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 directed 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 points which gives the shortest travel distance, with maximum number of visits. A local search heuristic is proposed based on bi-objective integer programming model, for time splitting and search scope. 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 service management.

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 the 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 addition to combinatorial complexity, the presence of more than one optimization goals, needed to be achieved on the way to accurate and adequate modeling, may influence problem complexity. It is well-known that under multiobjective 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. Thus, many problems arising in optimization, applied mathematics and operations research should be ultimately considered under multicriteria framework due to existing of several conflicting goals or interests [9].

2 Problem formulation

Assume a variant of TSP problem which is defined in a dynamic network of n vertices, whose locations change during the time. Vertices can appear in different times, move along a random route and disappear after a while. The question is how to travel among moving vertices, such that we visit as many points as we can, with the shortest possible distance. Such optimization problem raised from real logistic services and supply chain planning. Furthermore, it can be used in many other artificial intelligence applications in similar circumstances. The proposed model of this research, is an extension of Picard and Queyranne [10] formulation, whose associated polytope, is studied by Abeledo et al. [11].

The problem can be formulated as follows. Let $\mathcal{N}^t = \{v_i^t | i \in \{1, ..., n\}, t \in T\}$ be the set of n nodes whose location depends on time $t \in T$, where T is the total travel time interval considered, and $\mathcal{A}^{tl}$ is the set of directed edges connecting nodes of a pair of time frames 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 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 window, that passes through as many nodes as possible with a constant speed, while every visit needs a processing time p_i before leaving and visiting the next node.

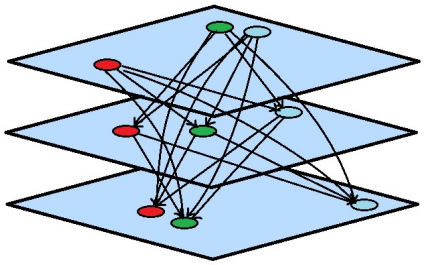

Figure 1: Representation of graph $\mathcal{G}$ with $n = 3$ vertices and three time slots

In order to formulate the above problem as a combinatorial model, 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 we have a source node v_0 which is not moving during the time. A Hamiltonian circuit, 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 vertex v_j^l is visited immediately after vertex v_i^k; and 0 otherwise:

$$\sum_{l=1}^{m-1}\sum_{j=1}^{n} x_{0j}^{0l} = 1 \tag{1}$$

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

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

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

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

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

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

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

Initially, the flow starts at point v_0. If the decision is made to move to j^{th} point at time slot l, then $x_{0j}^{0l} = 1$. So, we have the first constraint 1. If we are at node j in time slot l, we should exit the node at some later stages. So, initialized by the previous constraint, a unit of flow propagates throughout two stages k and l according to the balance constraints 2 (flow conservation). The flow turns back to v_0 at some time slot k using constraint 3. The inequalities 4 and 5 prohibit multiple moves at each time slot. Constraints 6 and 7 tell us that every point j is visited at most once. Therefore, we start from the stationary point v_0, visit a sequence of points along the route $\{(v_i^k, v_j^l) | x_{ij}^{kl} = 1\}$ and turn back to v_0. The primary objective is to minimize the travel distance, using the following objective function:

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

where d_{ij}^{kl} is the edge weight between vertices v_i^k and v_j^l in $\mathcal{G}$.

The secondary objective (maximizing the number of visits) is used as constraint to generate the Pareto frontier. The parameter α is defined to indicate the number of points that should be visited. Thus, the other objective can be defined as the

following inequality, and be added to the above constraints:

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

In order to create a time dependent route, we need to force visits occurring within the given time window. In other words, each start time s_k should satisfy the following time interval:

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

where s_0 and w are the start time and length of each time slot, respectively. In addition, enough time should be reserved for processing and travelling to the next point 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 mixed integer linear programming model, will return the shortest distance for visiting at least α moving points, as well as start times of each visit in a given time interval T. Despite the introduced model returns global optimal solution, and can be solved by branch-and-bound and cutting plane methods integrated into standard MIP solvers, but the large number of binary variables and constraints make it inefficient when using on large-size real datasets. In the next section, some local search heuristics on time and locations are applied which returns efficient solution within an acceptable execution time.

