Control and Cybernetics
vol. 49 (2020) No. 2

Bi-objective routing in a dynamic network: an application to maritime logistics*

by

Alaleh Maskooki and Yury Nikulin

University of Turku, Department of Mathematics and Statistics,
FI-20014 Turku, Finland
alamas@utu.fi
yurnik@utu.fi

Abstract: A bi-objective MILP model for optimal routing in a dynamic network with moving targets (nodes) is developed, where all targets are not necessarily visited. Hence, our problem extends the moving target travelling salesman problem. The two objectives aim at finding the sequence of targets visited in a given time horizon by minimizing the total travel distance and maximizing the number of targets visited. Due to a huge number of binary variables, such a problem often becomes intractable in the real life cases. To reduce the computational burden, we introduce a measure of traffic density, based on which we propose a time horizon splitting heuristics. In a real-world case study of greenhouse gas emissions control, using Automatic Identification System data related to the locations of ships navigating in the Gulf of Finland, we evaluate the performance of the proposed method. Different splitting scenarios are analysed numerically. Even in the cases of a moderate scale, the results show that near-efficient values for the two objectives can be obtained by our splitting approach with a drastic decrease in computational time compared to the exact MILP method. A linear value function is introduced to compare the Pareto solutions obtained by different splitting scenarios. Given our results, we expect that the present study is valuable in logistic applications, specifically maritime management services and autonomous navigation.

Keywords: travelling salesman; time dependent network; multi-objective optimization; integer programming

1. Introduction

In many real-world applications, there is a need to find an efficient route passing through all or a subset of nodes in a network. One of the classical routing problems is to find a Hamiltonian circuit of minimal total weight, connecting all nodes in a weighted undirected graph; this fundamental problem, known as the Travelling Salesman Problem (TSP), is NP-hard (Garey and Johnson, 1979), as it appears in network logistics and computer science (Papadimitriou and Steiglitz, 1982). Classical formulations of TSP

*Submitted: April 2020; Accepted:June 2020.

include integer linear programming models, but the presence of an enormous (usually exponential) number of sub-tour elimination constraints makes the problem intractable in terms of solving by means of standard exact methods. Significant improvement can be achieved whenever branch and bound methods are equipped with cutting plane generation techniques (Applegate et al., 2007) allowing for computing the optimum for truly large scale networks with the help of supercomputers. The usage of modern heuristics allows for processing of networks up to a million nodes on standard CPUs, and such instances can be resolved to optimality with 2-3 percentage accuracy (see Rego et al., 2011).

In addition to the size of the network, the presence of multiple objectives essentially contributes to problem complexity. Under multi-objective framework, a solution that is optimal with respect to one objective may have poor values for the others, and thus may be unacceptable for a decision maker in practical situations. Therefore, many problems arising in optimization should be ultimately considered under multi-criteria framework due to the existence of several conflicting goals or interests (Branke et al., 2008).

Classical TSP has many variations, including Time-Dependent TSP (TDTSP), where the edge cost depends not only on the distance between nodes (cities), but also the time when transition happens in the tour. This feature introduces a new level of complexity since now optimal sequencing depends on the time, in which every node is visited. Time dependent nature of routing has to be properly addressed in such models (see, e.g., Androutsopoulos and Zografos, 2009; Groba, Sartal and Vázques, 2015). The problem was formulated as an integer linear programming model by Fox (1973) for brewing industry. Other early linear and quadratic integer programming models can be found in Fox, Gavish and Graves (1980) and Picard and Queyranne (1978). In distribution and scheduling problems, the travel cost between customers in a congested urban environment is a function of traffic density, which varies over the time of the day (Malandraki and Daskin, 1992). Vander Wiel and Sahinidis (1996) introduced a Mixed Integer Linear Programming (MILP) formulation for the problem, using linearization of the quadratic assignment TDTSP model of Picard and Queyranne (1978) and solved it by an exchange heuristic on the underlying multipartite graph. The formulations of Picard and Queyranne are the base models for many other approaches, developed for TDTSP, including the studies of Bianco, Mingozzi and Ricciardelli (1993) and Lucena (1990). Unlike previous studies, where time-dependency is addressed in terms of travel times, Taş et al. (2016) focused on minimizing the total route duration with linear and quadratic service time functions. They describe the basic properties for certain classes of service time functions, and, based on that, they state conditions under which waiting can be beneficial. Another variant of TSP is referred to as *moving-target TSP*, where the problem is to find the fastest tour while intercepting all moving targets. Helvig, Robins and Zelikovsky (2003) studied moving-target TSP with the assumption that nodes are moving linearly with a constant velocity. Under such assumption, no waiting time exists in an optimal tour.

TDTSP models assume nodes of the graph to be stationary. As a result, the location of the last visit does not need to be reserved. One common approach for weighting the edges

of a time-dependent graph is to handle the travel cost function as a step function (Furini, Persiani and Toth, 2016; Malandraki and Daskin, 1992) depending on the time of the day with a small number of time slots. In our model, we deal with a significantly larger number of time slots, compared to studies referred above. On the other hand, they deal with minimization of the total travel time. A time indexed formulation was recently published by Vu et al. (2019) for solving the Time-Dependent Travelling Salesman Problem with Time Windows (TD-TSPTW), which is defined on a time-expanded network. It is solved in the dynamic discretization framework, based on FIFO (First-In First-Out) property on travel times, with the objective of minimizing the total tour duration including waiting time. In such a case, waiting occurs when the traveller arrives at a location before the corresponding time window opens. Vu et al. (2019) tested their method on generated instances using a piecewise linear travel time function, which satisfies the FIFO property.

The aim of our research is to develop a mathematical model and solution approach for finding an optimal itinerary plan for an environmental surveillance vessel navigating in a specific area of the sea among traffic lines for measuring greenhouse gas emissions from major ships, such as passenger ferries, cargo ships and tankers. The importance of the subject is due to the fact that shipping accounts for over three percent of global greenhouse gas emissions. Sulphur emission in European waters are several times higher than from all passenger cars in Europe in total, according to Mikkonen (2019). As for other possible applications, the proposed formulation can be applied in maritime surveillance operations (Marlow, Kilby and Mercer, 2007) to detect illegal activities, for purposes of airdrop for replenishment of naval vessels (Hewgley and Yakimenko, 2012), in the resupply of patrolling ships by a supply vessel, or for fish aggregating devices (Groba, Sartal and Vázques, 2015).

In this case study problem there are two objectives, aiming to minimize the total travel distance and maximize the number of measurements done. The proposed model finds efficient alternatives for the number and the sequence of ships visited, as well as the start times and locations of each measurement process. Additional features of the real-world case study create some main differences between the present and previously proposed models. In our model, there is no assumption on traffic density distribution or travel time sequence chart. In a dynamic network, the objective of minimizing travel distance and travel time are very different. In contrast to the moving target model in Helvig, Robins and Zelikovsky (2003), waiting in some time slots can be beneficial when minimizing the distance or fuel consumption. We include the waiting possibility such that the time and location of each ship visited is reserved for possible connection to the next target. In distinction from previous models, visiting all ships is no longer possible in the given time window, due to multiple factors, such as a large working area, speed of the surveillance vessel, limitation in working hours, and fuel consumption. Consequently, our problem turns into a bi-objective model for minimizing the travel distance and maximizing the number of nodes visited.

