The Railway Traveling Salesman Problem ⋆

Georgia Hadjicharalambous[1] Petrica Pop[1] Evangelia Pyrga[1,2]
George Tsaggouris[1,2] Christos Zaroliagis[1,2]

[1] Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece
[2] Dept of Computer Eng. & Informatics, University of Patras, 26500 Patras, Greece
{hadjicha,ppop,pirga,tsaggour,zaro}@ceid.upatras.gr

Abstract. We consider the Railway Traveling Salesman Problem (RTSP) in which a salesman using the railway network wishes to visit a certain number of cities to carry out his/her business, starting and ending at the same city, and having as goal to minimize the overall time of the journey. RTSP is an $\mathcal{NP}$-hard problem. Although it is related to the Generalized Asymmetric Traveling Salesman Problem, in this paper we follow a direct approach and present a modelling of RTSP as an integer linear program based on the directed graph resulted from the timetable information. Since this graph can be very large, we also show how to reduce its size without sacrificing correctness. Finally, we conduct an experimental study with real-world and synthetic data that demonstrates the superiority of the size reduction approach.

1 Introduction

We consider a problem of central interest in railway optimization. We assume that we are given a set of stations, a timetable regarding trains connecting these stations, an initial station, a subset $\mathcal{B}$ of the stations, and a starting time. A salesman wants to travel from the initial station, starting not earlier than the designated time, to every station in $\mathcal{B}$ and finally return back to the initial station, subject to the constraint that s/he spends the necessary amount of time in each station of $\mathcal{B}$ to carry out his/her business. The goal is to find a set of train connections such that the overall time of the journey is minimized. We call this the *Railway Traveling Salesman Problem* (RTSP).

RTSP is related to the Generalized Asymmetric Traveling Salesman Problem (GATSP) [2–4]. In that problem, a weighted complete directed graph is given whose nodes are partitioned into clusters $V_1, V_2, \ldots, V_k$. The goal is to find a minimum-weighted tour containing at least one node from each cluster. By virtue of TSP, GATSP is an $\mathcal{NP}$-hard problem. It can be easily seen that TSP is polynomially time reducible to RTSP as well: for each pair of cities, for which

⋆ This work was partially supported by the Human Potential Programme of EU under contract no. HPRN-CT-1999-00104 (AMORE), by the IST Programme (6th FP) of EC under contract No. IST-2002-001907 (integrated project DELIS), and by the Action PYTHAGORAS funded by the European Social Fund (Operational Program for Educational and Vocational Training II) and the Greek Ministry of Education.

there exists a connection, consider a train leaving from the first one to the second with travel time equal to the cost of the corresponding connection in TSP. This reduction sets RTSP in the class of $\mathcal{NP}$-hard problems.

Consider the so-called time-expanded digraph G constructed from the timetable information [5]. In that graph, there is a node for every time event (departure or arrival) at a station, and edges between nodes represent either elementary connections between the two events (i.e., served by a train that does not stop in-between), or waiting within a station. The weight of an edge is the time difference between the time events associated with its endpoints. Now, roughly speaking, and considering each set of nodes belonging to a specific station as a node cluster, RTSP reduces in finding a minimum-weight tour that starts at a specific node of a specific cluster, visits at least one node of each cluster in $\mathcal{B}$, and ends at a node of the initial cluster.

Despite their superficial similarity, RTSP differs from GATSP in at least three respects:

(i) GATSP is typically solved by transforming it to an instance of the Asymmetric TSP. The transformation is done by modifying the weights such that inter-cluster edges are "penalized" by adding a large weight to them and consequently the tour has to visit all the nodes within a cluster before moving to the next one. It is not clear whether such an approach can be directly applied to RTSP, where we have to take into account that the salesman must spend a minimum amount of time at each station (city).

(ii) RTSP starts from a specific node within a specific cluster, while GATSP starts from any node within any cluster.

(iii) GATSP requires to visit all clusters, while RTSP asks for visiting only a subset of them.

