
| 20 | 1.8 | 8.1 | 13.5 | 16.7 | 19.6 |
| 100 | 9.4 | 40.3 | 68.8 | 87.1 | 98.2 |

we can track the trade-off between the accuracy and execution time of *RDP* over iterations. Such comparisons on the average absolute error (over seven days) versus the average speed-up factor, for the six batches of instances with time horizon $|T| = 8$ hours of Table 1 are illustrated in Fig. 6(a)–(f). Note that after 10 iterations, *RDP* reaches an average error of 0.1 kilometer or less in all cases, except for the case $\lambda = 30$, $M = 10$ where the error drops below 0.1 at iteration 14. In 100 iterations, the gain in terms of kilometers compared to early iterations is quite small. Therefore, an efficient solution is obtained at early iterations. According to Table 4, 10 iterations for cases with $M = 6, 10$ are done in a range of 6 to 100 seconds (averaged over seven days). The average speed-up rate

**Table 3**
Expected number of visits and expected travel distance $(\alpha, z)$ (averages over seven days) by $\lambda, M$ and $T$. In general, for each horizon $T$ and weight $\lambda$, the criteria values $(\alpha, z)$ are robust with respect to the number of scenaris $M$.

| $\lambda$ | $M$ | $\|T\|=4$ | | $\|T\|=6$ | | $\|T\|=8$ | | $\|T\|=10$ | | $\|T\|=12$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 6 | 6.3 | 52.7 | 9.2 | 54.4 | 10.8 | 57.2 | 11.2 | 58.2 | 11.8 | 58.9 |
| 5 | 10 | 6.3 | 52.8 | 9.0 | 55.8 | 10.4 | 56.3 | 11.1 | 57.0 | 10.5 | 56.4 |
| 5 | 20 | 6.2 | 52.3 | 9.2 | 55.1 | 10.5 | 56.5 | 11.4 | 57.8 | 11.8 | 58.3 |
| 5 | 100 | 6.2 | 52.4 | 9.2 | 55.2 | 10.8 | 57.5 | 11.4 | 58.0 | 11.7 | 58.3 |
| 10 | 6 | 8.0 | 63.0 | 12.2 | 73.8 | 14.0 | 78.3 | 15.4 | 87.7 | 15.8 | 85.9 |
| 10 | 10 | 8.1 | 63.7 | 12.8 | 78.4 | 12.4 | 75.5 | 14.5 | 80.8 | 14.4 | 82.3 |
| 10 | 20 | 8.0 | 63.2 | 12.2 | 74.9 | 13.9 | 79.1 | 15.2 | 82.8 | 15.5 | 83.5 |
| 10 | 100 | 8.0 | 63.2 | 12.1 | 74.5 | 14.2 | 80.3 | 15.1 | 83.0 | 15.7 | 85.2 |
| 30 | 6 | 8.9 | 78.1 | 13.6 | 94.8 | 16.0 | 110.2 | 17.8 | 126.5 | 18.4 | 128.2 |
| 30 | 10 | 9.0 | 78.6 | 13.9 | 97.2 | 15.7 | 111.4 | 17.6 | 122.8 | 17.6 | 132.2 |
| 30 | 20 | 8.9 | 78.6 | 13.8 | 98.0 | 16.1 | 111.8 | 17.6 | 122.1 | 18.5 | 131.3 |
| 30 | 100 | 8.9 | 78.9 | 13.7 | 98.3 | 16.3 | 113.8 | 17.7 | 126.8 | 18.6 | 134.2 |