We develop a bi-objective model for optimal routing in a dynamic network with moving targets, where all targets are not necessarily visited. Hence, our problem extends the

moving target travelling salesman problem, in which the goal is to minimize total travel time while visiting all targets. The problem setting is in our case different: given n targets and a positive integer $\alpha \leq n$, choose simultaneously a subset A (of α targets) from a set N (of n targets), as well as an optimal itinerary for visiting all targets in the subset A. Thus, we define our problem as the following two-level optimization problem:

$$\min_{A \subseteq N} \{ z(A) \mid |A| = \alpha \}$$

where $z(A)$ is defined as the minimum travel distance in the inner level problem of a moving target TSP, defined by the set A. We introduce a mathematical programming model to solve such new extension of TSP for which, to the best of our knowledge, there is no model in the existing literature that can be directly applicable to solve our case study. Of course, when we discuss extensions of TSP, appearing in the literature, unavoidably there are similarities in various formulations as well.

In the next section, the problem is defined and a MILP model is proposed for finding the efficient frontier of the bi-objective problem. In Section 3 a density measure is introduced for splitting the time horizon, and the proposed model is applied on sequential sub-intervals, each finding a sub-route. The numerical experiments and discussion on the performance of the proposed model on real-world datasets are presented in Section 4. Section 5 concludes the paper and introduces the directions for further works.

2. Problem description and MILP formulation

The optimization problem addressed in this paper arises from a real logistic problem of finding an optimal routing for a vessel measuring greenhouse gas emissions from ships in a specific area of the sea, referred to as the *working area*. The present model can be considered as an extension of Picard and Queyranne's (1978) formulation. We characterize our problem as follows: Consider a variant of TSP problem with dynamic nodes; i.e. locations of nodes change over time. The nodes (ships) move along estimated routes. They are present in the network only during their time window: the window starts when the ship enters the working area and ends when it leaves the area. The bi-objective vector maximization problem is to determine how the surveillance vessel with a given maximum speed should travel among moving nodes assuming two criteria: (1) traveling the shortest possible distance, and (2) visiting as many ships as possible. Unlike in TSP, all ships need not be visited. An animated illustration of the underlying dynamic network and an example of optimal routing can be found in Github (2020).

2.1. Mathematical formulation

Let T denote the time horizon, during which the travelling by the surveillance vessel is done, let indices $i = 1, \ldots, n$ refer to ships present in the working area sometime during T, and let $i = 0$ refer to the depot (harbour).

In order to formulate the problem as a combinatorial problem, we discretize the time horizon T into m time slots by time points $t_0 < \cdots < t_m$ where t_0 and t_m points refer to

the beginning and the end of T, respectively, and we assume that the locations of ships remain unchanged during each time slot. The m time slots are of equal length w; hence $t_0 = 0$ and $t_k = kw$, for $k = 1,\dots m$. Henceforth, for $k = 1\dots m$, time slot k refers to the interval $[t_{k-1}, t_k) = [(k-1)w, kw)$, and time slot $k = 0$ denotes the initial time point t_0. Visiting of a ship i needs a processing time p_i before leaving and visiting the next ship; for the depot $i = 0$, let $p_0 = 0$. The time slots are chosen short enough not to include more than one processing, that is $w < 2p_i$, for all $i > 0$.

For $i = 0, 1, \dots, n$, a ship i is considered as being present in the working area only during its time window $\{t_{a_i}, \dots, t_{b_i}\} \subseteq \{t_0, t_1, \dots, t_m\}$*. For the depot, $i = 0$, we assume $\{t_{a_0}, \dots, t_{b_0}\} = \{t_0, \dots, t_m\}$. For $i \geq 0$, let $v_i^k \in \mathbb{R}^2$ be a coordinate vector stating the location of ship i at time slot $k \in \{a_i, \dots, b_i\}$. Given that the locations of ships remain unchanged during each time slot, we obtain the coordinates of v_i^k from the moment $(k-1)w$, which is the beginning moment of time slot k †.

We define a network flow model over a layered graph $\mathscr{G}$. Each layer corresponds to a fixed time slot $k = 0, \dots, m$ and consists of nodes defined by coordinate vectors v_i^k, for $i = 0, ..., n$, that is, the coordinates of ships in their k^{th} time slot. If for a ship i, we have $k \notin \{a_i, \dots, b_i\}$, then ship i is not present in time slot k and thus the corresponding node does not exist in layer k. Let $\mathscr{N}^k = \{v_i^k | i \in \{0, ..., n\},\ k \in \{a_i, \dots, b_i\}\}$ be the set of nodes v_i^k of all ships present at time slot k. We assume that v_0^k, the depot node, is present in all layers of $\mathscr{G}$ and $i = 0$ is non-moving during the time horizon T. Therefore, the traveller can start from and return to the depot at any time to complete the tour. The edges of the graph connect the nodes of a layer to those of the later layers.

Let $\mathscr{A}^{kl} = \{(v_i^k, v_j^l) \in \mathscr{N}^k \times \mathscr{N}^l | l > k\}$ be the set of directed edges connecting nodes in a pair of layers k and l, with sources in $\mathscr{N}^k$ and terminals in $\mathscr{N}^l$. Thus, $\mathscr{G}$ is a multi-partite directed graph consisting of the union of all sets of nodes $\mathscr{N}^k$ and their pair-wise connecting edges in $\mathscr{A}^{kl}$. Because at most one processing is possible in each time slot, the nodes in the same layer in $\mathscr{G}$ are not connected. The schematic representation of the graph $\mathscr{G}$ is illustrated in Fig. 1.

The weight of each edge (v_i^k, v_j^l) is defined as the Euclidean distance between the coordinates of the nodes $d_{ij}^{kl} = ||v_i^k - v_j^l||$. The speed of the vessel is assumed to be limited by a fixed value c. Therefore, corresponding to each distance d_{ij}^{kl} there is a minimum travel time with the speed c, which is denoted by $\delta_{ij}^{kl} = d_{ij}^{kl}/c$.

As mentioned before, the resulting problem has two objectives: minimizing the travel distance and maximizing the number of nodes visited during the time horizon T. We denote the two criteria by z and α, respectively. If $u(\alpha, z)$ is a value function representing

*Ships can exit the working area and come back several times, in which case their time window is composed of disjoint sequences of time slots.

†Different ships i and j can be in the same location if $v_i^k = v_j^l$; however, we have $k \neq l$; i.e., the ships can be at the same locations but not at the same time.

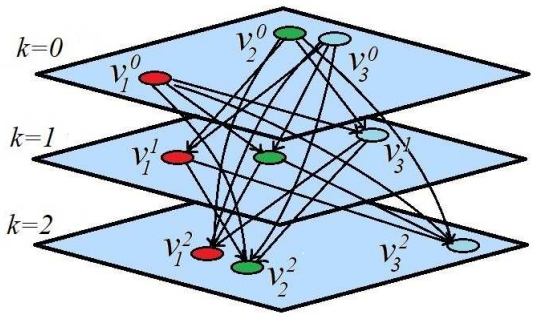

Figure 1. Representation of graph $\mathscr{G}$ with three ships and three time slots

preferences of the decision maker, then the objective is to find α and z which maximize $u(\alpha,z)$. Given that α and $-z$ are subject to maximization, we have $\partial u/\partial\alpha > 0$ and $\partial u/\partial z < 0$ and the optimal solution (α,z) is Pareto optimal; i.e., an optimal (α,z) is on the efficient frontier. Given that an estimation for u is available, the problem results in a single objective optimization. In a special case, the value function u is linear. In this case u is a weighted sum scalarization; this amounts to giving each objective a weight and maximizing the weighted aggregation. The case of a linear value function is demonstrated in Section 4.