Consequently, we feel that a different approach is worth pursuing for the solution of RTSP; that is, an approach that does not follow the classical pattern of reducing RTSP into an instance of GATSP or to an instance of the Asymmetric TSP.

In this paper, we follow such a direct approach and present a modelling of RTSP as an integer linear program based on the time-expanded graph. Since this graph can be very large, we also show how to reduce its size without sacrificing correctness. This turns out to be rather beneficial, especially in the case where the number of stations $|\mathcal{B}|$ the salesman wants to visit is much smaller than the total number of stations. Finally, we conduct an experimental study with real-world and synthetic data that demonstrates the superiority of the size reduction approach.

2 Preliminaries

In this section, we describe the input of an RTSP instance and we provide some definitions. In the following we assume timetable information in a railway system, but the modelling and the solution approaches can be applied to any other public transportation system provided that it has the same characteristics.

A *timetable* consists of data concerning: *stations* (or bus stops, ports, etc), *trains* (or busses, ferries, etc), connecting stations, *departure* and *arrival times* of trains at stations. More formally, we are given a set of *trains* $\mathcal{Z}$, a set of stations $\mathcal{S}$, and a set of *elementary connections* $\mathcal{C}$ whose elements are 5-tuples of the form $(z, \sigma_1, \sigma_2, t_{\rm d}, t_{\rm a})$. Such a tuple (elementary connection) is interpreted as train z leaves station σ_1 at time $t_{\rm d}$, and the *immediately next* stop of train z is station σ_2 at time $t_{\rm a}$. The *departure* and *arrival times* $t_{\rm d}$ and $t_{\rm a}$ are integers in the interval $T_{\rm day} = [0, 1439]$ representing time in minutes after midnight.

Given two time values t and t', $t \leq t'$, their *cycle-difference*(t, t') is the smallest nonnegative integer l such that $l \equiv t' - t \pmod{1440}$.

We are also given a starting station $\sigma_{\rm o} \in \mathcal{S}$, a time value $t_{\rm o} \in T_{\rm day}$ denoting the earliest possible departure time from $\sigma_{\rm o}$, and a set of stations $\mathcal{B} \subseteq \mathcal{S} - \sigma_{\rm o}$, which represents the set of the stations (cities) that the salesman should visit. A function $f_{\mathcal{B}} : \mathcal{B} \to T_{\rm day}$ is used to model the time that the salesman must spend at each city $b \in \mathcal{B}$, i.e., the salesman must stay in station $b \in \mathcal{B}$ at least $f_{\mathcal{B}}(b)$ minutes.

We naturally assume that the salesman does not travel continuously (i.e., through overnight connections) and that if s/he arrives too late in some station, then s/he has to rest and spend the night there. Moreover, the salesman's business for the next day may not require taking the first possible connection from that station. Consequently, we assume that the salesman never uses a train that leaves too late in the night or too early in the morning.

3 The Time-Expanded Graph

The formulation of RSTP is based on the so-called *time-expanded* digraph [5]. Such a graph $G = (V, E)$ is constructed using the provided timetable information as follows. There is a node for every time *event* (departure or arrival) at a station, and there are four types of edges. For every elementary connection $(z, \sigma_1, \sigma_2, t_{\rm d}, t_{\rm a})$ in the timetable, there is a *train-edge* in the graph connecting a *departure node*, belonging to station σ_1 and associated with time $t_{\rm d}$, with an *arrival node*, belonging to station σ_2 and associated with time $t_{\rm a}$. In other words, the endpoints of the train-edges induce the set of nodes of the graph. For each station $\sigma \in \mathcal{S}$, all departure nodes belonging to σ are ordered according to their time values. Let $v_1, \ldots, v_k$ be the nodes of σ in that order. Then, there is a set of *stay-edges*, denoted by $Stay(\sigma)$, (v_i, v_{i+1}), $1 \leq i \leq k-1$, and (v_k, v_1) connecting the time events within a station and representing waiting within that station. Additionally, for each arrival node in a station there is an *arrival-edge* to the immediately next (w.r.t. their time values) departure node of the same station. The cost of an edge (u, v) is *cycle-difference*(t_u, t_v), where t_u and t_v are the time values associated with u and v, respectively.

