Publié par :
Published by:
Publicación de la:

Faculté des sciences de l'administration
2325, rue de la Terrasse
Pavillon Palasis-Prince, Université Laval
Québec (Québec) Canada G1V 0A6
Tél. Ph. Tel. : (418) 656-3644
Télec. Fax : (418) 656-7047

Disponible sur Internet :
Available on Internet
Disponible por Internet :

http://www4.fsa.ulaval.ca/la-recherche/publications/documents-de-travail/

DOCUMENT DE TRAVAIL 2019-006

The Time-Dependent Shortest Path and Vehicle Routing Problem

Rabie JABALLAH
Marjolein VEENSTRA
Leandro C. COELHO
Jacques RENAUD

Document de travail également publié par le Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, sous le numéro CIRRELT-2019-12

Mars 2019

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2019
Bibliothèque et Archives Canada, 2019

ISBN 978-2-89524-485-1 (PDF)

The Time-Dependent Shortest Path and Vehicle Routing Problem[+]

Rabie Jaballah[1], **Marjolein Veenstra**[2], **Leandro C. Coelho**[1,2*], **Jacques Renaud**[1]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325, rue de la Terrasse, Université Laval, Québec, Canada, G1V 0A6

[2] University of Groningen, Faculty of Economics and Business, Department of Operations, Nettelbosje 2, 9747 AE Groningen,The Netherlands

**Corresponding author: leandro.coelho@cirrelt.ca*

ABSTRACT

In this paper we introduce the time-dependent shortest path and vehicle routing problem. In this problem, a set of homogeneous vehicles is used to visit a set of customer locations dispersed over a very large network, such that the travel times between any two customers must be computed as a time-dependent shortest path problem. The travel time of each arc is time-dependent and therefore the shortest path between two locations changes over time. The aim of the problem is to simultaneously determine the sequence in which the customer locations are visited and the arcs traveled on the paths between each pair of consecutively visited customers, such that the sum of the arrival times of the vehicles back at the depot is minimized. We are the first to formally define and solve this fully integrated problem, giving bounds to it. We then propose a dynamic time-dependent shortest path algorithm embedded within a simulated annealing metaheuristic to solve the problem. We test our formulations and algorithms on a set of real-life instances generated from a dataset of the road network in Québec City, Canada. Our results indicate that neglecting the real traffic can impose substantial delays for the visits, which would require more trucks and more mileage to perform the same deliveries. Moreover, the results show that one must consider the whole underlying road network in order to truly obtain the fastest paths between two points. Our work adds new research avenues to city logistics and congestion/emission studies.

Keywords: Time-dependent vehicle routing problem, time-dependent shortest path problem, green logistics, city logistics, heuristic, simulated annealing.

Acknowledgments: This project was partly funded by the Dutch Institute for Advanced Logistics (Dinalog) and by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2014-05764 and 2018-03712. This support is greatly acknowledged. We thank Khaled Belhassine for creating and providing the instances used in this research.

[+] Revised version of the 2017-012 (CIRRELT 2017-57)

1 Introduction

Distribution companies performing several deliveries per day across a city have to deal with congestion and delays when planning their routes. Therefore, they need to incorporate time-dependent travel times in their routing tools. Time-dependent travel times imply that the time it takes to traverse an arc depends on the departure time at the starting node of the arc. As a result, the shortest path between two delivery locations can change depending on the departure time at the origin. Consider the road network as presented in Figure 1, consisting of six different road intersections: A, B, C, D, E, and F. Suppose a vehicle consecutively visits delivery locations A and F in this multigraph [Ticha et al., 2019]. Then, there exist three paths from location A to location F, namely (A, B, D, F), (A, B, E, F) and (A, C, E, F). The travel time of the arcs between the intersections is time-dependent and therefore the shortest path between locations A and F can change between these three paths throughout the day. Hence, planning the vehicle routes includes not only assigning customers to vehicles and deciding upon the sequence in which the customer locations are visited, but also determining the arcs which are traveled between two consecutive delivery locations. Obviously, on large networks there are much more than three ways to travel between two customers, and also the number of customers to visit and their sequence can be very large. For a detailed analysis of time-dependent minimum cost paths we refer the reader to Heni et al. [2019].