For the bi-objective problem, we may determine the efficient frontier first and let the decision maker choose the most preferred solution thereafter. For computing the efficient frontier, several options exist. First, using a set of linear value functions, the calculation is efficient since the value function $u(\alpha,z)$ is a linear objective function; however, only the supported Pareto points can be found. Second, using the reference point method (Wierzbicki, 1982) employing Chebyshev scalarization, all the Pareto points can be detected, but this type of scalarization is computationally more challenging. More on multi-objective optimization methods can be found in Miettinen (1998).

Observing that the objective α (ships visited) has positive integer values, a simple method, which we use to handle the bi-objective problem, is to deal with the objective α as an additional constraint. That is, we set a goal for the value of α by considering it as a constraint, and solve the problem of minimizing the objective z (the distance travelled) while meeting the goal on α. Solving such problems for $\alpha = 1,2,3,\ldots$ yields the efficient frontier[‡].

The minimization of travel distance employs binary variables x_{ij}^{kl}, which are equal to 1 if node v_j^l is visited immediately after node v_i^k and 0 otherwise. Formally, define the

[‡]In practice, finding solutions that involve the inspection of only a few ships, for instance 1 or 2, may be unnecessary. Thus, evaluating only a part of the efficient frontier is needed, and the decision maker might set a threshold level for α, under which the efficient points are of no interest, and therefore, need not be evaluated.

domain of binary variables as follows:

$$x_{ij}^{kl} \in \{0,1\} \qquad \forall\, i,j = 0 \ldots n\,(i \neq j) \qquad \forall\, k,l = 0 \ldots m\ (k < l),$$

where each binary variable x_{ij}^{kl} is defined only if both nodes v_i^k and v_j^l are in the working area. While the latter condition is of course implemented in our computer code, for notational convenience, in the sequel we suppress such feasibility sets for v_i^k and v_j^l.

In this notation, the total travel distance is given as follows:

$$z = \sum_{\substack{k,l=0 \\ (l>k)}}^{m} \sum_{\substack{i,j=0 \\ (i\neq j)}}^{n} d_{ij}^{kl} x_{ij}^{kl}. \tag{1}$$

We define an integer parameter α to indicate the number of visits. The Pareto frontier could then be generated in terms of z versus parameter α by varying it over the set $\{1,\ldots,n\}$. Given α, the goal constraint is formulated as follows:

$$\sum_{\substack{k,l=0 \\ (l>k)}}^{m} \sum_{\substack{i=0,j=1 \\ (i\neq j)}}^{n} x_{ij}^{kl} = \alpha. \tag{2}$$

Initially, the flow starts from depot node v_0^k at the time slot $k \geq 0$. If the decision is made to move to some v_j^l $(l > k, j \neq 0)$, then $x_{0j}^{kl} = 1$. At the time slot m the surveillance vessel is supposed to be back to the depot, so the time slot m is not included in the summation. Therefore, the following constraint ensures exactly one exit from depot node and the first visit after that:

$$\sum_{\substack{k,l=0 \\ (k<l)}}^{m-1} \sum_{j=1}^{n} x_{0j}^{kl} = 1. \tag{3}$$

If ship $j > 0$ is visited in the time slot l, $0 < l < m$, then the flow should enter the node v_j^l from some node v_i^k with $i \neq j$ at an earlier time slot $k < l$, and exit the node v_j^l to some node v_i^k with $i \neq j$ at a later time slot $k > l$. So, initialized by the constraint (3), one unit of flow propagates throughout the nodes v_i^k and v_j^l in each intermediate step, which is ensured by the following balance constraints (flow conservation):

$$\sum_{k=0}^{l-1} \sum_{\substack{i=0 \\ (i\neq j)}}^{n} x_{ij}^{kl} = \sum_{k=l+1}^{m} \sum_{\substack{i=0 \\ (i\neq j)}}^{n} x_{ji}^{lk} \qquad \forall j = 1 \ldots n \quad \forall l = 1 \ldots m-1. \tag{4}$$

The flow conservation constraint (4) maintains the connection of time slots and eliminates route breaks at the time slot l. Finally, for some $i > 0$ and $k < m$, after visiting ship i at the time slot k the flow is forced to turn back to depot node v_0^l at some time slot $l > k$. This is achieved using the following constraint:

$$\sum_{\substack{k,l=1 \\ (l>k)}}^{m} \sum_{i=1}^{n} x_{i0}^{kl} = 1. \tag{5}$$

The constraints (3)–(5) are based on the classical formulation of TSP, which is modified for the dynamic network. However, we observe that constraint (5) is in fact redundant, because it is implied by (3) and (4). The single equation (5) presumably does not harm optimization, or it may even speed up computation. Note that sub-tour elimination results from $l > k$ for all x_{ij}^{kl}, so that no node v_i^k can be re-visited. To see this, assume that a sub-tour starts at some node v_i^k, passes a sequence of nodes $\{v_j^l\}$ with $l > k$ and returns from some node v_j^l to the starting node v_i^k; but then $k > l$, which is a contradiction. Hence, a sub-tour is not possible. Thus, $l > k$ implies that the vessel will not visit node v_i^k more than once, but we also need to guarantee that each ship $i > 0$ will not be revisited at some later time $l > k$, by the following inequality:

$$\sum_{\substack{l.k=1 \\ (l>k)}}^{m} \sum_{\substack{j=0 \\ (i\neq j)}}^{n} x_{ij}^{kl} \leq 1 \qquad \forall i = 1...n. \tag{6}$$

By the set of constraints (2)–(6), we start from the depot, visit a sequence of α nodes along the route $\{(v_i^k, v_j^l) | x_{ij}^{kl} = 1\}$ and return back to the depot. So, for each fixed value of α in (2), a Hamiltonian circuit $\{v_0^{k_0}, v_{i_1}^{k_1}, \ldots, v_{i_\alpha}^{k_\alpha}\}$ can be generated.