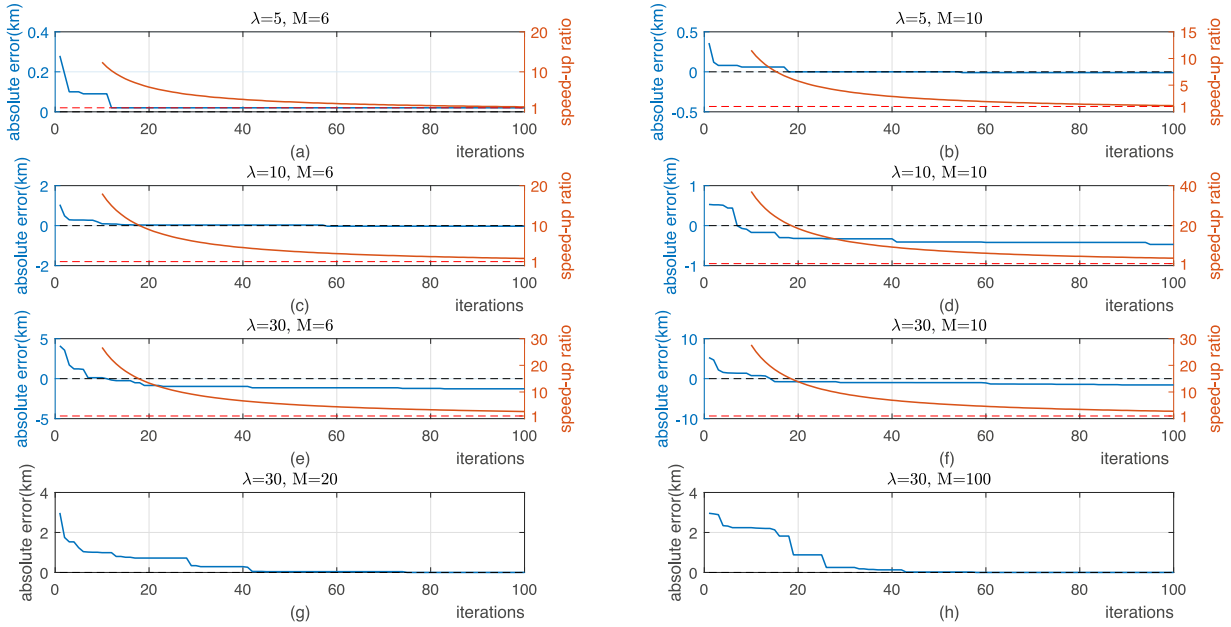


**Fig. 6.** Comparisons on the average absolute error (over seven days) versus the average speed-up factor over *RDP* iterations for time horizon $|T| = 8$. Figures (a)–(f) relates to six batches of instances with $M = 6, 10$ shown in Table 1, based on the total execution time and best solutions returned by the ILP solver. Figures (g) and (h) show for two batches of instances with $\lambda = 30$ and $M = 20, 100$ the daily average difference of the best solution obtained by *RDP* by iteration 100 and the current iteration.

are shown in Fig. 6 starting at iteration 10 onward up to iteration 100. As an example, regarding the case $\lambda = 30$, $M = 10$, *RDP* at iteration 20 returns a feasible set of solutions for seven days with an absolute average error $e = -0.50$ kilometer within an execution time, on average, 15 times faster than the ILP solver for solving the same problems. Speed-up curves show that,as the problem size and complexity increase, the efficiency of *RDP* tends to improve compared to ILP solver.

### 5.3. Further analysis of RDP algorithm and two-stage SP

Next, we discuss a number of items concerning *RDP* and its application to two-stage SP: increasing the number of scenarios, determinants of execution time, the interaction of *Br* and *Fr*, the impact of tour improvements, testing the performance of standard DP for producing feasible solutions, and the value of stochastic solution.