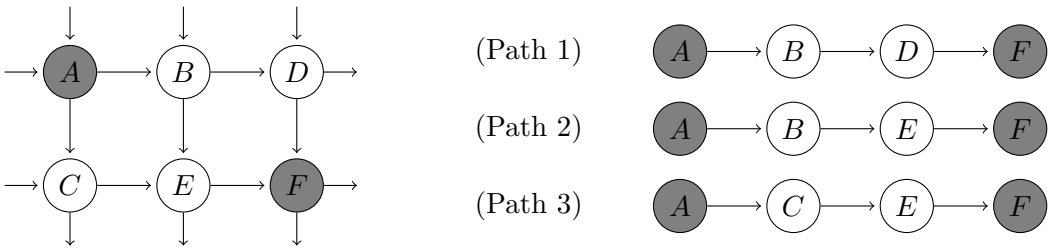

Figure 1: On the left, the graphical representation of a part of a road network with one-way streets and six intersections. On the right, the three possible paths from location A to location F.

In this paper, we introduce the time-dependent shortest path and vehicle routing problem (TDSPVRP). In the TDSPVRP, a homogeneous fleet of capacitated vehicles based at the depot is used to visit a set of customer locations. In this problem, we simultaneously determine the sequence in which the customer locations are visited by the vehicles and the arcs traveled on the paths between two consecutively visited customers. The travel time of each arc is time-dependent and therefore the shortest path between two locations, with respect to travel time, is dependent on the departure time at the first location. The goal of the problem is to find feasible routes that specify the sequence in which the customers are visited and the paths between those customers, such that each customer is *served* exactly once. Note that a customer node can be visited more than once because its node is on the path leading to another customer. The objective is to minimize the sum of the arrival times of the vehicles back at the depot. Hence, instead of minimizing driving costs, we focus on avoiding congestion and traffic resulting in running at more reasonable speeds, which in turn results in less fuel consumption and less emissions.

This objective is in line with recent research on routing in a city logistics context. For example, Bruglieri et al. [2019] proposes computing several different paths to solve a green vehicle routing problem. Moreover, Koç et al. [2016] consider depot location, fleet composition and routing in a city logistics context and define the routing cost in terms of fuel consumption and CO_2 emissions, and Bektaş and Laporte [2011], Franceschetti et al. [2013] and Koç et al. [2014] that each consider a specific pollution-routing problem. The pollution-routing problem is a variant of the vehicle routing problem with an objective that includes not only travel distance, but also accounts for the amount of greenhouse gas emissions, fuel, travel times and their costs. Also from a practical point of view this objective is of interest, since it aims at avoiding congestion for all vehicles. By considering a finite time horizon, we prevent that too much of the workload is assigned to only a small set of the vehicles.

The time-dependent vehicle routing problem (TDVRP) is a distribution problem with time-dependent travel times but considering only one path linking any two customers. Hence, solving the TDVRP does not require the time-dependent shortest path aspect to be present. Therefore, the travel times between two nodes, i.e., customer or depot nodes, are supposedly known and are input to the problem. As a result, the graphs for the TDVRP are very dense, i.e., a complete graph where all nodes need to be visited and the only arcs that are present are those connecting each pair of customers and each customer node with the depot node. This is in contrast with a TDSPVRP instance that requires the complete road network as input. As a consequence, TDSPVRP graphs are relatively sparse, with many nodes and relatively few of them requiring a visit, i.e., the customer nodes. Figure 2 illustrates two TDSPVRP instances, containing 10 and 50 customers. We observe that the number of customers in the instance has no impact on the size of the graph. Note that the TDSPVRP can be converted into a TDVRP by solving *all pairs* of shortest paths for *each* starting time. This procedure is, however, very time consuming and prohibitive. For this reason, we formally and explicitly define the TDSPVRP.