To formulate RTSP, we introduce the following modifications to the time-expanded digraph.

First, we do not include any elementary connections that have departure times greater than the latest possible departure time, or smaller than the earliest.

Second, we explicitly model the fact that the salesman has to wait at least $f_{\mathcal{B}}(b)$ time in each station $b \in \mathcal{B}$ by introducing a set of *busy-edges*, denoted by $Busy(b)$. We introduce a busy-edge from each arrival node in a station $b \in \mathcal{B}$, to the first possible departure node of the same station that differs in the time value by at least $f_{\mathcal{B}}(b)$.

Third, to model the fact that the salesman starts his journey at some station σ_o and at time t_o, we introduce a *source node* s_o in station σ_o with time value t_o. Node s_o is connected to the first departure node d_o of σ_o that has a time value greater than or equal to t_o, using an edge (called *source edge*) with cost equal to *cycle-difference*(t_o, t_{d_o}). In addition, we introduce a *sink node* s_f in the same station and we connect each arrival node of σ_o with a zero-cost edge (called a *sink edge*) to s_f.

Figure 1 gives an example of two stations in the time-expanded graph that illustrates our construction.

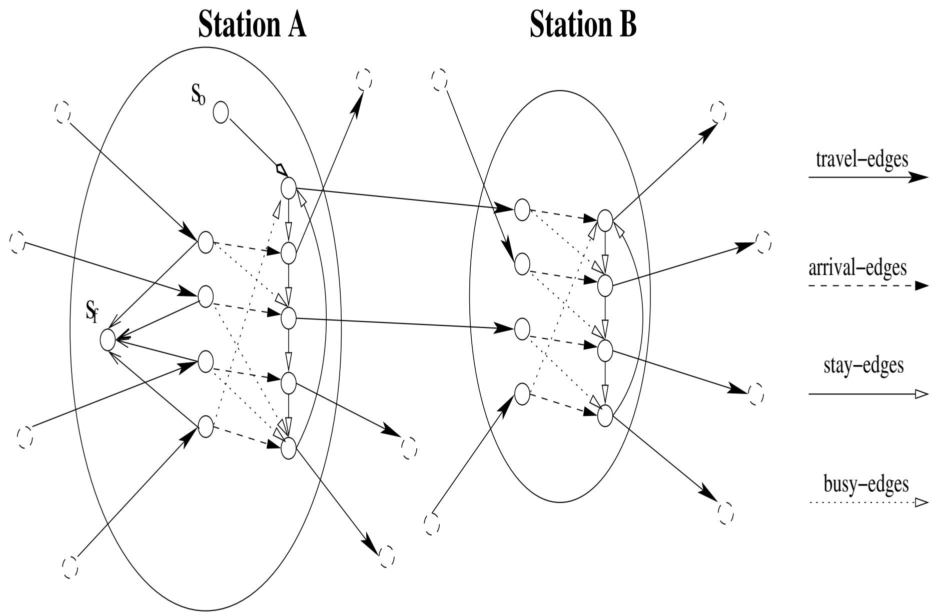

Fig. 1. An example of two stations in the time-expanded graph, where A is the starting station.

4 Integer Linear Programming Formulation

The objective of RTSP is to find a tour from node s_o to node s_f that passes through each station $b \in \mathcal{B}$, with minimum total cost.