#### 5.3.1. Increasing the number of scenarios

For large values of number of scenarios $M$, ILP solutions are not attempted.However, we ran *RDP* on the case examples with $M = 20, 100$, $|T| = 4, \ldots, 12$ and $\lambda = 5, 10, 30$ as well. Fig. 6(g) and (h) illustrates the convergence for time horizon $|T| = 8$ hours in case of $\lambda = 30$ and $M = 20, 100$. In these cases, errors (compared with ILP solution) are not available, instead, value function differences are calculated at each iteration based on the best solutions obtained by *RDP* in 100 iterations. Time per iteration is shown in Table 4 and results on daily average expected values of $(\alpha, z)$ are shown in Table 3. According to Table 3, the expected number of visits and the corresponding travel distance during each time horizon $T$, mainly depend on the choice of parameter $\lambda$ and remain robust with increasing the number of scenarios. We see in Fig. 6(g) and (h) similar performance in terms of convergence for cases with $M = 20, 100$ and for cases with a smaller number of scenarios. Therefore, with a large number of scenarios, we may expect the speed-up factors further improve. As an instance, the ILP model with August 9 data, 100 scenarios and time horizon of 12 hours has over 14 million binary variables and over 33 thousand constraints. *RDP* returns a feasible solution at each iteration in about 100 seconds.

#### 5.3.2. Determinants of execution time

From the experiments we observed that the execution times per iteration of *RDP* algorithm (Table 4), is not sensitive to parameter $\lambda$. However, the problem complexity resulting from increased $\lambda$ may call for a larger number of iterations. Furthermore, we already pointed out based on Table 3, that the optimal solution in terms of expected values of the two criteria, $\alpha$ and $z$, is rather insensitive to an increase in the number of scenarios. Thus, weight $\lambda$ is the decisive parameter for a given time horizon.

#### 5.3.3. Interaction on Br and Fr

In the alternating $Br - Fr$ iterative procedure, one method (say *Br*) may choose a node later than its optimal position in a sequence. However, this error may be less likely to happen in the counter method (*Fr*) using a counter directed procedure in time. More precisely, in *Br* the optimization problem at node $v_{ik}$ becomes harder when $k$ decreases, because the remaining time slots to the end of time horizon increase. On the contrary, smaller $k$ leads to an easier problem in *Fr*. Passing the information via dynamic values of probabilities $P_f$ can increase the chances of choosing the correct node in the later choices of the counter method *Br* and finally leads to the optimal choice. Memorizing the previously found tour orderings in terms of the probabilities of visiting times, turns out to be effective in speeding up the convergence to the optimal solution. Based on our experiments, random selection of the successor node with constant probabilities (e.g., 0.5 chance of rejection) replacing the empirical probabilities, showed poor performance for the problem instances in Table 1; for brevity, reporting is omitted.

Using the 210 test problem instances discussed in Section 5.2, we also tested the impact of omitting the *Fr* stage from *RDP* and calculating the probabilities $P_f$ based on tours produced by *Br*. For the sample *Case* 2 in Table 2, the average error increased from -0.06 to 0.61 and the omission of *Fr* leads to a statistically significant decrease in performance at $p$-value less than 0.01. A similar test for omitting *Br* resulted in an average error 0.25; also

**Table 5**
Expected loss (kilometer) from using deterministic optimization instead of stochastic optimization. Time horizon is $|T| = 4, 6, 8, 10, 12$ hours, with a 3 hours first period $T_0$; weight $\lambda = 5, 10, 30$; number of scenarios $M = 6, 10$; data of August 6–12, 2018; *ave* = average expected loss over the seven days of August; *max* = maximum expected loss over seven days. The expected loss increases with increasing time horizon $|T|$ and weight $\lambda$ but sensitivity with respect to $M$ is less significant.