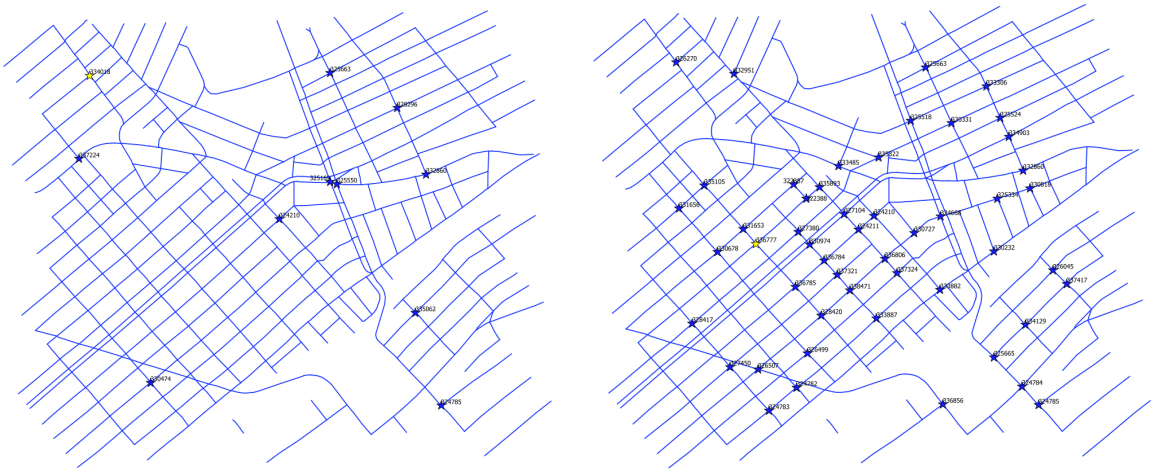

Figure 2: Instances with 10 customers (left) and 50 customers (right); the depot and customers locations are marked with stars.

The TDSPVRP is a combination of the time-dependent shortest path problem (TDSPP) and the TDVRP. The TDSPP is a shortest path problem with time-dependent travel times and

was introduced by Cooke and Halsey [1966]. Their algorithm was quickly improved by Dreyfus [1969]. Other algorithms for the TDSPP are developed by, e.g., Chabini [1998], Ding et al. [2008], Orda and Rom [1990] and Ziliaskopoulos and Mahmassani [1993]. The TDVRP was introduced by Malandraki and Daskin [1992] who proposed a mixed integer formulation for the problem. Many have formulated the problem exactly, e.g., Chen et al. [2006] and Soler et al. [2009] who have proposed a conversion of the TDVRP into a larger graph defining a simple capacitated vehicle routing problem. Several heuristics are proposed for the problem, e.g., Donati et al. [2008] who proposed a multi-ant colony system and Hashimoto et al. [2008] who developed an iterated local search heuristic.

Closely related to our research are the papers reviewed next. Kok et al. [2012] consider the combination of time-dependent shortest path problems and time-dependent vehicle routing problems. They use a restricted dynamic programming heuristic to solve four different combinations of problems and show significant improvements when considering time-dependent shortest paths in the TDVRP. Huang et al. [2017] consider the time-dependent vehicle routing problem with path flexibility (TDVRP-PF). In the TDVRP-PF, path selection is explicitly considered and integrated in the TDVRP. This means that each arc between two customer nodes has multiple corresponding paths in the underlying road network. The path selection decision is based on the departure time at the customer node and the level of congestion. For each pair of customers a small set of candidate paths are preprocessed, based on the knowledge of the network. The authors formulate the deterministic version of the TDVRP-PF as a mixed integer program and the stochastic version as a two-stage stochastic mixed integer program. Results show that path flexibility provides significant savings. In the formulations of Huang et al. [2017] only the candidate paths can be selected. The advantage of our approach is that all paths between two customers can be used and therefore a better solution might be found. Moreover, our approach is applicable to general networks, for which no usage information is known or available. Finally, embedding the decision on the arcs traveled between two consecutive delivery locations in the mixed integer program ensures that the shortest paths do not have to be preprocessed. Angelelli et al. [2018] also uses a subset of possible paths to generate paths for proactive route guidance avoiding congestion. Unlike us, Nasri et al. [2018] controls the speed along fixed paths, not trying to avoid congestion but aiming to minimize emissions, fuels and driver costs.

In this paper we develop an exact method, with the aim of providing and improving dual bounds for the problem. Given the size and difficulty of the problem, a dynamic time-dependent algorithm to determine the shortest path linking two nodes in the graph and a heuristic based on local search and simulated annealing are proposed providing much better primal bounds than our exact method. The goal of this paper is to provide the first set of benchmark instances for the TDSPVRP and to give bounds to the problem, opening a research avenue for others.