3 Local search heuristics

When there are a lot of points simultaneously moving in close distance to each other, the number of possible links among vertices and thus the number of variables and constraints grow exponentially. However, many of those points which are too far to reach, or have low chance of being visited can be omitted without violating the optimal solution. Based on this idea, we investigate the effect of splitting the time interval and limiting the search scope which are explained in the following.

In order to split the total time interval T, we need to find good 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 points in each time slot k as follows:

$$D(k) = \frac{1}{r^2(k)} \sum_{\substack{i,j=0 \\ (i\neq j)}}^{n} d_{ij}^{kk} \tag{13}$$

Table 1: Maximum possible α and their travel distances for 3 consecutive splits related to data of Jan.21, 2018

Interval	No. of Targets	Max. No. of Visits (α)	Total Distance (km)	Exe. Time (min)
0-48	17	10	150.24	2
49-103	24	16	56.46	2
104-192	16	16	61.62	1
Entire day	42	42	268.32	5

where $r(k)$ is the total number of points present during the k^{th} time slot. If the value of $D(k)$ is small in a time slot, it means that most of the points are located near each other, and thus there is a high chance of achieving a bigger α which increases the computational complexity. On the other hand, large value of $D(k)$ implies that points are distant from each other, and can possibly be omitted by limiting the search scope. As a result, good candidates for split points are those time slots with minimum values of $D(k)$ and maximum values of $r(k)$.

When two points are present simultaneously in one time split, but too far from each other, it would (not be impossible but) be less probable for that link to be included in the optimal path, specially if there are not many points present or coming after, in the side of the distant area. Therefore, excluding the corresponding links will not violate the optimal solution. Based on this, we can 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.

4 Computational performance

In order to validate the performance of the exact method and comparing the effect of local search on the optimal solution, we implemented 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 two sets of data belong to 21^{th} and 12^{th} of January, 2018, with 192 time slots (every 5 minutes during 16 hours) and total number of 42 and 60 vessels passing through the working area, respectively. For the Jan.12 dataset, three splits and for Jan.21, six splits are created. The diagram of $D(k)$ at a scale of $1000/1$ and $r(k)$ for Jan.12 dataset are illustrated in Figure 2. Vertical lines show six split points of the interval T. The processing time $p = 3$ minutes, and constant speed of 46 km per hour is used in the experiments.

As the input information including time window, stationary point v_0, location/time information of target points (of those which are missed in the previous round but still present, or those which arrive later in the interval) and the value of α differ in each time split, we need to update input data before each solving of the model. The output of each solution will be the optimal path (a sequence of labels

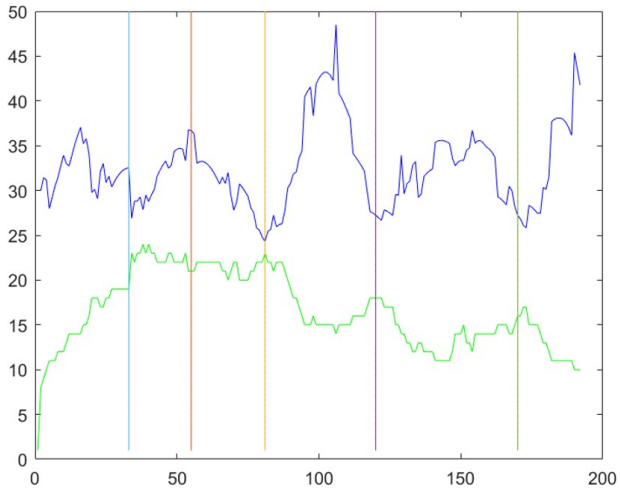

Figure 2: The horizontal line shows the time slots. Vertical lines show six split points of the interval T. The blue diagram is the value of $D(k)$ at a scale of $1000/1$. The green diagram is the number of ships present in each time slot, for the data of Jan.12, 2018