| | $\|T\|=4$ | $\|T\|=6$ | $\|T\|=8$ | $\|T\|=10$ | $\|T\|=12$ | $\|T\|=4$ | $\|T\|=6$ | $\|T\|=8$ | $\|T\|=10$ | $\|T\|=12$ |
|---|---|---|---|---|---|---|---|---|---|---|
| | $M=6\lambda=5$ | | | | | $M=10\lambda=5$ | | | | |
| *ave* | 0.04 | 0.52 | 1.45 | 0.97 | 1.71 | 0.04 | 0.64 | 0.72 | 1.13 | 1.46 |
| *max* | 0.26 | 1.93 | 7.51 | 2.23 | 6.50 | 0.24 | 2.04 | 3.71 | 4.43 | 6.06 |
| | $M=6\lambda=10$ | | | | | $M=10\lambda=10$ | | | | |
| *ave* | 0.12 | 1.52 | 1.89 | 3.71 | 2.21 | 0.05 | 1.46 | 1.90 | 2.39 | 2.20 |
| *max* | 0.72 | 2.76 | 6.13 | 12.86 | 4.72 | 0.50 | 4.71 | 7.25 | 6.45 | 6.34 |
| | $M=6\lambda=30$ | | | | | $M=10\lambda=30$ | | | | |
| *ave* | 0.29 | 0.75 | 2.28 | 3.31 | 2.52 | 0.49 | 1.71 | 2.29 | 2.09 | 2.25 |
| *max* | 1.78 | 1.50 | 4.96 | 8.13 | 4.84 | 3.20 | 8.28 | 6.11 | 6.19 | 5.39 |

showing similarly statistically significant decreasein performance. Hence, the interaction of *Br* and *Fr* via empirical probabilities has a positive impact on the performance of *RDP*. Besides, for the fruitful interplay of *Br* and *Fr* in *RDP*, a possible explanation is that the absolute errors obtained from *Br* and *Fr* are uncorrelated making both *Br* and *Fr* valuable.

*5.3.4. The impact of tour improvements*

As mentioned earlier, *RDP* is frequently able to discover the subset of ships included in the optimal solution, even for large sequences and often at early iterations, but in a different order compared to optimal. *RDP* usually ends with the correctly ordered blocks of sub-sequences. Intuitively it seems to be enough to run the swapping method to fix the ordering and run $DP_r$ for adjusting the new order to the optimal time slots. Good performance of combined swapping and $DP_r$ explains why in Figure B.7 (Appendix B) an optimal solution is frequently found in one iteration. Swapping is effective in a majority of cases although there are cases where it fails, since the method does not swap the blocks of sub-sequences.

*5.3.5. Testing the performance of standard DP*

Using standard DP (one iteration of *Br* without randomization) for producing feasible solutions, we get the following results. For the sample *Case* 2 (for which optimal solutions are known for ILP models) in Table 2, the average absolute error increased considerably from the level −0.06 kilometer of *RDP* to 4.0 kilometers of DP.The error often exceeds 20 kilometers for the 42 most difficult problems[9] with $|T| \geq 8$ and $\lambda = 30$. The values obtained from DP lag behind those from *RDP* by 15.5 kilometers on average.

*5.3.6. Testing the impact of the transition node relaxation*

In Section 4 we discussed the relaxation of the transition node employed in our formulation. First, using the 153 instancesof *Case 2* in Table 2, the average of the relaxation error $\Delta V$ is below 0.1 kilometer; hence, the error is negligible given the expected travel distance for a tour ranges from 50 kilometers to 130 kilometers in Table 3. The small error is explained by frequent occurrence problem instances where the relaxed SP already solves the non-relaxed SP involving optimization of the location of a dummy transition node along with the rest of the tour. Second, to assess the excess computational effort due to optimal choice of a dummy transition node, we employed a grid of $1 \times 1$ square kilometers covering the work area with 1320 alternative dummy nodes. Using test instances of Table 1, the ILP formulation was solvable (with gap tolerance 0.01 and 2 hours time limit) in 93% of cases with time horizon $|T| = 4$ hours; however, for $|T| = 6$ hours, only 26% of instances were solvable. Also *RDP* was implemented to include the same grid of dummy nodes. For all cases with $|T| = 4, \ldots, 12$ hours, weight $\lambda = 30$, and $M = 6, 10, 20, 100$ the computing time per RDP iteration (on average over the seven days) increased by a factor ranging from 4 to 12 in comparison with the figures in Table 4.