The remainder of this paper is as follows. In Section 2 we give the mathematical formulation for the TDSPVRP and we develop inequalities to strengthen this formulation and to improve the lower bounds. In Section 3 we describe the heuristic algorithm. Section 4 reports the results from the experiments performed on real-life instances from a dataset of the road network in Québec City, discusses the impact of the valid inequalities, the performance of the heuristic algorithm, and provides insights on time-dependent optimization. Conclusions are given in Section 5.

2 Problem statement and mathematical formulation

The TDSPVRP is defined on a directed time-dependent graph $\mathcal{G} = (\mathcal{N}, \mathcal{A}, \mathcal{H})$, where $\mathcal{N}$ is the set of nodes, $\mathcal{A}$ is the set of arcs and $\mathcal{H}$ is the set of time periods. Let $\mathcal{N} = \mathcal{N}' \cup \mathcal{N}_p$, where we refer to $\mathcal{N}'$ as the set of location nodes and $\mathcal{N}_p$ as the set of intermediate nodes. Let the set of location nodes $\mathcal{N}' = \mathcal{N}_c \cup \mathcal{N}_o \cup \mathcal{N}_d$, where $\mathcal{N}_c$ is the set of customer nodes, $\mathcal{N}_o$ is the set of origin depot nodes and $\mathcal{N}_d$ is the set of destination depot nodes. The origin and destination depot nodes are duplicated such that there is a unique origin and destination depot node for each vehicle from the homogeneous set $\mathcal{K} = \{1, \ldots, K\}$. For modeling purposes, we created for each vehicle an arc between its origin and destination depot node. We set the travel times of these K arcs equal to zero for each time period. Traveling along this arc represents not using this vehicle, but satisfying other constraints such as the ones imposing that every vehicle must leave the origin depot and arrive at the destination depot. Let $\mathcal{N}_p$ be the set of nodes that can be visited on a path between two location nodes, including nodes $i \in \mathcal{N}'$. The location nodes $\mathcal{N}'$ need to be visited exactly once, whereas the intermediate nodes do not necessarily have to be visited, but can be visited once or multiple times. Note that visiting an intermediate node $j \in \mathcal{N}_p$ that is a copy of a location node $i \in \mathcal{N}'$, corresponds to driving by this location without serving it. Let $\mathcal{H} = \{1, \ldots, H\}$ define H time periods, with each time period representing a time slot of length $\overline{T}$. We assume that the travel time of arc $(i, j) \in \mathcal{A}$ starting in time period h is known and is equal to t_{ijh}. The service time of a node $i \in \mathcal{N}'$ is given by s_i, where $s_i = 0 \; \forall i \in \mathcal{N}_o \cup \mathcal{N}_d$ and $s_i > 0 \; \forall i \in \mathcal{N}_c$. Waiting time is allowed at each node, which might be beneficial if the FIFO property does not hold, i.e., if it may occur that when traveling arc (i, j), postponing the departure time at node i results in an earlier arrival time at node j. The demand at a node is given by $d_i \; \forall i \in \mathcal{N}'$, where $d_i = 0 \; \forall i \in \mathcal{N}_o \cup \mathcal{N}_d$ and $d_i > 0 \; \forall i \in \mathcal{N}_c$. The capacity of each vehicle $k \in \mathcal{K}$ is equal to Q. The goal of the problem is to determine vehicle routes satisfying the demands of all customers, creating proper paths between two consecutive customer visits, such that the total of the arrival time of the vehicles back to the depot is minimized.

In the definition of our variables, we use the term *shortest path towards a node*. If $i \in \mathcal{N}'$ is the last location node visited before the visit to location node $j \in \mathcal{N}'$, then the nodes on the shortest path towards node j are nodes i, j, and all intermediate nodes visited between nodes i and j, and the arcs on the shortest path towards node j are all arcs that are traversed between nodes i and j. An example is given in Figure 3, which depicts a partial route $\mathcal{R} = (1, 2, 3, 4, 5, 6)$, where the location nodes are given by $\mathcal{N}' = \{1, 3, 6\}$ and the intermediate nodes are in the set $\mathcal{N}_p = \{2, 4, 5\}$. The nodes and arcs on the shortest path towards node 3 are given by $\{1, 2, 3\}$ and $\{(1, 2),\ (2, 3)\}$, respectively. The nodes and arcs on the shortest path towards node 6 are given by $\{3, 4, 5, 6\}$ and $\{(3, 4),\ (4, 5),\ (5, 6)\}$, respectively.