To create a feasible schedule, we need to force visiting each node v_i^k within the time slot k. Firstly, if a visit occurs within the time slot k, then the associated start time of processing s_k should fall into the same time slot, and as a result satisfy the following constraint:

$$t_0 + w(k-1) \leq s_k \leq t_0 + wk \qquad \forall k = 1...m. \tag{7}$$

For $k = 0$, $s_0 = t_0$ is the time of starting from v_0^0 at the beginning of the horizon T. Recall that parameter w is the length of the time slots; i.e., $w = t_k - t_{k-1}$, for $k = 1, \ldots, m$, determining the scale of granularity for time discretization. Smaller value for w gives more accurate locations for the nodes v_i^k, but a larger size of the network in return. So, there is a trade-off between the time complexity and accuracy of the location. Secondly, we need to reserve enough time for processing and travelling to the next ship for a visit. This is assured by the following scheduling constraint:

$$s_k + \sum_{\substack{i,j=0 \\ (i\neq j)}}^{n} (p_i + \delta_{ij}^{kl}) x_{ij}^{kl} \leq s_l \qquad \forall k, l = 0...m \ (k < l). \tag{8}$$

To further clarify what variable s_k represents, if a ship is visited in time slot k, then s_k is the time stage of starting the measurement (serving) within the time slot k. Under our assumptions, at most one ship can be served in a single time slot. If no ship is served in time slot k, then s_k may be any time stage in time slot k, satisfying (7). The binary variables x_{ij}^{kl} indicate which ship (if any) is served in time slot k; i.e., $x_{ij}^{kl} = 1$ (for some ship j and time slot $l > k$) implies that ship i is served in time slot k. Additionally, constraint (8) builds the link between ships i and j, and starting times s_k and s_l. Consequently, there

is no need to include the ship index in starting times s_k. Note that optimal waiting time at node v_j^l is implied by the slack in the inequality (8).

To interpret the waiting time, assume node v_i^k is visited first, followed by visiting node v_j^l thereafter, and the vessel can reach location v_j^l with constant speed c before the beginning of time slot l; i.e., before time stage $(l-1)w$.

The solution of the MILP model (1)–(8) gives the shortest path for processing α ships, as well as the start times of processing each ship in a given time horizon T. We refer to (1)–(8) as the Bi-objective Dynamic TSP (BDTSP) model.

In TDTSP a function of time t is used for calculating the travel time among nodes, but the nodes are stationary and thus a small number of time slots is sufficient to plan for the whole day (for example morning and evening rush hours and lighter traffic in mid-day). Unlike in TDTSP, due to the moving targets, the number of time slots m in the above model is relatively large and the size of our model grows quadratically with the number of time slots. More precisely, it contains $0.5m^2+(1.5+n)m+3$ constraints. An upper bound for the number of binary variables is $0.5n^2m^2$. The exact number depends on the length of the time window for each ship in terms of the time slots. For different values of α, the model returns the Pareto optimal solutions for the problem BDTSP. It can be solved by standard MIP solvers, however, the huge number of binary variables and constraints make the problem intractable when based on large-size real-world datasets. In the next section, a method is proposed, which returns a near-optimal (near-efficient) solution within a significantly lower execution time compared to that of the exact method.

3. Solution methods

Standard software may be directly applied for solving BDTSP, defined by (1)–(8); however, in real-world cases the number of ships and time slots tend to be prohibitively large. Next, to deal with such curse of dimensionality, we consider two rather straightforward but valuable options: first, reducing the number of binary variables by preprocessing, and second, employing heuristics for splitting the time horizon T into consecutive subintervals and finding for the sub-intervals the inter-linked partial routes, which jointly form a Hamiltonian circuit over T. The novelty of the splitting heuristic is to introduce a measure of traffic density to be employed for time horizon splitting.

Preprocessing

For each ship i, each node v_i^k of the graph on the layer of time slot k is connected to all other nodes v_j^l of ships $j \neq i$ on all later layers of time slots $l > k$. However, not all of these links are feasible in terms of actual travelling with the limited speed of the surveillance vessel. As defined earlier, given the distance d_{ij}^{kl} from node v_i^k to v_j^l, the time to travel between the two nodes with the constant speed c is $\delta_{ij}^{kl} = d_{ij}^{kl}/c$. Given the length w of a single time slot, the binary variable value $x_{ij}^{kl} = 1$, indicating travel from node v_i^k to v_j^l, cannot occur if $\delta_{ij}^{kl} \geq w - p_i + w(l-k)$; i.e., even if the processing of ship i starts at

the beginning of time slot k, the arrival time in the location v_j^l is beyond the time slot l. In such instances, the edge (v_i^k, v_j^l) and the binary variable x_{ij}^{kl} are omitted from the model. In practice, a large fraction - say about one half - of the binary variables may be omitted.

The splitting approach

One simple heuristic is to split the time horizon T into several sub-intervals and run the model on each sub-interval to construct a part of the route within the entire horizon T. We consider heuristics for choosing such splitting shortly. As a result of splitting, Q sub-intervals are created, and we solve BDTSP separately on each sub-interval T_q, for $q = 1, 2, \ldots, Q$. For the first sub-interval T_1, v_0 is the depot node to start with, and for the last sub-interval T_Q, the flow is forced to turn back to the depot node v_0. For each sub-interval T_q with $q < Q$ the flow is forced to end at some node v_j^l for $j \neq 0$ such that time slot l is in T_q. Since the route over a sub-interval will be connected to the end of the route generated for the preceding sub-interval, for all sub-intervals T_q with $q > 1$ the flow begins in the location of terminal node of the preceding sub-interval T_{q-1}, and the time slot when service is completed for that terminal node. These rules imply minor modifications in the depot node conditions (3) and (5). Also the service completion time from the preceding sub-interval T_{q-1} determines the start time in (8). For brevity, we omit further discussion on such straightforward reformulations.

For each q, after solving BDTSP on sub-interval T_q, all ships $i > 0$ visited during T_q will be removed from the the original set of n ships so that they are not visited again in the subsequent sub-intervals. Let n_q denote the number of remaining ships in sub-interval T_q, i.e., ships present in the work area sometime during T_q, but not visited before the beginning of T_q. After solving BDTSP on all sub-intervals T_q, the sum of travel distances and total number of ships visited are calculated yielding a near-efficient solution (α, z) for the bi-objective problem.

For each sub-interval T_q, for choosing the number $\alpha \in \{1, 2, \ldots, n_q\}$ in (2), obviously a number of alternative strategies may be employed. However, we restrict our discussion below in a strategy aiming to find a near-efficient solution on the entire horizon T such that the total number of ships visited is possibly large. This is justified from the perspective of emissions control needs. Such strategy is also studied in the computational comparisons reported in Section 4.

Time splitting

Splitting of the time horizon T will remove links between nodes that fall into distinct sub-intervals. In general, the splitting approach leads to sub-optimality over the entire horizon T. Therefore, we wish to detect those links, which have low chance of being a part of the global optimal solution. Based on the dataset properties, we try to find good split points in the sense that optimization over sub-intervals would only mildly violate the global optimality, and yet it would help reducing the time complexity considerably. For this aim, we define a criterion $\mathscr{S}(k)$ for measuring the sparsity of ship distribution in

the working area at each time slot k. We call it the sparsity function, defined as follows:

$$\mathscr{S}(k) = \frac{1}{r^2(k)} \sum_{\substack{i,j=0 \\ (i \neq j)}}^{n} d_{ij}^{kk}, \tag{9}$$

where $r(k)$ is the number of ships present in the work area in the time slot k. In fact, $\mathscr{S}(k)$ is proportional to the average pair-wise distances of ships present in the time slot k. Intuition suggests that the extreme values of the sparsity function are relevant to choosing good split points. When the function value is maximal, the ships may be concentrated in a small area. This would minimize excessive travel distance of the surveillance vessel between the end node of the previous sub-interval and the starting node of the next sub-interval. On the other hand, when the split point is selected at the minimum value of the sparsity function, more ship movements are concentrated inside the sub-intervals, thus the optimal edge will not probably be a link to a faraway future time slot. This could give more room for optimizing corresponding optimal routes. However, these are not general rules as the location of best split point might depend on other characteristics of the traffic pattern during one day. For example, if a large number of ships arrive in the next split, we make this unseen by cutting the link to those ships. To discover the best candidate for a split point, we experiment with varying degree of sparsity, ranging between the extreme points of the sparsity function (9). Results are discussed in Section 4.