*5.3.7. Value of stochastic solution*

It is well known that in general the solutions from the deterministic versions of a problem can behave rather badly in a stochastic environment. The reasons are outlined by Wallace (2000). In the comparison test to measure how much worse the deterministic model will act, the deterministic solution is evaluated using the scenarios from the stochastic version of the problem. In this way, the stochastic model is solved with all first-stage variables fixed to the deterministic solution. The resulted value is compared with the objective value of the full stochastic version. In the literature, the difference in expected value is known as the *Value of Stochastic Solution* (*VSS*) (see Maggioni, Kaut, & Bertazzi, 2009; Thapalia, Wallace, Kaut, & Crainic, 2012).

To see the importance of modeling our problem with stochastic parameters, we develop a performance comparison test following a similar concept, using the solution of two-stage SP (*sp*) of Section 4 and the corresponding deterministic model (*det*) discussed in Section 3. *VSS* shows the expected value gained by using stochastic model provided that the given scenarios are true representation of reality.

Suppose the scenarios of ships' trajectories for *sp* discussed above in Section 5.2 depict the true state of the world. At initial time $t_0$ it is not known yet which scenario occurs. The predicted trajectories are realized during the first stage $T_0$, and exactly one of the scenarios is realized during the second stage $T_1$, each of which is equally likely. Both *sp* and *det* are used to optimize the itinerary plan at time $t_0$ over the time horizon $T$ (composed of $T_0$ and $T_1$) and the plans are implemented for $T_0$. At the end of $T_0$, the realized scenario $s \in C$ is observed and the plans are updated by optimizing over the scenario which is deterministic after it is observed. The question is that, how much the expected loss is in terms of value function using *det* compared with *sp*.

At time $t_0$ we calculate the expected value over $T$ (including $T_0$ and $T_1$) in two cases: (i) using *sp* (based on predictions over $T_0$ and $M$ scenarios over $T_1$) to find the tour for $T_0$ and (ii) using *det* (based on predicted trajectories over $T$) to find a tour in $T_0$. In both cases the tour for $T_0$ is implemented and at the end of $T_0$ both find the optimal plan for $T_1$, given the observed scenario and the plan chosen for $T_0$. In case (i) the best tour in $T_1$ for each scenario is already found by solving *sp* and the optimal expected value $V$ (tak-

[9] Over half of these ILP problems were not solved within the 2 hours time limit.

ing into account all $M$ scenarios) is the optimal objective function value of *sp*. For the *det* in case (ii), we need the optimal value for the tour in $T_1$ under each scenario, given the choice for the tour in $T_0$. Such optimal tours are found by solving the scenario-wise separable problem, referred to as the *restricted sp* with binary variables for $T_0$ fixed to optimal levels from the initial *det* over $T$. The optimal objective function value of the restricted *sp*, $V'$, is the expected value if *det* is used for tour optimization in two stages: at $t_0$ for $T_0$ and at the end of $T_0$ for $T_1$. Given the restriction in *sp*, we have $V' \leq V$, and we call $V - V'$ the *expected loss* (*VSS*) revealing how much one expects to lose (in terms of the value function) if *det* is used for tour optimization instead of *sp*. In terms of expected value, *sp* is always at least as good as *det* and the expected loss tells how much better. Table 5 summarizes the expected loss (kilometer) for the 210 problems reported in Table 1.

Table 5 shows the average and maximum expected loss (kilometer) over seven days for the 210 problems reported in Table 1. The average loss ranges from 0.0 kilometer to 3.7 kilometers and the maximum from 0.2 kilometer to 12.9 kilometers. Small values mean that approximating the stochastic parameters (trajectories) by the predicted values (instead of random trajectories) is a good choice. For short time horizon $|T| \leq 6$ hours, the average loss is small compared with $|T| \geq 8$ hours. Cases with a large weight $\lambda \geq 10$ for $\alpha$ lead to a higher average loss than cases with $\lambda = 5$. The cases with $|T| \geq 8$ hours and $\lambda \geq 10$ are of most interest in practice; for these cases the smallest average loss is larger than the largest average loss among the other cases with $|T| \leq 6$ or $\lambda = 5$.