In order to model the RTSP problem as an Integer Linear Program (ILP), we introduce for each edge (u,v) a variable $x_{(u,v)} \in \mathbb{N}_0$ that indicates the number of times the salesman uses the edge (u,v). If $c(u,v)$ denotes the cost of edge (u,v), then the ILP becomes:

$$\min \sum_{(u,v)\in E} c(u,v)x_{(u,v)} \tag{1}$$

$$s.t. \sum_{(v,u)\in E} x_{(v,u)} - \sum_{(u,v)\in E} x_{(u,v)} = 0, \qquad \forall u \in V - \{s_\mathrm{o}, s_\mathrm{f}\} \tag{2}$$

$$x_{(s_\mathrm{o},d_\mathrm{o})} = \sum_{(v,s_\mathrm{f})\in E} x_{(v,s_\mathrm{f})} = 1 \tag{3}$$

$$\sum_{e\in Busy(b)} x_e \geq 1, \qquad \forall b \in \mathcal{B} \tag{4}$$

$$x_e \in \mathbb{N}_0, \quad \forall e \in E \tag{5}$$

Constraints (2) & (3) are the flow conservation constraints that form a path from node s_o to node s_f, while constraints (4) ensure that the salesman spends the required time in each station $b \in \mathcal{B}$.

The problem with the formulation so far is that it permits feasible solutions that contain cycles, disjoint from the rest of the path (*subtours*). To deal with this problem, we have to add some more constraints that will force the solution not to use any subtours.

Suppose that the selected stations are numbered from 0 to $|\mathcal{B}|-1$. For each such station i, we introduce a new node f_i called the *sink node* of that station. Moreover, we create a copy $\overline{d}_k^i$ for the k-th departure node d_k^i of station i that has one or more incoming busy edges. We connect this new node with a zero cost edge to the original node. All busy edges now point to $\overline{d}_k^i$ instead of d_k^i, while all other edges remain unchanged. We also add an edge $(\overline{d}_k^i, f_i)$, for all k such that $\overline{d}_k^i \in V$. An example is given in Fig. 2.

We introduce now for each edge $e \in E$ a new set of variables y_e^i, $0 \leq i < |\mathcal{B}|$, and the following constraints:

$$\sum_{(v,u)\in E} y^i_{(v,u)} - \sum_{(u,v)\in E} y^i_{(u,v)} = 0, \qquad \forall u \in V - \{s_\mathrm{o}, f_i\} \tag{6}$$

$$y^i_{(s_\mathrm{o},d_\mathrm{o})} = \sum_{(v,f_i)\in E} y^i_{(v,f_i)} = 1 \tag{7}$$

$$y_e^i \leq x_e, \quad \forall e \in E - \{(u,f_i) \in E\} \tag{8}$$

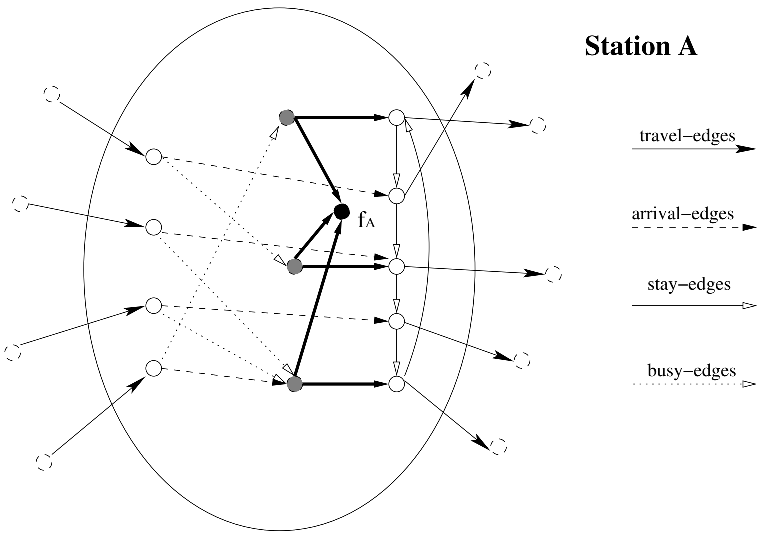

Fig. 2. Station A of Fig. 1 after introducing the new nodes and edges used for the subtour elimination constraints. For simplicity, nodes s_o and s_f are not shown. The grey nodes are the copies of the departure nodes that had incoming busy edges, while the black node is the sink node of A. The thick black edges are the new edges that have been introduced to the graph.

