Alaleh Maskooki | Yury Nikulin

Multiobjective Efficient Routing In a Dynamic Network

Turku Centre *for* Computer Science

TUCS Technical Report
No 1198, June 2018

Multiobjective Efficient Routing In a Dynamic Network

Alaleh Maskooki
University of Turku, Department of Mathematics and Statistics
Vesilinnantie 5, Turku, Finland FI-20014
alamas@utu.fi
Yury Nikulin
University of Turku, Department of Mathematics and Statistics
Vesilinnantie 5, Turku, Finland FI-20014
yurnik@utu.fi

TUCS Technical Report
No 1198, June 2018

Abstract

The paper presents a bi-objective integer programming model for routing and scheduling in a time-dependent network, where edge weights vary with time. It can be considered as an extension of the network flow model for the time-dependent travelling salesman problem. The objective is to find an algorithmic solution for the optimal sequence of location/time nodes which gives the shortest travel distance, with maximum number of visits. A heuristic algorithm is proposed based on the bi-objective integer programming model, for time splitting. The performance of the algorithm on real large scale sets are evaluated. The results of this research can be used in various logistic applications specifically maritime management services.

Keywords: Time-dependent network, Dynamic TSP, Single-machine scheduling, Bi-objective programming

TUCS Laboratory
Turku Optimization Group

1 Introduction

In many real life applications, there is a need to construct 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. The fundamental problem is known as travelling salesman problem (abbreviated as TSP). It is an NP-hard [1] problem in combinatorial optimization that is of great importance also in operations research, network logistics and computer science [2]. Classical formulations of TSP include integer linear programs but the presence of an exponentially growing number of subtour elimination constraints makes it intractable for solving by exact methods, for example branch and bound type of algorithms that can only process networks up to 40-60 nodes on average [3]. Some significant improvement can be achieved whenever branch and bound methods are equipped with ad-hoc cutting plane generation techniques [4] allowing us to calculate exact optimum for the middle size networks up to one hundred thousand nodes with the help of supercomputers. The usage of modern heuristics allows to process network of size up to a million nodes on standard CPUs, and such instances can be resolved to optimality with 2-3 percentage accuracy [5].

Classical TSP has many variations including the one where nodes have dynamically changing locations. This feature introduces a new level of complexity since now the decision has to be made on not only node optimal sequencing but also on time that every node must be visited. Time dependent nature of routing has to be properly addressed in such models (see e.g. [6, 7]). As for applications, dynamic TSP formulations appear very often in transportation and logistics, including maritime logistics analytics [8]. In this report, we define the bi-objective dynamic TSP as follows:

> *"Assume a variant of TSP problem, defined in a dynamic network of n nodes, whose locations change during the time. The moving nodes can appear in the area of measurement in different times, move along a random route and exit the area after a while. The question is how to travel among moving targets, such that we visit as many nodes as we can, with the shortest possible distance"*

The above optimization problem raised from a real logistic service problem for optimizing the navigation of a service boat, measuring vessels' gas emission. The problem is a new variant of transportation in a dynamic network due to its unique features and goals, and cannot be solved by any of the existing Mixed Integer Linear Programming (MILP) models for the time-dependent TSP. The present model in this research, can be considered as an extension of Picard and Queyranne [10] formulation, whose associated polytope is studied by Abeledo et al. [11].

In addition to combinatorial complexity, the presence of more than one optimization goal, needed to be achieved on the way to accurate and adequate mod-

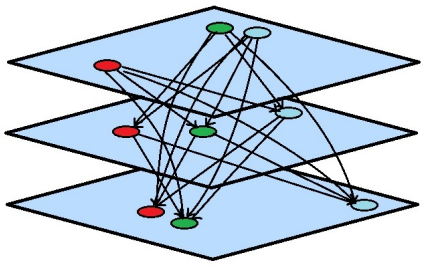

Figure 1: Representation of graph $\mathcal{G}$ with $n = 3$ vertices and three time slots.

elling, may influence problem complexity. It is well-known that under multi-objective framework a solution that is optimal with respect to one objective may have arbitrarily bad value of the others, and thus, may be unacceptable for a decision maker in practical situations. Therefore, many problems arising in optimization, applied mathematics and operations research should be ultimately considered under multi-criteria framework due to existing of several conflicting goals or interests [9].