## 6. Conclusion

In this article we model a bi-criteria moving-target travelling salesman problem with the assumption that the locations of targets are predicted but uncertain. The model is applied for decision support in optimal routing of a surveillance boat measuring greenhouse gas emission from vessels navigating in the Baltic sea. Given that all targets cannot be reached, the management sets two goals: to maximize the number of visits and minimize the total travel distance. For solving the subsequent two-stage stochastic programming problem for maximizing the expected value of a linear value function, we introduce an iterative randomized dynamic programming algorithm where forward and backward recursion stages are linked by empirical probabilities of timing the measurements. Each iteration of *RDP* provides a feasible tour for the problem. Based on the randomization, we show that the algorithm converges to an optimal solution with probability one. The exchange of information between forward and backward stages acts as a reinforcement mechanism in learning the favorable probabilities, and as a key for success of *RDP*. Using either forward or backward stage independently leads to a significant loss in the efficiency.

Using a medium size deterministic problem with a hundred thousand binary variables, we illustrate that the expected number of *RDP* iterations for obtaining an optimal solution can be small, from one to ten iterations. For the two-stage stochastic programming case with 10 scenarios, the size of such problem is eight times bigger in terms of binary variables. Our tests show that *RDP* can reach in 10 iterations an optimal solution within a small acceptable tolerance.

We also formulate the problem as a stochastic ILP model. In our sample of 210 test problems with real data, small to medium size ILP problems are solvable with 1% gap tolerance in a reasonable time and solutions found by RDP and ILP solver are shown equally good; however, RDP is significantly faster and able to deal with large-scale problems as well. For larger problem sizes, *RDP* can become extraordinarily efficient compared to a general ILP solver which often fails to return a feasible solution within a desired time limit. The *RDP* algorithm can be applied efficiently for the stochastic routing problem of huge size and return a high-quality solution even in minutes. We develop a model for generating scenarios based on the predicted locations of ships (targets), and observe that the stochastic programming solution is robust to increasing the number of scenarios from 10 to 100.

The convergence of *RDP* iterations is sensitive to the choice of distribution interval parameters employed for randomization. Loose boundaries lead to extra calculation on producing low quality solutions while too tight boundaries can result in being trapped in local optima.

We also demonstrate the value of stochastic solution (*VSS*) using the sample of the 210 test problems and show that applying deterministic (instead of stochastic) model in the uncertain dynamic network leads to an expected loss up to 13 kilometers in extra travel distance.

In theory, convergence to a global optimum with probability one is guaranteed. However, for real applications one needs to be content with the best solutions found in one or a few hundred iterations. Nevertheless, our computational tests indicate that good enough solutions for practical needs are found in early iterations.

Possible future extensions and developments of our work include adaptation of *RDP* to parallel processing to deal with a large number of scenarios. Scalability is ensured by the fact that recursive computations in *RDP* can be carried out in parallel for each scenario; hence significant benefit can be gained for models with a large number of scenarios. Regarding further methodology improvement, reinforcement learning approach is an interesting candidate for extending *RDP* to be applied for solving deterministic or stochastic MT-TSP.

## Acknowledgment

The first author was financially supported by the University of Turku graduate school, doctoral programme in exact sciences (EXACTUS).

## Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2023.01.009.

## References

Abeledo, H., Fukasawa, R., Pessoa, A., & Uchoa, E. (2013). The time dependent traveling salesman problem: Polyhedra and algorithm. *Mathematical Programming Computation, 5*, 27–55.

Adasme, P., Andrade, R., Leung, J., & Lisser, A. (2016). A two-stage stochastic programming approach for the traveling salesman problem. In *Proceedings of 5th the international conference on operations research and enterprise systems 1: ICORES 163–169, Rome, Italy*.