The above constraints form a multicommodity flow problem, introducing a commodity for each selected station and asking for one unit of flow of each commodity $i \in [0, |B|-1]$ to be routed from the source node s_o to the corresponding selected station's sink node f_i. An additional condition imposed on the y-variables is that the flow y_e^i of commodity i on any edge $e \in E$ cannot have a value greater than x_e. This means that if the x-flow is zero on some edge e, then the y-flows will all be zero on that edge, too. Constraints (6) – (8) force the x variables to be assigned appropriately so that for each selected station there exists a path from s_o to some departure node in that station, using only edges for which the corresponding x variables are non-zero. Since now the only way to reach f_i is through the busy edges of station i, the flow modelled by the x variables is forced to use some busy edge for each selected station, which makes constraints (4) redudant.

4.1 Path Retrieval

The reason that the variables $x_{(u,v)}$, $(u,v) \in E$, are not restricted to be 0-1 variables is the fact that the salesman is allowed to pass through the same station σ more than one times, regardless of whether σ belongs to $\mathcal{S}$, or to $\mathcal{B}$, or it is the starting station σ_o.

Knowing the values of the variables $x_{(u,v)}$, we can easily retrieve the path, using the Flow Decomposition Theorem (see e.g., [1, Chap. 3]). Let n (resp. m) be the number of nodes (resp. edges) of the time-expanded graph G. We decompose the flows into at most m simple cycles and one simple path (from s_o to s_f) in time $O(nm)$. We can then construct the minimum cost tour as a linked list of edges as follows. We initially set the tour equal to the path, and then we iteratively expand it by merging it with a cycle that has a common edge with it. This can be done in time linear to the length of the final tour which is at most $O(nm)$.

4.2 Size Reduction through Shortest Paths

The time-expanded graph can be rather large. In this section, we present a method that reduces the size of the graph, which in turn can be beneficial in the case where the salesman wishes to visit a relatively small number of stations compared to the total number of stations. We reduce the problem size by transforming the time-expanded graph G to a new graph G_{sh}, called the *reduced time-expanded graph*. Graph G_{sh} can be constructed by precomputing shortest paths among the stations that belong to $\mathcal{B}$, as follows.

Let again σ_o denote the start station. For each station σ in $\mathcal{B}\bigcup\{\sigma_o\}$ we introduce a sink-node s_σ and we connect each arrival-node of σ with a zero cost edge to s_σ. Then, for each departure node d of $\sigma \in \mathcal{B}\bigcup\{\sigma_o\}$, we compute a shortest path to the sink-nodes of every other station in $\mathcal{B}\bigcup\{\sigma_o\}$. If such a shortest path does not pass through some other station in $\mathcal{B}\bigcup\{\sigma_o\}$, then we insert a *shortest path edge* from d to the last arrival node of that path. The cost of this edge is equal to the cost of the corresponding shortest path.

We can now transform G in the following way. We first remove the sink-nodes (and their incoming edges) that were previously introduced. Next we delete from G all the nodes and edges that belong to stations that are not in $\mathcal{B}\bigcup\{\sigma_o\}$. The remaining arrival nodes (that belong to some station $\sigma \in \mathcal{B}\bigcup\{\sigma_o\}$) that were not used by any of the shortest paths that were previously computed are also deleted, as well as the remaining travel-edges. This procedure results in the reduced time-expanded graph G_{sh}.