2 Problem formulation

Let $\mathcal{N}^t = \{v_i^t | i \in \{1, ..., n\}, t \in T\}$ be the set of n nodes whose locations depend on time $t \in T$, where T is the total time interval during which visiting a node is permitted, and $\mathcal{A}^{tl}$ is the set of directed edges connecting nodes in a pair of times t and l, with sources in $\mathcal{N}^t$ and terminals in $\mathcal{N}^l$. Assume $\mathcal{G} = (\cup_{t \in T} \mathcal{N}^t, \cup_{(t,l) \in T^2} \mathcal{A}^{tl})$ is a multi-partite directed graph consisting the union of all sets of nodes $\mathcal{N}^t$s and their pairwise connecting edges. The schematic representation of graph $\mathcal{G}$ is depicted in Figure 1. The problem is to find an itinerary with the shortest route length within a specified time interval, that passes through as many nodes as possible with a constant speed u, while every visit needs a processing time p_i before leaving and visiting the next node.

In order to formulate the above problem as a combinatorial problem, we need to discretize the time interval T into a finite set of m time slots, indicated by $\mathcal{K}$, where the location of nodes are supposed to remain unchanged in each time slot. Suppose that processing more than one node is not possible in a single time slot. Thus, we define $s_k \in [t_k, t_{k+1}]$ to be the starting time for processing a node during the time slot $k \in \mathcal{K}$. Assume a depot node v_0, which is not moving during the time, to be the starting and ending points of the salesman's tour. A feasible tour, consisting of a sequence of $\{v_i^k\}$ can be generated by the following set of constraints, in terms of binary variable x_{ij}^{kl}, which is equal to 1 if node v_j^l is visited immediately after node v_i^k; and 0 otherwise:

$$\sum_{k=1}^{m-1} \sum_{i=1}^{n} x_{0i}^{0k} = 1 \tag{1}$$

$$\sum_{k=1}^{l-1} \sum_{\substack{i=0 \\ (i\neq j)}}^{n} x_{ij}^{kl} = \sum_{k=l+1}^{m} \sum_{\substack{i=0 \\ (i\neq j)}}^{n} x_{ji}^{lk} \qquad \forall j = 1...n \quad \forall l = 2...m-2 \tag{2}$$

$$\sum_{k=1}^{m-1} \sum_{i=1}^{n} x_{i0}^{km} = 1 \tag{3}$$

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

$$\sum_{k=0}^{l-1} \sum_{\substack{i,j=0 \\ (i\neq j)}}^{n} x_{ij}^{kl} \leq 1 \qquad \forall l = 0...m \tag{5}$$

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

$$\sum_{\substack{k,l=0 \\ (l>k)}}^{m} \sum_{\substack{i=0 \\ (i\neq j)}}^{n} x_{ij}^{kl} + \sum_{\substack{l,k=0 \\ (k>l)}}^{m} \sum_{\substack{i=0 \\ (i\neq j)}}^{n} x_{ji}^{lk} \leq 2 \qquad \forall j = 0...n \tag{7}$$

$$x_{ij}^{lk} \in \{0,1\} \qquad \forall i,j = 1...n\,(i \neq j) \quad \forall k,l = 1...m\,(k < l) \tag{8}$$