Applegate, D. L., Bixby, R. E., Chvâtal, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study*. Princeton University Press.

Archetti, C., Feillet, D., Mor, A., & Speranza, M. G. (2020). Dynamic traveling salesman problem with stochastic release dates. *European Journal of Operational Research, 280*, 832–844.

Balas, E. (1989). The prize collecting traveling salesman problem. *Networks, 19*(6), 621–636.

Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press.

Bellman, R. E. (1962). Dynamic programming treatment of the travelling salesman problem. *Journal of the Association for Computing Machinery, 9*, 61–63.

Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). Robust optimization. *Princeton series in applied mathematics*. Princeton University Press.

Beraldi, P., Ghiani, G., Musmanno, R., & Vocaturo, F. (2010). Efficient neighborhood search for the probabilistic multi-vehicle pickup and delivery problem. *Asia-Pacific Journal of Operational Research, 27*(3), 301–314.

Bertsimas, D. (1992). A vehicle routing problem with stochastic demand. *Operations Research, 40*, 574–585.

Bertsimas, D., & Ryzin, G. V. (1991). A stochastic and dynamic routing problem in the euclidean plane. *Operations Research, 39*(4), 601–615.

Birge, J. R., & Louveaux, F. V. (2011). *Introduction to stochastic programming* (2nd ed.). New York: Springer Verlag.

Blackmore, L., Ono, M., & Williams, B. C. (2011). Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics, 27*(6), 1080–1094.