Finding a near-efficient route visiting many ships

Again, as a result of splitting, assume Q sub-intervals T_q, for $q = 1, 2, \ldots, Q$, are created. We solve the routing problems one by one for sub-intervals T_q, and let n_q denote the number of ships present in sub-interval T_q after removing those already visited before T_q. When the model on T_q is solved, the value of α is initially fixed to n_q, and this maximum value is assigned to the equation (2). In case no feasible solution is found, the value of α is reduced by one in the next trial. Such iterations continue until an optimal solution is found for the BDTSP on sub-interval T_q. The next sub-interval starts after the processing ends at the last visit in T_q. Hence, a continuous route is generated after optimal solutions are found for all sub-intervals. Finally, the total travel distance z and the total number of ships visited α are calculated over the entire horizon T.

4. Computational results

The objectives and outline of this section are as follows. After describing the empirical setting, we show illustrative examples of Pareto solutions, depict an efficient frontier, and demonstrate computation of some efficient points. The overall objective of this section is to compare the heuristics against exact solution in terms of computing time and accuracy of the heuristics (relative distance from the efficient frontier).

For the exact MILP model and for the splitting approach we use the CPLEX 11.2.1 solver on an Intel(R) Core i7 6500U @ 2.5GHz with 12GB of RAM on Windows 64 bit operating system. The MIP relative gap tolerance is set to 0.01 with other default

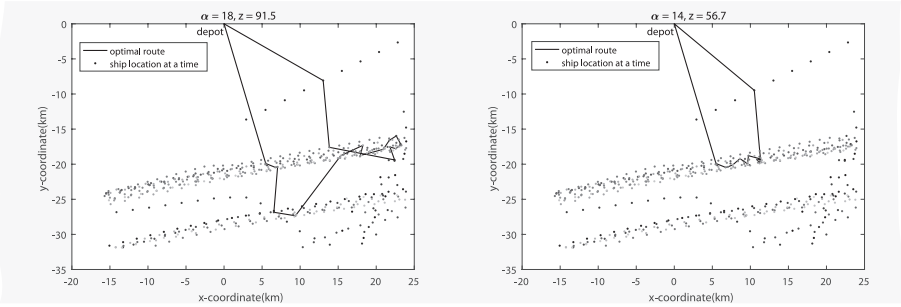

Figure 2. Pareto optima for *July* 5 dataset with 29 ships present in the working area during T=[8:10,14:30] UTC+3. Circles show the real locations of ships in the working area at different time slots (nodes of the graph). Black line shows the optimal route. α is the goal for the number of visits and z is the corresponding minimum travel distance (km)

settings for the solver in all experiments. The time limit is set to 4 hours for CPLEX on the complete time horizon (to find an optimal solution within given tolerances without splitting) and to 10 minutes for sub-intervals in the splitting approach.

The experiments are done on seven real-world data sets for which AIS (Automatic Identification System) data of all ships navigating in the Gulf of Finland are collected via open interface of the Finnish transport infrastructure agency (Väylävirasto, 2019). Based on the real locations of ships, the geographic coordinates are predicted for every minute over 11 hours per day for one week in July 2018. The prediction model of Virjonen et al. (2018) is used for location estimation. The length of time slots is set to $w = 5$ minutes ($m = 132$ time slots for one day). The processing time $p_i = 3$ minutes for all ships i and speed limit of 25 knots (46.3 km/h) for the surveillance vessel are used in all experiments. We do not restrict the network to any distribution or traffic density assumption.

As an illustrative example, *July* 5 dataset is chosen with $n = 29$ ships present during the time horizon T=[8:10,14:30] equivalent to $m = 76$ time slots. Two Pareto solutions are depicted in Fig. 2 and the Pareto frontier for (α, z) is plotted in Fig. 3. As expected, the minimum distance z as a function of the number of measurements α is increasing. However, some of the efficient points are not supported due to violation of convexity. We use equality in constraint (2) to get all Pareto points including the non-supported points.

In order to study the performance of the splitting approach, we carry out two sets of experiments from small to moderate scale problems with 43 and 86 time slots m. The Pareto solutions are computed using the exact method for the maximum possible α values as well as for some smaller values used for the comparisons with the splitting approach. Finding the maximum possible α for the exact MILP model is done by iteratively solving the model for different values of α; this is a highly time consuming task for large problems, since long execution time is also needed for each α to prove infeasibility. For

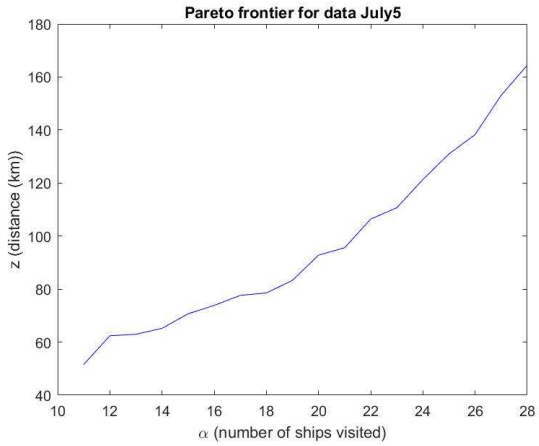

Figure 3. Pareto frontier for *July* 5 dataset with 29 ships present during T=[8:10,14:30] UTC+3

the comparisons, we also report solutions for each dataset with different scenarios using the splitting approach, introduced at the end of Section 3 for finding solutions in case of large α values.

Based on the exact model, solved by CPLEX, for the small and moderate scale problems, each with seven datasets of July, Tables 1 and 2 show efficient solutions; the total distance (km) z travelled, and the number of measurements α. The *Nodes* column indicates the number of nodes $|\cup_{k\in\{0,\ldots,m\}} \mathscr{N}^k|$ in the generated graph; *Targets* is the number n of ships present in the working area during the chosen time horizon; *Time* is the solver CPU time in seconds. Note that within the time horizon, it is not possible to visit all n ships; i.e., we have $\alpha < n$ in all cases.

For the splitting approach, based on the sparsity function $\mathscr{S}(k)$, we adopt three scenarios for comparison to identify good candidates for split points. The scenarios are defined for $Q = 2$, splitting the time horizon T into two sub-intervals. To avoid too short or too long sub-intervals, some lower and upper limits l and u are chosen for split points k as follows: the limits for small problems are set to $l = 19$ and $u = 31$, and for moderate scale problems to $l = 30$ and $u = 60$. The three splitting scenarios are the following:

Scenario 1**.** Split point k maximizes $\mathscr{S}(k)$ in the range of $l \le k \le u$,
Scenario 2**.** Split point k minimizes $\mathscr{S}(k)$ in the range of $l \le k \le u$,
Scenario 3**.** Split point k is given by the mid-point $k = (l+u)/2$.