with their time/location information), predicted value of total distance needs to be travelled, start times of each process and list of labels (points) which cannot be visited. Table 1 shows total distance travelled for maximum possible α for consecutive time splits related to data of Jan.21. The optimal route for the first interval includes the distance from the harbor to the working area. The stationary point is supposed to be the location/time point of the last visit in the previous round. The execution time relates to calculation of pairwise distances to generate the network and solving the model for a given α.

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. Table 2 shows Pareto optima of each split corresponding to the maximum number of visits, and the difference in travel distances if we decide to miss one visit in order to get a shorter distance, for the data of Jan.12. The search scope is set to $\tau = 40$ km. As can be seen in the results, 166 km of travel distance is saved if we visit one point less than maximum possible α.

In the second set of experiments, we investigate the effect of splitting. Table 3 shows the maximum number of visits and their corresponding travel distances, when less number of splits are generated for the same data of Table 2. We decided to merge each two consecutive splits to see the difference in maximum number of visits as well as the optimal distance travelled between the merged interval and two splits. According to the result, good splitting enormously reduces the execution time without violating the maximum number of visits in total. However, increasing the travel distance up to 38.4 km is the price we pay for accelerating the execution time (from one hour to about three minutes).

The third set of experiments, investigated the effect of removing weighted links according to a predetermined threshold, on the number of visits and the computa-

Table 2: Pareto optima for max and max-1 of α values for the specified time splits on data of Jan.12, 2018

Interval	No. of Targets	Max. No. of Visits (α)	Total Distance (km)	Exe. Time (min:sec)
0-33	21	10	66.6	0:28
34-55	20	8	31.9	0:09
56-81	22	10	62.6	0:17
82-120	17	12	89.9	1:54
121-170	15	9	89.9	0:25
171-192	6	5	55.5	0:01
Entire day	60	54	396.4	3:14
0-33	21	9	52.1	0:27
34-55	20	7	26.7	0:10
56-81	23	9	29.0	0:17
82-120	20	11	69.3	2:26
121-170	19	8	29.7	0:43
171-192	10	4	23.2	0:04
Entire day	60	48	230	4:07

tional burden. Computational results for $\tau = 10$ and $\tau = 15$ in Table 4, reveal almost no change in execution time and minor difference between the total distances, but considerable loss in number of visits when the search scope is up to 10 km. One the other hand, comparing the results of Table 2 with $\tau = 40$ and Table 4 with $\tau = 15$, we see that limiting search scope up to 15 km, can result in minor loss in number of visits and in return, can speed up the calculation in some dense splits (e.g. up to 4 times in the fourth split). Therefore, choosing the optimal threshold can increase the efficiency.

Table 3: Maximum possible α and their travel distances for 3 splits 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	18	90.7	0:21:16
56-120	29	20	120.1	0:27:38
121-192	18	16	147.2	0:19:10
Entire day	60	54	358	1:08:04

Table 4: The effect of limiting search scope

Search Scope	Interval	No. of Targets	No. of Visits	Total Distance	Exe. Time (min:sec)
Up to 10 km	0-33	21	10	66.7	0:20
	34-55	20	8	31.9	0:06
	56-81	22	9	43.4	0:09
	82-120	18	9	37.0	0:29
	121-170	18	10	50.3	0:33
	171-192	6	0	0	0:01
	Total	60	46	229.3	1:38
Up to 15 km	0-33	21	10	68.6	0:23
	34-55	20	8	31.9	0:07
	56-81	22	10	62.1	0:33
	82-120	17	11	80.6	0:28
	121-170	16	8	36.8	0:21
	171-192	8	5	32.0	0:01
	Total	60	52	312.0	1:53

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

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