5 Implementation Details

In order to speed-up the computations (trying to reduce the number of variables used by the integer programs) when we do not use the shortest path reduction, we try to eliminate the unnecessary nodes of the graph. To be more precise, we remove all arrival nodes of all stations $\sigma \notin \mathcal{B}$. These nodes have a single incoming edge (the one from the corresponding departure node), and a single out-going edge (to some departure node of the same station), while there are no busy edges starting from them. Therefore, we can eliminate these nodes (as well as their adjacent edges), and introduce a new virtual edge connecting the departure node that was the source of the incoming edge to the departure node that was the target of the outgoing edge, with cost the sum of the costs of the two eliminated edges. In this way, we can reduce the number of edges and nodes in the graph, and consequently the number of variables and constraints in the integer program.

6 Experiments

6.1 Data Sets

The construction of the graphs is based both on synthetic as well as on real-world data. For the synthetic case, we have considered grid graphs (with nodes representing stations). Each node has connections (in both directions) with all of its neighbouring nodes, i.e., the stations that are located immediately next to it in its row or column in the grid.

The connections among the stations were placed at random among neighbouring stations, such that there is at least one connection in every pair of neighbouring stations (for both directions) and the average number of elementary connections for each station is 10. The time-differences between the departure and the arrival time of each elementary connection are independent uniform random variables, chosen in the interval $[20, 60]$ (representing minutes), while the departure times are random variables in the time interval between the earliest and the latest possible departure time. We have created graphs whose number of stations varies from 20 to 80.

Data Set	Number of Stations $\lvert\mathcal{S}\rvert$	Number of Nodes	Number of Edges	Number of Connections	Conn/ Stations
`nd_loc`	23	778	1556	389	16.91
`nd_ic`	21	684	1368	342	16.29
synthetic 1	20	400	800	200	10
synthetic 2	30	600	1200	300	10
synthetic 3	40	800	1600	400	10
synthetic 4	50	1000	2000	500	10
synthetic 5	60	1200	2400	600	10
synthetic 6	70	1400	2800	700	10
synthetic 7	80	1600	3200	800	10

Table 1. Graph parameters for the original time-expanded graph in each data set. Note that the number of connections is equal to half the number of nodes, since two nodes form one connection, and the number of edges is twice the number of nodes. The number of edges does not include the incoming edges of s_{f} and the outgoing edge of s_{o} or any other artificial nodes or edges.

The real-world data represent parts of the railroad network of the Netherlands. The first data set, called `nd_ic`, contains the Intercity train connections among the larger cities in the Netherlands, stopping only at the main train stations, and thus are considered faster than normal trains. These trains operate at least every half an hour, while the number of stations is equal to 21. The second real-world data set, `nd_loc`, contains the schedules of the trains that connect the cities in only one region, including some main stations, while trains stop at each intermediate station between two main ones. The total number of stations in this case is 23.

The characteristics of all the graphs that were used, for both real and synthetic data, are shown in Table 1.

6.2 Description of Experiments

For each data set, several problem instances were created, varying the number $|\mathcal{B}|$ of the *selected stations*, i.e., the set of stations that the salesman must visit. For both graphs based on real and synthetic data, we have used two values for $|\mathcal{B}|$, namely 5 and 10. Note that $|\mathcal{B}|$ does not contain the starting station. Because of that, when $|\mathcal{B}|$ is set equal to the total number of stations, the actual value that will be used is $|\mathcal{B}| - 1$, since one station has to be the starting one.

Real graphs

Data Set	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{B}\rvert = 5$ Nodes	$\lvert\mathcal{B}\rvert = 5$ Edges	$\lvert\mathcal{B}\rvert = 10$ Nodes	$\lvert\mathcal{B}\rvert = 10$ Edges
`nd_loc`	23	209.4	415.2	343.9	698.1
`nd_ic`	21	181.6	385	303.6	708.7

Synthetic graphs

Data Set	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{B}\rvert = 5$ Nodes	$\lvert\mathcal{B}\rvert = 5$ Edges	$\lvert\mathcal{B}\rvert = 10$ Nodes	$\lvert\mathcal{B}\rvert = 10$ Edges
1	20	95.75	209.35	197.7	420.3
2	30	91.3	204.7	181.1	430.4
3	40	95.7	213.0	188.3	437.8
4	50	92.7	205.2	184.6	440.3
5	60	94.8	216.2	181.9	456.8
6	70	93.0	215.4	184.0	491.7
7	80	91.2	207.2	174.6	457.7

Table 2. Graph parameters for the reduced graphs for all data sets (average values).