Scenario 3 serves as a benchmark for verifying possibly enhanced quality of solutions of the *Scenarios* 1 and 2, which are based on the extrema of $\mathscr{S}$. As an example, Fig. 4 shows the fluctuations in the number of ships $r(k)$ present in time slot k and the sparsity function $\mathscr{S}(k)$ for *July* 5 dataset over the time horizon T=[8:10,14:30].

Table 1. Efficient solutions for seven datasets of July 2-8, 2018, with 43 time slots (small scale problems)

Dataset	*Nodes*	*Targets*	α	z (km)	*Time* (s)
July 2	289	19	15	76.9	2.3
			14	65.9	1.9
			13	55.9	1.7
			12	50.4	2.6
July 3	421	27	20	98.8	9.7
			19	80.6	6.1
July 4	271	19	14	74.6	2.4
July 5	340	22	19	107.1	10.3
			18	87.7	9.4
July 6	192	21	9	56.0	0.7
July 7	437	25	19	107.1	2616
			18	73.9	268
			17	68.4	52.6
			16	60.0	6.2
July 8	360	28	16	93.8	1110
			15	67.7	8.8
			14	58.9	5.1
			13	54.4	5.8

Table 2. Efficient solutions for seven datasets of July 2-8, 2018, with 86 time slots (moderate scale problems)

Dataset	*Nodes*	*Targets*	α	z (km)	*Time* (s)
July 2	507	25	24	139.4	15.2
			23	104.1	13.5
			22	92.2	12.6
July 3	775	44	35	174.1	11865
July 4	500	30	25	180.7	359
			24	88.8	5.42
July 5	725	38	34	150.4	1382
			33	135.5	4211
July 6	619	33	25	159.0	2359
			24	129.4	1280
July 7	683	35	31	156.9	3020
			30	133.9	2747
			29	122.2	2543
July 8	748	45	33	146.8	6489
			32	128.0	2495
			31	112.7	1364

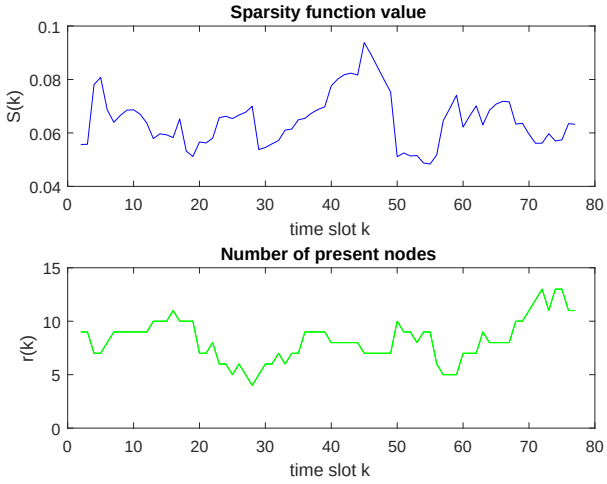

Figure 4. Characteristics of July 5 dataset for 29 present ships during T=[8:10,14:30] UTC+3 ($m = 76$ time slots). The horizontal axis shows the time slots k. The upper diagram is the value of $\mathscr{S}(k)$. The lower diagram is the number of ships $r(k)$ present in each time slot k

Computational results for the three splitting scenarios are shown in Table 3 for small problems and in Table 4 for moderate scale problems. Given α, the *Speed-up* factor is defined as the CPU time taken by CPLEX to solve the exact model (column *Time* in Tables 1-2) divided by the time taken by the splitting approach. According to the results, the splitting approach finds values of α very near to the maximum levels reported in Tables 1 and 2. The decrease in computational time is highly significant with respect to the time required for solving the exact problem (see *Speed-up* columns in Tables 3 and 4). Furthermore, in general, the splitting approach frequently returns near-efficient distance z; see columns $\rho = (z - z^*)/z^*$ in Tables 3 and 4, showing the relative deviation (%) of distance z from the exact minimum distance z^*. Results obtained on larger scale problems of Table 4 show in most cases the practicality of using the splitting approach in terms of computing time for finding a near-maximum α, in addition to a near-efficient distance value z. The largest problem solved among these experiments, in terms of the number of binary variables, is related to *July* 3 dataset, using 86 time slots with 286,819 binary variables. As can be seen in Table 4, *Scenario* 2 returns a near-optimal solution while reducing the solution time from 3 hours and 18 minutes to 2 minutes, thus achieving the speed-up factor of 99. Similarly, for *July* 4, both *Scenario* 1 and 2 found in few seconds a near-efficient solution with 1 to 2 % relative deviation. The results of Table 3 and Table 4 provide us with a perspective for computational speed-up in case of larger sets of test data. Thus, extra computations may not yield extra value to our results. Furthermore, for our practical purpose, the small to moderate scale examples are the case, whose speed-up level is sufficient.

Table 3. Three split scenarios for seven datasets of July 2-8, 2018, with 43 time slots (small scale problems); $\rho = (z - z^*)/z^*$ (%) is the relative distance from the efficient frontier

	Scenario 1			*Scenario* 2			*Scenario* 3		
Dataset	α	ρ (%)	*Speed-up*	α	ρ (%)	*Speed-up*	α	ρ (%)	*Speed-up*
July 2	**13**	**50**	1.0	**12**	**47**	2.2	12	87	2.2
July 3	**19**	**12**	1.7	19	15	1.7	**6**	**6**	1.1
July 4	**14**	**1**	2.7	14	2	3.4	14	16	3.4
July 5	**19**	**0**	2.3	18	19	5.2	**18**	**1**	5.2
July 6	9	9	1.4	**9**	**0**	1.4	9	6	1.7
July 7	**18**	**1**	39.9	16	33	2.8	18	31	29.1
July 8	**15**	**22**	4.9	**13**	**51**	3.4	14	16	3.4
Average	**15.3**	**13.5**	7.7	14.4	24.0	2.9	15.0	30.4	6.6

Regarding the good candidate for a split point, let first consider an aggregate comparison based on each criterion separately: the average of ρ (the relative distance from the efficient frontier) and the average of α (the number of ships visited). The average values are shown in the last rows of Tables 3 and 4. Results show that *Scenario* 1 dominates *Scenarios* 2-3 in small examples, but not in moderate scales examples; *Scenario* 2 has less relative distance from the optimum with the same value of visited nodes on average. Therefore, we cannot rank any of the Scenarios over the others based on the separate comparisons for individual objectives.

Taking into account both objectives z and α in a pairwise comparison, it is less clear which among the three solutions is most qualified. By the term *solution quality* we refer to a characteristic, which indicates the answer to the following question: *Is the greater number of visits we achieve in one solution worth the amount of increase in travel distance compared with the other solution?* As an example, we consider *July* 8 dataset in Table 3, where two solutions (α, z), $(15, 82.4)$ and $(13, 82.1)$, are obtained by the first two scenarios. Although the former does not dominate the latter, the solution of *Scenario* 1 is likely to be preferable, since by a relatively small increment in the distance z, two additional measurements in α are possible. Non-dominated solutions[§] are found by different *Scenario*s for each dataset. Among the solutions shown in Tables 3 and 4, the non-dominant solutions in both tables are indicated in boldface. In the following, based on a unary indicator within a dataset, we would like to find out which scenario returns the most qualified solutions among all.