Initially, the flow starts from depot node v_0. If the decision is made to move to i^{th} node at time slot k, then $x_{0i}^{0k} = 1$. So, we have the first constraint (1). after reaching the node v_i^k, the flow should exit that node at some later time slots l. So, initialized by the previous constraint, a unit of flow propagates throughout two time slots k and l according to the balance constraints (2) (flow conservation). The flow turns back to depot node v_0 at some time slot k using constraint (3). The inequalities (4) and (5) prohibit entering to or exiting from a single node at multiple time slots respectively. Constraints (6) and (7) restrict the flow to pass through every node j at most once. Furthermore, constraint (2) maintains the connection of time slots and (6) eliminates route breaks through the time slots sequence. Inequality (7) acts as a subtour elimination constraint. So, we start from the depot, visit a sequence of nodes along the route $\{(v_i^k, v_j^l) | x_{ij}^{kl} = 1\}$ of length at least 2 and turn back to the depot. However, to create a Hamiltonian circuit of maximum length, we need to define a parameter α to indicate the number of nodes that should be visited. Thus, the objective of maximizing the number of visits can be defined as the following inequality, and be added to the above set of constraints:

$$\sum_{\substack{k,l=0 \\ (l>k)}}^{m} \sum_{\substack{i,j=0 \\ (i\neq j)}}^{n} x_{ij}^{kl} \geq \alpha \tag{9}$$

The primary objective is to minimize the travel distance, using the following objective function:

$$z = \sum_{\substack{k,l=0 \\ (l>k)}}^{m} \sum_{\substack{i,j=0 \\ (i\neq j)}}^{n} d_{ij}^{kl} x_{ij}^{kl} \tag{10}$$

where d_{ij}^{kl} is the distance between nodes v_i^k and v_j^l in $\mathcal{G}$. This is one way to handle multi-objective problems. The idea is referred to as "goal programming". That is, we set a goal for one (or several) objective value(s) and find the optimal solution for the other objective while meeting the goal(s). Here, the secondary objective (maximizing the number of visits) is used as constraint to generate the Pareto frontier. If we want to treat all objectives equally, we may formulate them as flexible constraints and use minimization of their corresponding deviation variables. Another way of dealing with multi objectives is to give each objective a weight and minimize the weighted aggregation. It can be proved that, if x is a Pareto optimal solution for the problem, then there exist multipliers for the weighted Chebyshev aggregation which gives the same Pareto optimal solution x. In case the weighted sum scalarization is used, the calculation is more efficient but only the supported Pareto points are found. More information on multi-objective optimization methods can be found in [13].

In order to create a time dependent route, we need to force visits occurring within the given time window. In other words, if some processing starts at time s_k then the start time should occur within the k^{th} time slot, and as a result satisfy the following constraint:

$$s_0 + wk \leq s_k \leq s_0 + w(k+1) \qquad \forall k = 0...m \tag{11}$$

where s_0 is the start time at the first time slot (can be set to the time when working starts), and w is the length of each time slot. In addition, enough time should be reserved for processing and travelling to the next node before each visit, which can be defined by the following scheduling constraint:

$$s_k + \sum_{\substack{i,j=0 \\ (i\neq j)}}^{n} (p_i + t_{ij}^{kl}) x_{ij}^{kl} \leq s_l \qquad \forall k, l = 1...m\,(k \neq l) \tag{12}$$

where t_{ij}^{kl} is the time takes to travel along the edge (v_i^k, v_j^l) with a specified constant speed.

The solution of the introduced MILP model will return the shortest distance for visiting at least α moving nodes, as well as start times of each visit in a given time interval T. The model contains $m^2 + n(m-1) + 2m + 8$ constraints. The upper bound for the number of variables is n^2m^2. However, it depends on how long each node is present in the area. The model returns the Pareto optimal solutions for the defined problem. It can be solved by branch-and-bound and cutting plane

methods integrated into standard MIP solvers, however, the large number of binary variables and constraints make it inefficient when using on large-size real datasets. In the next section, some heuristics on time and locations are applied which returns an efficient solution within an acceptable execution time.