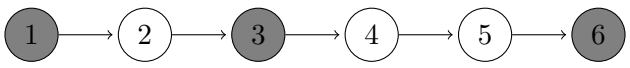

Figure 3: Partial route with $\mathcal{N}' = \{1, 3, 6\}$ and $\mathcal{N}_p = \{2, 4, 5\}$.

Our mathematical model is defined as follows. Let x_{ij} be a binary variable equal to one if and only if node $i \in \mathcal{N}'$ is the last location node visited before location node $j \in \mathcal{N}'$ and y_{ijl} be a binary variable equal to one if and only if arc $(i, j) \in \mathcal{A}$ is traversed on the shortest path towards node $l \in \mathcal{N}'$. Hence, in Figure 3, we have $x_{13} = x_{36} = 1$, $x_{ij} = 0$ otherwise, and $y_{123} = y_{233} =$

$y_{346} = y_{456} = y_{566} = 1$, $y_{ijl} = 0$ otherwise. Let $T_{il} \in \mathbb{Z}$ be the time at which the vehicle departs from node $i \in \mathcal{N}$ on the shortest path towards node $l \in \mathcal{N}'$, and $q_i \in \mathbb{Z}$ reflects the cumulated load carried by the vehicle departing from node $i \in \mathcal{N}'$. The binary variable δ_{plh} is equal to one if and only if a vehicle leaves node p on the shortest path towards node $l \in \mathcal{N}'$ in time period $h \in \mathcal{H}$. Let $\delta^2_{pml} \in \mathbb{Z},\ p \in \mathcal{N},\ m \in \mathcal{N}_p \cup l : (p,m) \in \mathcal{A},\ l \in \mathcal{N}'$ be the time at which a vehicle leaves node p on the shortest path towards node l if arc (p,m) is traversed on the shortest path towards node l, zero otherwise. Let $\delta^3_{pmlh} \in \mathbb{Z},\ p \in \mathcal{N},\ m \in \mathcal{N}_p \cup l : (p,m) \in \mathcal{A},\ l \in \mathcal{N}',\ h \in \mathcal{H}$ be the travel time of arc (p,m) in time period h if arc (p,m) is traversed on the shortest path towards node l in time period h, zero otherwise.

The model can then be defined as follows:

$$\min \quad \sum_{i \in \mathcal{N}_d} T_{ii} \tag{1}$$

s.t.
$$\sum_{i \in \mathcal{N}'} x_{ij} = 1 \qquad \forall j \in \mathcal{N}' \backslash \mathcal{N}_o \tag{2}$$

$$\sum_{j \in \mathcal{N}'} x_{ij} = 1 \qquad \forall i \in \mathcal{N}' \backslash \mathcal{N}_d \tag{3}$$

$$\sum_{j:(i,j) \in \mathcal{A}} y_{ijl} - \sum_{j:(j,i) \in \mathcal{A}} y_{jil} = \begin{cases} 0 & \text{if } i \notin \mathcal{N}' \\ -1 & \text{if } i = l \\ x_{il} & \text{if } i \in \mathcal{N}' \backslash l \end{cases} \qquad \forall i \in \mathcal{N},\ l \in \mathcal{N}' \backslash \mathcal{N}_o \tag{4}$$

$$x_{ij} - \sum_{p:(i,p) \in \mathcal{A}} y_{ipj} = 0 \qquad \forall i,\ j \in \mathcal{N}' \tag{5}$$

$$q_j = 0 \qquad \forall j \in \mathcal{N}_o \tag{6}$$

$$q_i - q_j + Q x_{ij} + (Q - d_i - d_j)\, x_{ji} \leq Q - d_i \qquad \forall i,\ j \in \mathcal{N}_c : i \neq j \tag{7}$$