In general, we need a solution quality indicator, a value function, to identify the

[§]A non-dominated solution here refers to an efficient point in the set of three feasible solutions for a dataset given in Tables 3 and 4, not in the set of all feasible solutions of the bi-objective problem.

Table 4. Three split scenarios for seven datasets of July 2-8, 2018, with 86 time slots (moderate scale problems); $\rho = (z - z^*)/z^*$ (%) is the relative distance from the efficient frontier

	Scenario 1			*Scenario* 2			*Scenario* 3		
Dataset	α	ρ (%)	*Speed-up*	α	ρ (%)	*Speed-up*	α	ρ (%)	*Speed-up*
July 2	**23**	**7**	4.2	**22**	**6**	5.2	22	12	6.6
July 3	35	15	188.3	**35**	**2**	99.6	35	5	192.0
July 4	**25**	**1**	74.8	**24**	**2**	2.1	24	8	2.2
July 5	33	17	68.9	**33**	**9**	170.5	33	16	8.7
July 6	**24**	**9**	55.7	**25**	**14**	92.1	24	10	224.6
July 7	29	31	279.5	**29**	**2**	19.9	**30**	**19**	4.6
July 8	**31**	**18**	24.1	32	34	4.1	**32**	**31**	10.8
Average	**28.6**	14.0	99.3	**28.6**	**9.9**	56.2	**28.6**	14.4	64.2

preferred splitting scenario, to be found among the non-dominated cases. Various metrics have been proposed for comparing the quality of Pareto solutions of multi-objective optimization problems; for a review on quality assessment metrics of Pareto points we refer the reader to Riquelme, Lücken and Barán (2015) and Zitzler, Knowles and Thiele (2008). The metrics which we apply for scenario comparisons is a normalized linear scalarization (equivalent to a linear value function) of the two objectives in the bi-objective problem. This measure employs reference points (α_{min}, z_{max}) and (α_{max}, z_{min}), where α_{min} and z_{max} (defining the nadir point) are the worst values in the dataset for each objective, and α_{max} and z_{min} (defining the ideal point) are the best values. The solution quality indicator I_N is defined as follows:

$$I_N(\alpha, z) = \frac{\alpha - \alpha_{min}}{\alpha_{max} - \alpha_{min}} + \frac{z_{max} - z}{z_{max} - z_{min}}. \tag{10}$$

The indicator I_N is an equally weighted sum of distances to the nadir point level of each objective scaled by the difference of ideal and nadir point levels. In equation (10), in case of $\alpha_{max} = \alpha_{min}$ for two given reference points, we simply set the first term equal to zero. Similarly, if $z_{max} = z_{min}$, we set the second term equal to zero. Figure 5 is a schematic representation of Pareto optimal (solid line) and two nearly efficient (dashed lines) frontiers, defining the two reference points. In many applications of rational choice theory, a value function is a useful way of modelling preferences of the DM; however, the equal weighting of the two objectives in the linear value function I_N does not necessarily represent preferences of the DM; the weighting coefficients would need to be measured.

In distinction from what is common in value function applications, here for comparing the solutions obtained by splitting *Scenarios* 1-3, we have to define the solution quality indicator function I_N separately for each data set. Tables 5 and 6 show the values of I_N for each dataset in small and moderate scale cases. The best value of the three scenarios in each dataset is shown in boldface (zeros in the tables are computational

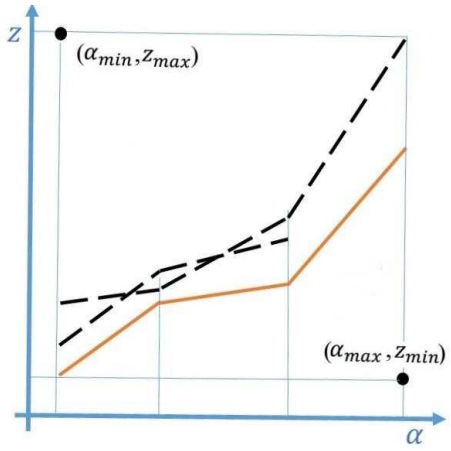

Figure 5. Schematic representation of an optimal (solid line) and two approximation (dashed lines) of Pareto front along with the reference points. The coordinates are the two objectives; α is the number of ships visited and z is the total travel distance (km). The point (α_{max}, z_{min}) is the ideal utopia point of the bi-criteria problem and (α_{min}, z_{max}) is the nadir point

zeros). In a given dataset, a scenario is best if it dominates the other two. Based on indicators I_N, it is evident from the results that choosing the extrema of $\mathscr{S}$ as split points (*Scenarios* 1 and 2) provides better chance of improving the quality of the solution than using the naive splitting (*Scenario* 3). Among the cases in Tables 5 and 6, the indicator I_N ranks *Scenario* 1 as the winner more frequently than *Scenarios* 2 and 3. Therefore, conforming with intuition, the time slots with large average distances among ships are good candidates for split points.

Table 5. Values of indicators $I_N(\alpha, z)$ for three scenarios on small scale problems

Dataset	(α_{max}, z_{min})	(α_{min}, z_{max})	*Scenario* 1	*Scenario* 2	*Scenario* 3
July 2	(15, 50.4)	(12, 94.2)	**0.57**	0.45	0.00
July 3	(20, 80.6)	(19, 104.3)	0.59	0.50	**1.00**
July 4	(14, 74.6)	(14, 86.4)	**0.95**	0.89	0.00
July 5	(19, 87.7)	(18, 107.1)	**1.01**	0.13	0.97
July 6	(9, 56.0)	(9, 60.9)	0.00	**1.00**	0.33
July 7	(19, 60.0)	(16, 107.1)	**1.36**	0.58	0.89
July 8	(16, 54.4)	(13, 98.1)	**1.03**	0.36	0.34

Table 6. Values of indicators $I_N(\alpha, z)$ for three scenarios on moderate scale problems.

Dataset	(α_{max}, z_{min})	(α_{min}, z_{max})	*Scenario* 1	*Scenario* 2	*Scenario* 3
July 2	(24, 92.19)	(22, 139.43)	**1.10**	0.89	0.77
July 3	(35, 174.06)	(35, 200.1)	0.00	**0.90**	0.67
July 4	(25, 88.80)	(24, 182.7)	**1.00**	0.98	0.93
July 5	(34, 135.50)	(33, 158.1)	0.00	**0.44**	0.04
July 6	(25, 129.42)	(24, 180.8)	0.78	**1.00**	0.74
July 7	(31, 122.16)	(29, 160.5)	0.00	**0.93**	0.54
July 8	(33, 112.71)	(31, 172.0)	**0.65**	0.50	0.57

5. Conclusion and further research

In this paper, we consider optimal routing of a surveillance vessel measuring greenhouse gas emissions of large ships in the Baltic sea. We develop a bi-objective MILP model for finding a Hamiltonian circuit in a dynamic network formed by locations of ships over time, maximizing the number of measurement jobs to be done and minimizing the total travel distance of the vessel. As an output, a set of Pareto optimal solutions can be produced. Standard optimization software may be used for computing the efficient frontier; however, cases in practice tend to lead to prohibitively large problems. To deal with the curse of dimensionality, we suggest the model to be computationally tackled with a time horizon splitting approach employing a sparsity function to determine the splitting points. Thereby, near-efficient values for the number of measurement visits to ships and for the total travel distance of the vessel can be obtained. The decrease in computational time by using the splitting approach is highly significant, even for the moderate scale test cases. For moderate size problems the solver execution time frequently decreases from hours to minutes. The characteristics of a good candidate for split point are investigated by introducing a traffic sparsity function and examining the split points based on extrema of the function versus naive splitting serving as a benchmark. According to the results, time slots corresponding to most sparse traffic in the network are good candidates for split points; an observation, which is in harmony with intuition.