Bourjolly, J., Gurtuna, O., & Lyngvic, A. (2006). On-orbit servicing: A time-dependent, moving-target traveling salesman problem. *International Transactions in Operational Research, 13*, 461–481.
Choubey, N. S. (2013). Moving target travelling salesman problem using genetic algorithm. *International Journal of Computer Applications,, 70*(2), 30–34.
Dantzig, G. B. (1955). Linear programming under uncertainty. *Management Science, 1*(3–4), 197–206.
Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., & Gambardella, L. M. (2008). Time dependent vehicle routing problem with a multi ant colony. *European Journal of Operational Research, 185*, 1174–1191.
Florio, A. M., Hartl, R. F., & Minner, S. (2020). Optimal a priori tour and restocking policy for the single-vehicle routing problem with stochastic demands. *European Journal of Operational Research, 285*(1), 172–182.
Furini, F., Persiani, C. A., & Toth, P. (2016). The time dependent traveling salesman planning problem in controlled airspace. *Transportation Research Part B: Methodological, 90*, 38–55.
García, J., & Peña, A. (2018). Robust optimization: Concepts and applications. In J. D. Ser, & E. Osaba (Eds.), *Nature-inspired methods for stochastic, robust and dynamic optimization*. London: IntechOpen.
Golden, B., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics, 34*(3), 307–318.
Groba, C., Sartal, A., & Vázquez, X. (2015). Solving the dynamic traveling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices. *Computers and Operations Research, 56*, 22–32.
Hammar, M., & Nilsson, B. J. (2002). Approximation results for kinetic variants of TSP. *Discrte and Computational Geometry, 27*, 635–651.
Hassoun, M., Shoval, S., Simchon, E., & Yedidsion, L. (2020). The single line moving target traveling salesman problem with release times. *Annals of Operations Research, 289*, 449–458.
Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society of Industrial and Applied Mathematics, 10*, 196–210.
Helvig, C., Robins, G., & Zelikovsky, A. (2003). The moving-target traveling salesman problem. *Journal of Algorithms, 49*, 153–174.
Jabali, O., Rei, W., Gendreau, M., & Laporte, G. (2014). Partial-route inequalities for the multi-vehicle routing problem with stochastic demands. *Discrete Applied Mathematics, 177*, 121–136.
Jaillet, P. (1988). A priori solution of a travelling salesman problem in which a random subset of the customers are visited. *Operations Research, 36*, 929–936.
Jiang, Q., Sarker, R., & Abbass, H. (2005). Tracking moving targets and the non-stationary traveling salesman problem. *Complexity International, 11*, 171–179.
Kaut, M., & Wallace, S. W. (2007). Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization, 3*(2), 257–271.
Kryazhimskiy, A., & Savinov, V. (1995). The travelling-salesman problem with moving objects. *Journal of Computer and System Sciences International, 33*, 144–148.
Laporte, G., Louveaux, F., & Mercure, H. (1992). The vehicle routing problem with stochastic travel times. *Transportation Science, 26*(3), 161–170.
Maggioni, F., Kaut, M., & Bertazzi, L. (2009). Stochastic optimization models for a single-sink transportation problem. *Computational Management Science, 6*, 251–267.
Malandraki, C., & Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science, 26*(3), 185–200.
Malandraki, C., & Dial, R. B. (1996). A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research, 90*(1), 45–55.
Maskooki, A., Deb, K., & Kallio, M. (2022). A customized genetic algorithm for bi-objective routing in a dynamic network. *European Journal of Operational Research, 297*(2), 615–629.
Maskooki, A., & Nikulin, Y. (2020). Bi-objective routing in a dynamic network: An application to maritime logistics. *Control and Cybernetics, 49*(2), 211–232.
Maskooki, A., Virjonen, P., & Kallio, M. (2021). Assessing the prediction uncertainty in a route optimization model for autonomous maritime logistics. *International Transactions in Operational Research, 28*(4), 1765–1786.
MJHB, G. (2006). Routing of platforms in a maritime surface surveillance operation. *European Journal of Operational Research, 170*, 613–628.
Oyola, J., Arntzen, A., & Woodruff, D. (2018). The stochastic vehicle routing problem, a literature review, part I: models. *Journal on Transportation and Logistics, 7*(3), 93–221.
Picard, J., & Queyranne, M. (1978). The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research, 26*, 86–110.
Ruszczynski, A., & Vanderbei, R. (2003). Frontiers of stochastically non-dominated portfolios. *Econometrica, 71*(4), 1287–1297.
Shapiro, A., Dentcheva, D., & Ruszczyński, A. (2009). Lectures on stochastic programming: Modeling and theory. Society for Industrial and Applied Mathematics and the Mathematical Programming Society (MPS-SIAM) series on optimization 9. Philadelphia.
Stieber, A., & Fügenschuh, A. (2022). Dealing with time in the multiple traveling salespersons problem with moving targets. *Central European Journal of Operations Research, 30*, 991–1017.
Thapalia, B., Wallace, S., Kaut, M., & Crainic, T. (2012). Single source single-commodity stochastic network design. *Computational Management Science, 9*, 139–160.
Toriello, A., Haskell, W. B., & Poremba, M. (2014). A dynamic traveling salesman problem with stochastic arc costs. *Operations Research, 62*(5), 1107–1125.
Viel, C., Vaultier, U., Wan, J., & Jaulin, L. (2019). Genetic algorithm-based multiple moving target reaching using a fleet of sailboats. *IET Cyber-Systems and Robotics, 1*(3), 93–100.
Vu, D., Hewitt, M., Boland, N., & Savelsbergh, M. (2019). Dynamic discretization discovery for solving the time dependent traveling salesman problem with time windows. *Transportation Science, 54*(3), 703–720.
Wallace, S. (2000). Decision making under uncertainty: Is sensitivity analysis of any use. *Operations Research, 48*(1), 20–25.
Zhang, G., Jia, N., Zhu, N., Adulyasak, Y., & Ma, S. (2023). Robust drone selective routing in humanitarian transportation network assessment. *European Journal of Operational Research, 305*(1), 400–428.
Zhang, S., Ohlmann, J. W., & Thomas, B. W. (2014). A priori orienteering with time windows and stochastic wait times at customers. *European Journal of Operational Research, 239*(1), 70–79.