3 Heuristic solution algorithm

When there are lots of nodes moving simultaneously in the area, the number of possible links and thus the number of feasible routes grows enormously. To formulate an efficient heuristic, we need to detect those links which have low (or zero) chance of being included to the optimal solution. First of all, we remove links to those nodes which are not accessible in their assigned time slots. That is, assuming $dt_{ij}^{kl} = d_{ij}^{kl}/u$ to be the time takes to travel between nodes v_i^k and v_j^l with the constant speed u, then the quadruples (i, j, k, l) is included only if:

$$(l - k)w + \epsilon_i \geq dt_{ij}^{kl} \tag{13}$$

The tolerance ϵ_i is added because processing can occur at any moment within the interval of a time slot. It can be adjusted according to the processing time, for instance $\epsilon_i = w - p_i$. We also define a threshold τ for the maximum length of edges that can be included in the search scope. The value of τ depends on the distances and is determined by trial and error.

Splitting the time interval: One simple heuristic is to split the whole interval T and construct a part of the route limited to each sub-interval. In experiments, we see that by splitting the total time interval, although we may achieve a near optimal number of visits, but it can violate the optimal travel distance considerably. Therefore, we need to find good split points in the sense that it should not violate the global optimality extremely, and yet it should help reducing the complexity considerably. For this reason, we define a criterion $D(k)$ for measuring the density with respect to the sum of distances per square number of nodes in each time slot k as follows:

$$D(k) = \frac{1}{r^2(k)} \sum_{\substack{i,j=0 \\ (i \neq j)}}^{n} d_{ij}^{kk} \tag{14}$$

where $r(k)$ is the total number of nodes present during the k^{th} time slot. The status of being present for a node is defined by the existence of its location/time information. Thus, when nodes are out of the measurement area there is no data for them and its link to other (present) nodes are removed so that they are not included in the underlying network. If the value of $D(k)$ is small and $r(k)$ is relatively big in a time slot k, it means that most nodes are located near each other, and thus there is a high chance of achieving a bigger α which increases the computational complexity. If we split the interval at dense points, we can probably reduce the execution time significantly but it violates the optimal travel distance. Best candidates for split points are waiting time slots of the optimal route which are unknown. However, they are more probable to happen when no or few nodes are around and others are

distant. Therefore, good candidates for split points are those with maximum values of $D(k)$ which implies that nodes are distant from each other when $r(k)$ is locally minimum or small.

4 Computational performance

In order to validate the performance of the introduced heuristic and comparing it with the exact method, we implement the algorithm on real large-size datasets, taken from [12], in which geographic coordinates of vessels during a 16-hour interval are recorded for several days. The experiments are done on three sets of data belonging to 21^{st} and 12^{th} of January, and 5^{th} of July 2018, with 192 time slots (every 5 minutes during 16 hours) and total number of 42, 60 and 29 vessels passing through the measurement area respectively. For the data of January, the interval T is splitted into 6 sub-intervals according to the local maxima of $D(k)$ and local minima of $r(k)$. Shorter sub-intervals are chosen for time slots with relatively large number of present nodes in the beginning of the time interval T, where there are lots of unvisited nodes. The diagram of $D(k)$ at a scale of $1000/1$ and $r(k)$ for the two datasets of January are illustrated in Figure 2. Vertical lines show split points of the interval T. The processing time $p = 3$ minutes, and constant speed of 46 km/h is used in the experiments.

As the input information including time window, stationary point v_0, location/time information of target nodes (of those which are missed in the previous round but still present, or those which arrive later in the area) and the value of α differ in each sub-interval, we need to update input data before each solving of the model. The output of each solution are the Pareto optimal travel distances with respect to each α, a sequence of visited nodes with their location/time information, start times of each process and a list of nodes which are not visited. The stationary point in each new sub-interval is supposed to be the location/time node corresponding to the last visit in the previous sub-interval. The execution time includes calculation of pairwise distances to generate the network and solving the model for a given number of targets. The number of targets is supposed to be the number of present nodes in a given sub-interval which are not yet visited. The number of targets is reduced by one in each iteration and is assigned to α value. The algorithm stops when a feasible solution is found for the given α. Tables 1 and 3 show total distance travelled for maximum possible α for consecutive sub-intervals related to data of $Jan.21$ and $Jan.12$ respectively.