For each combination of data set and a value of $\mathcal{B}$, we have selected the stations that belong to $\mathcal{B}$ randomly and independently from each other. The selection of stations has been repeated many times, and the mean values among all corresponding instances were computed. For each instance we have created, the corresponding integer program was given as input to `GLPSOL v4.6` (GNU Linear Programming Kit LP/MIP Solver, Version 4.6). The time needed by `GLPSOL` to find the optimum solution for each case has been measured.

We have tested both the original (with our modifications) version of the time-expended graph, as well as the reduced version based on precomputed shortest paths described in Section 4.2. The time for this precomputation has also been measured.

6.3 Results and Discussion

The results of the experiments performed are reported in Tables 2 and 3 (a graphical comparison is illustrated in Fig 3).

The values measured are the average values for 50 different sets of the selected stations. The standard deviation of the running times provided by GLPSOL for instances of the same parameters was large, showing a great dependence on the graph structure.

It can easily be seen that the value of $|\mathcal{B}|$ has a great impact on the running time, when the original graphs were used. The larger the $|\mathcal{B}|$, the larger the running time.

Running times for real data sets

Reduced Graphs

$\|\mathcal{B}\|$	5	10
`nd_loc`	319.0	9111.9
`nd_ic`	29.1	6942.6

Running times for the synthetic data sets

Original Graphs

$\|\mathcal{S}\|$	20	30	40
$\|\mathcal{B}\| = 5$	13.12	32.24	72.06
$\|\mathcal{B}\| = 10$	781.12	1287.00	16293.80

Reduced Graphs

$\|\mathcal{S}\|$	20	30	40	50	60	70	80
$\|\mathcal{B}\| = 5$	1.12	1.12	1.50	0.80	1.45	1.30	1.00
$\|\mathcal{B}\| = 10$	214.76	369.59	244.18	181.85	257.96	431.80	233.26

Shortest path computation times for synthetic data sets

$\|\mathcal{S}\|$	20	30	40	50	60	70	80
$\|\mathcal{B}\| = 5$	0.13	0.18	0.24	0.29	0.35	0.41	0.46
$\|\mathcal{B}\| = 10$	0.28	0.38	0.48	0.59	0.69	0.79	0.88

Table 3. Results for the synthetic data sets. Time is measured in seconds.

The size reduction approach based on precomputed shortest paths results in much smaller graphs than the original. Table 2 shows the characteristics of the reduced graphs that have resulted by the use of the size reduction technique.