$$q_j \leq Q \qquad \forall j \in \mathcal{N}_d \tag{8}$$

$$T_{ml} \geq T_{mm} - H\overline{T}\,(1 - x_{ml}) \qquad \forall l \in \mathcal{N}',\ m \in \mathcal{N}' \backslash \{\mathcal{N}_o \cup l\} \tag{9}$$

$$T_{ml} \geq \sum_{p:(p,m) \in \mathcal{A}} \left(\delta^2_{pml} + \sum_{h \in \mathcal{H}} \delta^3_{pmlh} \right) \qquad \forall l \in \mathcal{N}',\ m \in \mathcal{N}_p \tag{10}$$

$$T_{ml} \geq \sum_{p:(p,m) \in \mathcal{A}} \left(\delta^2_{pml} + \sum_{h \in \mathcal{H}} \delta^3_{pmlh} \right) + s_l \qquad \forall l \in \mathcal{N}',\ m = l \tag{11}$$

$$\sum_{h \in \mathcal{H}} \delta_{llh} = 1 \qquad \forall l \in \mathcal{N}' \tag{12}$$

$$\sum_{h \in \mathcal{H}} \delta_{ilh} = \sum_{j:(i,j) \in \mathcal{A}} y_{ijl} \qquad \forall i \in \mathcal{N},\ l \in \mathcal{N}' \backslash i \tag{13}$$

$$T_{pl} \geq (h-1)\,\overline{T}\,\delta_{plh} \qquad \forall p \in \mathcal{N},\ l \in \mathcal{N}',\ h \in \mathcal{H} \tag{14}$$

$$T_{pl} + 1 \leq H\overline{T}\,(1 - \delta_{plh}) + h\overline{T} \qquad \forall p \in \mathcal{N},\ l \in \mathcal{N}',\ h \in \mathcal{H} \tag{15}$$

$$\delta^2_{pml} \geq T_{pl} - (1 - y_{pml})\, H\overline{T} \qquad \forall p \in \mathcal{N},\ m \in \mathcal{N}_p \cup l : (p,m) \in \mathcal{A}, l \in \mathcal{N}' \tag{16}$$

$$\delta^3_{pmlh} \geq t_{pmh}\,(y_{pml} + \delta_{plh}) - t_{pmh} \qquad \forall p \in \mathcal{N},\ m \in \mathcal{N}_p \cup l : (p,m) \in \mathcal{A}, l \in \mathcal{N}',\ h \in \mathcal{H} \tag{17}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,\ j \in \mathcal{N}' \tag{18}$$

$$y_{ijl} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A},\ l \in \mathcal{N}' \tag{19}$$

$$T_{il} \in \mathbb{Z} \qquad \forall i \in \mathcal{N},\ l \in \mathcal{N}' \tag{20}$$

$$q_i \in \mathbb{Z} \qquad \forall i \in \mathcal{N}' \tag{21}$$

$$\delta_{plh} \in \{0,1\} \qquad \forall p \in \mathcal{N},\ l \in \mathcal{N}', h \in \mathcal{H} \tag{22}$$

$$\delta^2_{pml} \in \mathbb{Z} \qquad \forall p \in \mathcal{N},\ m \in \mathcal{N}_p \cup l : (p,m) \in \mathcal{A}, l \in \mathcal{N}' \tag{23}$$

$$\delta^3_{pmlh} \in \mathbb{Z} \qquad \forall p \in \mathcal{N},\ m \in \mathcal{N}_p \cup l : (p,m) \in \mathcal{A}, l \in \mathcal{N}',\ h \in \mathcal{H}. \tag{24}$$