In the second set of experiments, we investigate the effect of splitting. Tables 2 and 4 show the maximum number of visits and their corresponding travel distances, when less number of splits are generated for the same data of $Jan.21$ and $Jan.12$. In order to reduce the number of split points, we merge each two consecutive sub-intervals to see the difference of maximum number of visits and corresponding distances between the merged and splitted intervals.

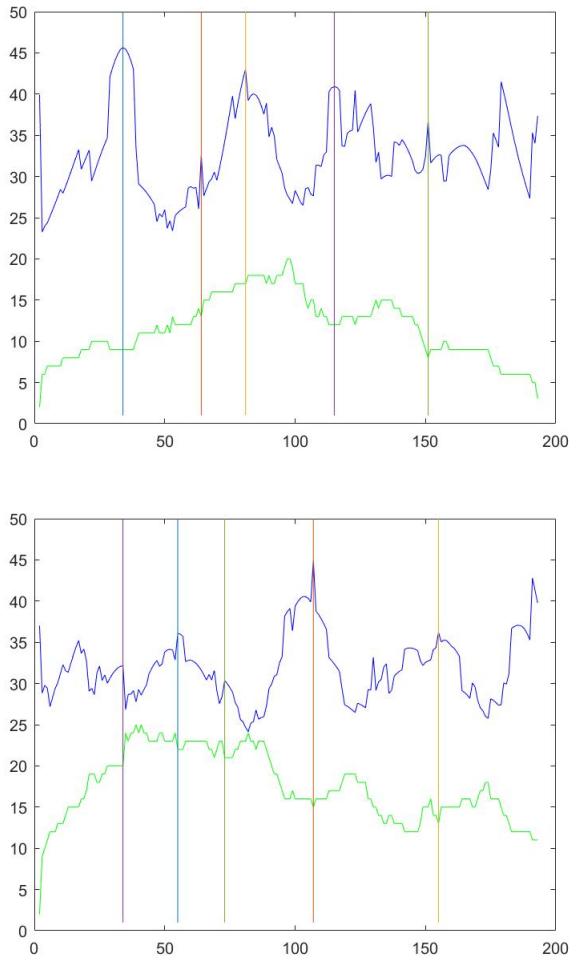

Figure 2: The upper and lower figures show status of $Jan.12$ and $Jan.21$ datasets respectively. The horizontal line shows the time slots. Vertical lines show five split points of the interval T. The blue diagram is the value of $D(k)$ at a scale of $1000/1$. The green diagram is the number of nodes $r(k)$ present in each time slot.

Table 1: Maximum possible α and their travel distances for 6 sub-intervals related to data of $Jan.21$, 2018

Interval	No. of Targets	Max. No. of Visits (α)	Total Distance (km)	Exe. Time (min:sec)
0-34	9	7	78.717	0:16
35-64	14	9	37.086	0:16
65-81	9	4	50.687	0:11
82-115	12	11	76.082	0:21
116-151	9	6	83.809	0:23
152-192	4	4	28.084	0:10
Entire day	42	41	354.465	1:37

Table 2: Maximum possible α and their travel distances for 3 sub-intervals related to data of $Jan.21$, 2018

Interval	No. of Targets	Max. No. of Visits (α)	Total Distance (km)	Exe. Time (min:sec)
0-64	22	18	189.850	6:38
65-115	15	14	115.908	1:17
116-192	10	10	64.391	1:14
Entire day	42	42	370.465	9:09