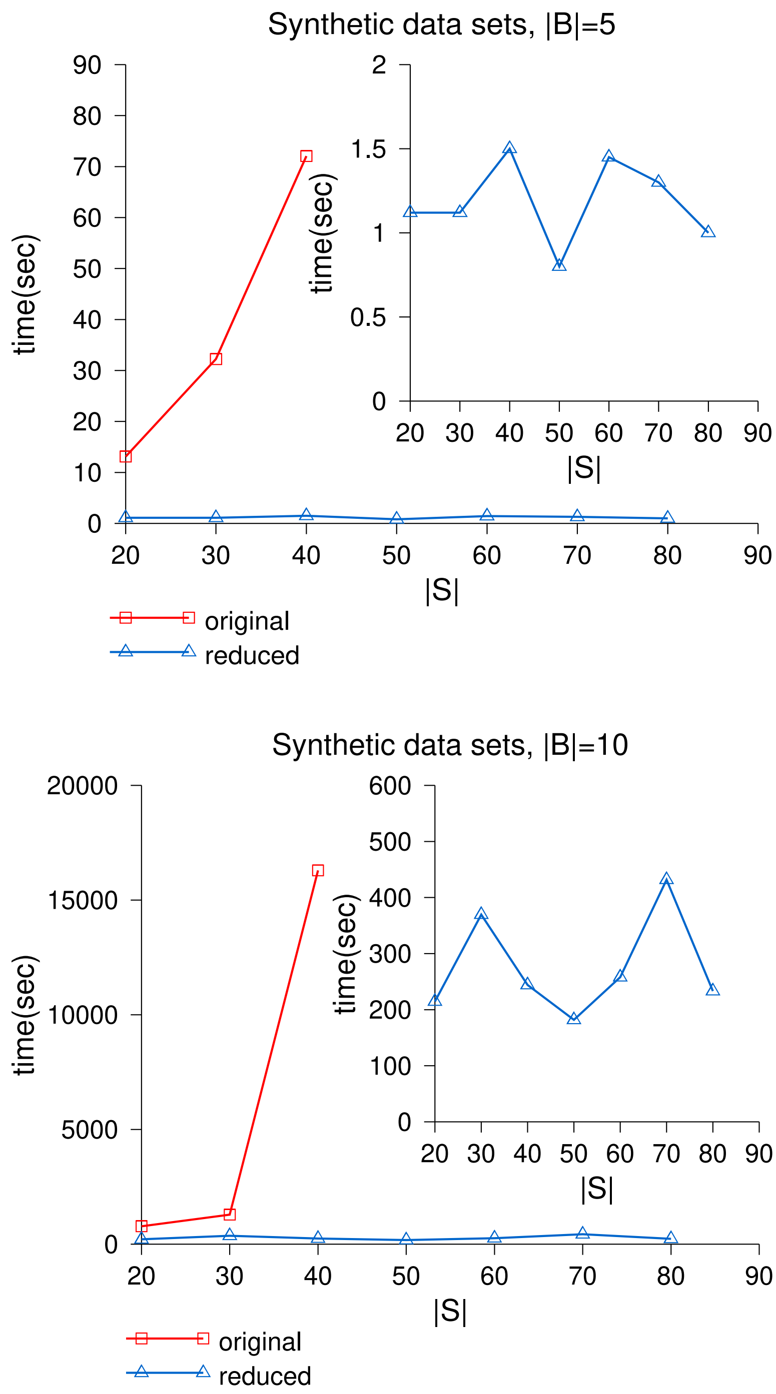

Fig. 3. Graphical comparison of running times for synthetic data sets.

Also, it is clear from Table 3 that the size reduction approach results in a great speedup w.r.t. the time achieved by considering the original time-expanded graph to find an optimal (or close to optimal) solution. Moreover, the smaller the value of $|\mathcal{B}|$, the larger the speedup.

For a fixed size of $\mathcal{B}$ the running time seems to grow rapidly with the increase of $\mathcal{S}$ when the shortest path technique is not being used. On the contrary, this is not the case when this technique is used. In this case, when we consider synthetic data, the running times are fluctuating. Nevertheless, it is quite clear that the time depends mainly on the size of $\mathcal{B}$, rather than the size of $\mathcal{S}$.

References

1. R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows*, Prentice-Hall, 1993.
2. M. Fischetti, J. Salazar-Gonzalez, and P. Toth, "The Generalized Traveling Salesman Problem and Orienteering Problem", in *The Traveling Salesman Problem and its Variations* (G. Gutin and A. Punnen, eds), Kluwer, 2002.
3. G. Gutin, "Traveling Salesman and Related Problems", in *Handbook of Graph Theory* (J. Gross and J. Yellen, eds), CRC Press, 2003.
4. C. Noon and J. Bean, "A Lagrangian Based Approach for the Asymmetric Generalized Traveling Salesman Problem", *Operations Research* 39 (1991), pp. 623-632.
5. F. Schulz, D. Wagner, and K .Weihe, "Dijkstra's Algorithm On-Line: An Empirical Case Study from Public Railroad Transport," *ACM Journal of Experimental Algorithmics*, 5:12 (2000).