The objective (1) is to minimize the sum of the arrival times of the vehicles at the depot. Constraints (2) and (3) are standard degree constraints. Constraints (4) and (5) ensure that $y_{ijl} = 1$ if arc (i,j) is on the shortest path towards node l, and 0 otherwise. Constraints (6)–(8) are the load constraints. The explanation is based on customers that require items to be picked, but the constraints can also be applied when dealing with customers that need items to be delivered. The formulation can be easily adapted to account for both pickup and delivery customers. Constraints (6) ensure that the vehicles leave the depot empty, constraints (7) update the load variables at the location nodes, and constraints (8) ensure that the vehicle capacity is never exceeded. Constraints (9)–(11) update the timing variable T_{ml}, where there are three possibilities: (i) if m is the first node on the shortest path towards node l, then due to constraints (9) T_{ml} is at least the departure time at node m on the shortest path towards node m; (ii) if m is an intermediate node on the shortest path towards node l, then due to constraints (10) T_{ml} is at least the departure time at the node visited before node m on the shortest path towards node l added to the time-dependent travel time between these nodes; and (iii) if m is the last node on the shortest path towards node l, i.e., $m = l$, then due to constraints (11) T_{ml} is at least the departure time at the node visited before node m on the shortest path towards node l added to the time-dependent travel time between these nodes added to the service time of node l. Note that the greater than or equal to signs in constraints (9)–(11) ensure that waiting time is allowed. Constraints (12)–(15) link paths and times via variable δ. More specifically, constraints (12) ensure that there is only one time interval associated with the departure time at a location node. Constraints (13) ensure that there is only one time period associated with the departure time at a node $i \neq l$ on the shortest path towards node l, and $\delta_{ilh} = 0\ \forall h \in \mathcal{H}$ if node i is not on the shortest path towards node l. Constraints (14)–(15) ensure that the departure time at node i on the shortest path towards node l is within the bounds of the time interval h for which $\delta_{ilh} = 1$. Constraints (16) define the variable δ^2_{pml}, which equals the time at which a vehicle leaves node p on the shortest path towards node l if arc (p,m) is used on the shortest path towards node l, and zero otherwise. Constraints (17) ensure that the variable δ^3_{pmlh} equals the travel time on arc (p,m) in time period h if the arc is traversed on the shortest path towards node l and a vehicle departs from node p on the shortest path towards node l in time period h,

and zero otherwise. Constraints (18)–(24) set the range and nature of the variables.

The following valid inequalities can be imposed to strengthen the formulation.

$$y_{ijl} = 0 \qquad \forall (i,j) \in \mathcal{A},\ l \in \mathcal{N}_o \tag{25}$$

$$H\overline{T} \sum_{j:(i,j)\in\mathcal{A}} y_{ijl} \geq T_{il} \qquad \forall i \in \mathcal{N},\ l \in \mathcal{N}'\backslash i \tag{26}$$

$$H\overline{T} y_{ijl} \geq \delta^2_{ijl} \qquad \forall (i,j) \in \mathcal{A},\ l \in \mathcal{N}' \tag{27}$$

$$\delta^3_{ijlh} \leq t_{ijh} \qquad \forall (i,j) \in \mathcal{A},\ l \in \mathcal{N}',\ h \in \mathcal{H} \tag{28}$$

$$\sum_{\substack{j\in\mathcal{N}_d \\ m\in\mathcal{N}\backslash\mathcal{N}_o:(m,j)\in\mathcal{A}}} y_{mjj} \geq \left\lceil \sum_{i\in\mathcal{N}'} d_i/Q \right\rceil \tag{29}$$

$$T_{ll} \geq T_{il} + s_l \qquad \forall l \in \mathcal{N}',\ i \in \mathcal{N}\backslash l \tag{30}$$

$$q_i \leq Q - \left(Q - \max_{j\in\mathcal{N}_c: j\neq i} \{d_j\} - d_i\right) \sum_{k\in\mathcal{N}_o} x_{ki} - \sum_{j\in\mathcal{N}_c} d_j x_{ij} \qquad \forall i \in \mathcal{N}_c. \tag{31}$$

Constraints (25) ensure that there are no arcs on the shortest paths towards the origin nodes. Due to constraints (26), if node i is not on the shortest path towards location node l, then $T_{il} = 0$. Constraints (27) and (28) ensure that if arc $(i,j) \in \mathcal{A}$ is not used on the shortest path towards location node l, then $\delta^2_{ijl} = 0$ and $\sum_{h\in\mathcal{H}} \delta^3_{ijlh} = 0$, respectively. Constraint (29) states that the number of vehicles used is at least the total demand divided by the capacity of a vehicle. Constraints (30) ensure that the departure time at a location node l on the shortest path towards node l, is at least the departure time of each of the other nodes on the shortest path towards node l added to the service time of node l. Constraints (31) are valid inequalities, as proven in Kara et al. [2004].