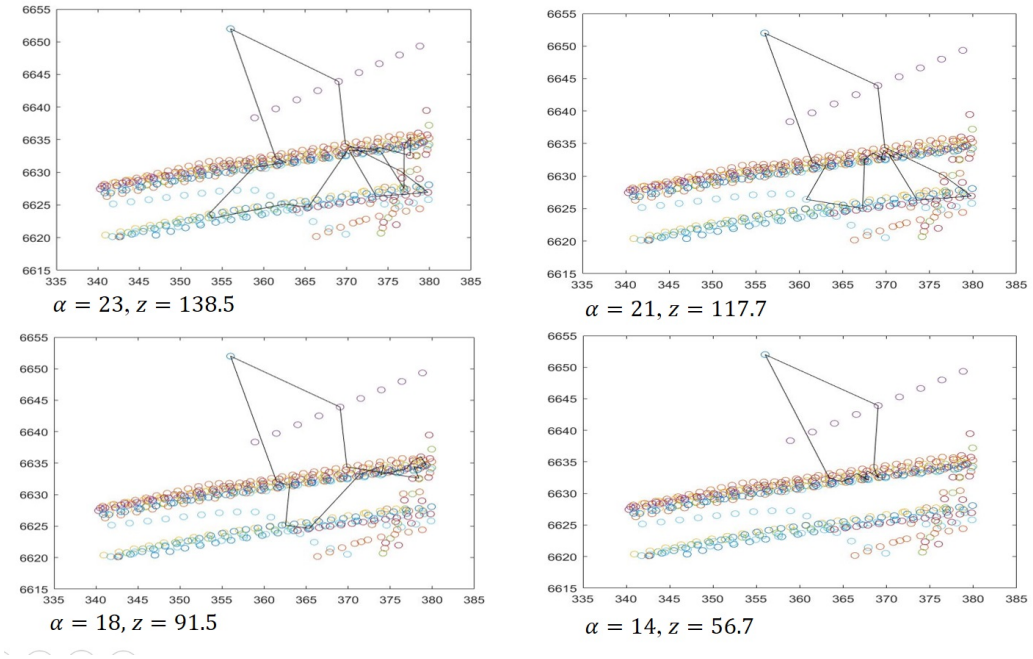

Figure 3: The optimal route for the corresponding maximum visits before changing the current line for data of $Jul.5$, 2018.

For instance, splitting the interval $[0, 64]$ into $[0, 34]$ and $[35, 64]$ for the data of $Jan.21$ (Tables 1 and 2) results in losing two visits. In order to find out the difference between travel distances for the same number of visits, we also calculated total travel distance corresponding to 16 visits for the interval $[0, 64]$ to be comparable with the splitted interval. The travel distance for the whole interval $[0, 64]$ without splits is 105.34 km for 16 visits. Comparing to the distance 115.80 km which is obtained by the sum of two distances for $[0, 34]$ and $[35, 64]$, there is about 10 km extra travelling caused by one time splitting. Similar results are obtained for $Jan.12$ dataset, shown in Tables 3 and 4. Comparing the interval $[0, 55]$ with the two sub-intervals $[0, 34]$ and $[35, 55]$, we see that the maximum number of visits, 19, can be achieved after splitting but with an extra travel distance of about 46 km. As for execution time, splitting could save about 2.5 hours of calculation for the interval of $[0, 55]$. In total, time splitting for $Jan.12$ dataset which includes great number of nodes and time slots, reduced the execution time from 4.7 hours to 13 minutes with a small change in the maximum α value. Therefore, good splitting enormously reduces the execution time without much violating the maximum visits in general. Increasing the travel distance is the price we pay for accelerating the execution time.

In practice, there is a limitation in fuel consumption and battery usage of the service machine. Thus, we need to set a limit on the total distance travelled. For this reason we can simply add a preferred upper bound to the objective. In this case study, most vessels move in two lines in opposite directions (from east to west

Table 3: Maximum possible α and their travel distances for 6 sub-intervals related to data of $Jan.12$, 2018

Interval	No. of Targets	Max. No. of Visits (α)	Total Distance (km)	Exe. Time (min:sec)
0-34	25	12	94.964	4:23
35-55	18	7	54.788	0:32
56-73	14	7	28.233	0:15
74-107	17	11	84.515	4:32
108-155	13	9	52.050	3:00
156-192	9	8	92.358	0:17
Entire day	60	54	406.908	12:59