As already noted, for large scale problems, exact solutions may not be available. Even if the problem is solvable, for practical purposes it can take too long to determine the solution. However, based on our results, using the splitting approach we can expect fast solutions, which are of high enough quality relative to the efficient frontier. Our experiments have been performed on real historical data sets and evidence confirms that the method can be efficiently used in practice for maritime routing management.

Further research will be directed at testing the model for some real situations in maritime logistics and incorporating the model into a fully autonomous logistics navigation system. Further work might also consist in studying whether the present results can be improved by splitting the time horizon based on other features of the traffic pattern, such

as clusters of ships instead of the sparsity function used here. A natural alternative as a commonly used heuristic for this class of problems could be a genetic algorithm. Dealing with uncertainty in the underlying graph and optimal routing under such uncertainty is another subject for the future. This also leads to extending heuristics to routing and scheduling under uncertainty.

Acknowledgements

This work was supported by the Business Finland grant (Dnro 2040/31/2017), related to the ”New 3D analytics methods for intelligent ships and machines project” via the academic partners: Department of Future Technologies and Department of Mathematics and Statistics, University of Turku. Authors would like to thank Petra Virjonen for providing us with all test datasets and professor Markku Kallio for his valuable remarks and comments, which significantly contributed to the quality of the paper. Authors also thank the anonymous referees for their constructive comments and suggestions that helped to improve the presentation and the content of this paper.

References

ABELEDO, H., FUKASAWA, R., PESSOA, A. AND UCHOA, E. (2013) The time dependent traveling salesman problem: Polyhedra and algorithm. *Mathematical Programming Computation*, **5**, 27–55.

ANDROUTSOPOULOS, K. AND ZOGRAFOS, K. (2009) Solving the multi-criteria time-dependent routing and scheduling problem in a multimodal fixed scheduled network. *European Journal of Operational Research*, **192**, 18–28.

APPLEGATE, D., BIXBY, R., CHVÁTAL, V. AND COOK, W. (2007) *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA, 2007.

BIANCO, L., MINGOZZI, A. AND RICCIARDELLI, S. (1993) The traveling salesman problem with cumulative costs. *Networks*, **23**, 81–91.

BRANKE, J., DEB, K., MIETTINEN, K. AND SŁOWIŃSKI, R. (2008) *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer-Verlag, Berlin, Heidelberg, Germany, 2008.

FOX, K. (1973) *Production scheduling on parallel lines with dependencies*. Ph.D. Dissertation, Johns Hopkins University, Baltimore, Md, 1973.

FOX, K., GAVISH, B. AND GRAVES, S. (1980) An n-constraint formulation of the (time-dependant) traveling salesman problem. *Operations Research*, **28**, 1019–1021.

FURINI, F., PERSIANI, C. AND TOTH, P. (2016) The time dependent traveling salesman planning problem in controlled airspace. *Transportation Research Part B: Methodological*, **90**, 38–55.

GAREY, M. AND JOHNSON, D. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

Github (2020) Github Repository Dynamic-TSP project, Online; accessed June 2020.

GROBA, C., SARTAL, A. AND VÁZQUEZ, X. (2015) Solving the dynamic travelling salesman problem using a genetic algorithm with trajectory prediction: An application to fish aggregating devices. *Computers and Operations Research*, **56**, 22–32.

HELVIG, C., ROBINS, G. AND ZELIKOVSKY, A. (2003) The moving-target traveling salesman problem. *Journal of Algorithms*, **49**, 153–174.

HEWGLEY, C. AND YAKIMENKO, O. (2012) Precision guided airdrop for vertical replenishment of naval vessels. *20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar,* May 2009, Seattle, Washington, 2012.

LUCENA, A. (1990) Time-dependant traveling salesman problem-the deliveryman case. *Networks*, **20**, 753–763.

MALANDRAKI, C. AND DASKIN, M. (1992) Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, **26**, 185–200.

MARLOW, D., KILBY, P. AND MERCER, G. (2007) The travelling salesman problem in maritime surveillance-techniques, algorithms and analysis. *Proceedings of the International Congress on Modelling and Simulation*, 2007, 684–690.

MIETTINEN, K. (1998) *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Springer, Boston, MA, 1998.

MIKKONEN, M. (2019) Järjestö: Maailman suurimman risteilyvarustamon laivat päästivät Euroopassa rikkiä kymmenen kertaa enemmän kuin kaikki autot yhteensä. *Helsingin Sanomat*, published June 8, 2019.

PAPADIMITRIOU, C. AND STEIGLITZ, K. (1982) *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., NJ, USA, 1982.

PICARD, J. AND 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.

REGO, C., GAMBOA, D., GLOVER, F. AND OSTERMAN, C. (2011) Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, **211**, 427–441.

RIQUELME, N., LÜCKEN, C. AND BARÁN, B. (2015) Performance metrics in multi-objective optimization. *Latin American Computing Conference (CLEI)*, 2015.

TAŞ, D., GENDREAU, M., JABALI, O. AND LAPORTE, G. (2016) The traveling salesman problem with time-dependent service times. *European Journal of Operational Research*, **248**, 372–383.

VANDER WIEL, R. AND SAHINIDIS, N. (1996) An exact solution approach for the time-dependent traveling-salesman problem. *Naval Research Logistics*, **43**, 797–820.

VIRJONEN, P., NEVALAINEN, P., PAHIKKALA, T. AND HEIKKONEN, J. (2018) Ship movement prediction using k-NN method. *Proceedings of the Baltic Geodetic Congress (Geomatics)*, Olsztyn, Poland, 2018, No. 56.

VU, D., HEWITT, M., BOLAND, N. AND SAVELSBERGH, M. (2019) Dynamic Discretization Discovery for Solving the Time Dependent Traveling Salesman Problem with Time Windows. *Transportation Science*, **54**, 703–720.

VÄYLÄVIRASTO (2019) Väylävirasto (Finnish Transport Infrastructure Agency), vayla.fi. Online; accessed October 2019.

WIERZBICKI, A. (1982) A mathematical basis for satisficing decision making. *Mathematical Modelling*, **3**, 391–405.

ZITZLER, E., KNOWLES, J. AND THIELE, L. (2008) Quality Assessment of Pareto Set Approximations. In: J. Branke et al., eds.,: *Multiobjective Optimization*, **LNCS 5252**, 373-404, Springer- Verlag Berlin Heidelberg, 2008.