Table 4: Maximum possible α and their travel distances for 3 sub-intervals related to data of $Jan.12$, 2018

Interval	No. of Targets	Max. No. of Visits (α)	Total Distance (km)	Exe. Time (hour:min:sec)
0-55	31	19	102.841	2:37:16
56-107	26	18	112.459	1:25:56
108-192	20	19	165.121	0:39:59
Entire day	60	56	380.421	4:43:11

and vice versa). Based on this information, the third set of experiments is focused on finding gaps between travel distances when number of visits increases. Experiments are done on data of 5^{th} of July 2018 which has small number of nodes moving during the time interval so that the Pareto optima can be calculated for the whole interval T. Results are shown in Table 5 for the data of $Jul.5$. Gaps in distances occur mostly because of changing the lines. The two lines are approximately 10 km apart from each other. Therefore, the preference to increase the number of visits up to the next gap should be dominant over the extra distance we need to add by changing the current line. The maximum number of visits for the interval T is 23 out of 29 with three times changes of line. Figure 4 illustrates the optimal route before each gap occurs. The rows in Table 5 are highlighted right before the need to change the current line, therefore they specify when a gap occurs in travel distance. As can be seen in the results, 20 km of travel distance is saved if we visit two nodes less than maximum possible α by avoiding one extra changing of current line.

Table 5: Pareto optima for data of $Jul.5$ 2018 with total number of 29 nodes and 192 stages.

No. of Visits (α)	Total Distance (km)	Exe. Time (min:sec)
>23	Not Possible	4:28
23	138.58	3:34
22	131.75	2:46
21	117.71	2:48
20	108.50	2:11
19	101.29	2:18
18	91.52	1:30
17	82.33	0:45
16	75.61	0:47
15	69.42	0:40
14	56.79	0:38

5 Conclusions

In this paper, we constructed a mathematical model for bi-objective problem of finding an efficient route, maximizing the number of nodes visited along a time-dependent route and minimizing its total distance in a dynamic network. As an output, a set of Pareto optimal solutions can be produced. The resulted integer linear programming model can be computationally tackled with a help of some splitting heuristics proposed. Initial experiments on real historical data sets have provided preliminary evidence that the method can be efficiently used in practice for some problems of maritime routing management. Further research will be related to direct testing of the model for some real situations in applied maritime logistics analytics and justifying its application as a part of higher level hierarchical systems targeting autonomous navigation.

References

[1] Garey M., Johnson D. (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA.

[2] Papadimitriou C., Steiglitz K. (1982), Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Inc., NJ, USA.

[3] Dantzig G. (1963), Linear Programming and Extensions, Princeton University Press, Princeton, NJ, USA.

[4] Applegate D., Bixby R., Chvátal V., Cook W. (2007), The Traveling Salesman Problem: A Computational Study, Princeton University Press, Princeton, NJ, USA.

[5] Rego C., Gamboa D., Glover F., Osterman C. (2011), Traveling salesman problem heuristics: Leading methods, implementations and latest advances, *European Journal of Operational Research*, 211, 427-441.

[6] Androutsopoulos K., 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.

[7] Groba C., Sartal A., 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.

[8] Marlow D., Kilby P., Mercer G. (2007), The travelling salesman problem in maritime surveillance-techniques, algorithms and analysis, *Proceedings of the International Congress on Modelling and Simulation*, 684-690.

[9] Branke J., Deb K., Miettinen K., Slowinski, R. (2008), Multiobjective Optimization: Interactive and Evolutionary Approaches, Springer-Verlag, Berlin, Heidelberg, Germany.

[10] 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.

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

[12] Finnish Transport Agency (Liikennevirasto). https://www.liikennevirasto.fi, Accessed on 24/6/2018

[13] Miettinen K. Nonlinear multiobjective optimization (1999), Kluwer Academic Publishers, Boston.

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Computer Science
- Computer Engineering

ISBN 978-952-12-3738-6
ISSN 1239-1891