To better improve the bound and create other valid inequalities, let S_{ij} be the shortest path from node $i \in \mathcal{N}'$ to node $j \in \mathcal{N}'$ based on the smallest travel time $\min_{h\in\mathcal{H}} t_{klh}$ for each arc $(k,l) \in \mathcal{A}$. Then we can add the following constraints to improve the lower bounds:

$$\sum_{i\in\mathcal{N}_d} T_{ii} \geq \sum_{i,j\in\mathcal{N}'} x_{ij} S_{ij} + \sum_{i\in\mathcal{N}_c} s_i, \tag{32}$$

$$\sum_{i\in\mathcal{N}_d} T_{ii} \geq T_{jj} + \min_{d\in\mathcal{N}_d} S_{jd} \qquad \forall j \in \mathcal{N}_c, \tag{33}$$

$$T_{ii} \geq \sum_{p\in\mathcal{N}\backslash\mathcal{N}_o:(p,i)\in\mathcal{A}} y_{pii} \cdot \left\lfloor \min_{\substack{k\in\mathcal{N}_c \\ l\in\mathcal{N}_o \\ m\in\mathcal{N}_d}} \{s_k + S_{lk} + S_{km}\} \right\rfloor \qquad \forall i \in \mathcal{N}_d. \tag{34}$$

Constraint (32) ensures that the total of the arrival time of the vehicles back to the depot is at least the sum of all pairs of shortest paths $S_{ij},\ i,j \in \mathcal{N}'$, that are visited consecutively added to the total service time of the customers. Constraints (33) ensure that the total of the arrival time of the vehicles back to the depot is at least the departure time at customer node $j \in \mathcal{N}_c$ added to the shortest path from node j to the depot. Constraints (34) ensure that if a vehicle is used to visit customers, then its arrival time at the depot is at least the smallest sum over

$k \in \mathcal{N}_c$ of the service time of customer k added to the shortest path from the depot to customer k and the shortest path from customer k to the depot.

We can improve shortest paths S_{ij} and end up with shortest paths P_{ij}, which are computed as follows. The shortest path P_{ij} from the depot $i \in \mathcal{N}_d$ to customer $j \in \mathcal{N}_c$ can be computed as $P_{ij} = S_{ij}$. Let e_i^1 be the earliest departure time at customer node $i \in \mathcal{N}_c$ computed as $e_i^1 = s_i + \min_{j \in \mathcal{N}_o} P_{ji}$ and let $\tilde{e}_i^1$ be the time period corresponding to e_i^1. Then, the shortest path P_{ij} from customer node $i \in \mathcal{N}_c$ to depot node $j \in \mathcal{N}_d$ can be computed as the shortest path based on the smallest travel time $\min\limits_{h \in \mathcal{H}: h \geq \tilde{e}_i^1} t_{klh}$ for each arc $(k,l) \in \mathcal{A}$. Let e_i^2 be the latest arrival time at customer node $i \in \mathcal{N}_c$ computed as $e_i^2 = H\overline{T} - s_i - \min_{j \in \mathcal{N}_d} P_{ij}$ and let $\tilde{e}_i^2$ be the time period corresponding to e_i^2. Then, the shortest path P_{ij} between two customers nodes $i \in \mathcal{N}_c$ and $j \in \mathcal{N}_c$ can be computed based on the smallest travel time $\min\limits_{h \in \mathcal{H}: \tilde{e}_i^1 \leq h \leq \tilde{e}_j^2} t_{klh}$ for each arc $(k,l) \in \mathcal{A}$.

We can then add the following valid inequalities to improve the lower bounds:

$$\sum_{i \in \mathcal{N}_d} T_{ii} \geq \sum_{i,j \in \mathcal{N}'} x_{ij} P_{ij} + \sum_{i \in \mathcal{N}_c} s_i, \tag{35}$$

$$\sum_{i \in \mathcal{N}_d} T_{ii} \geq T_{jj} + \min_{d \in \mathcal{N}_d} P_{jd} \qquad \forall j \in \mathcal{N}_c, \tag{36}$$

$$T_{ii} \geq \sum_{p \in \mathcal{N} \setminus \mathcal{N}_o : (p,i) \in \mathcal{A}} y_{pii} \cdot \left[\min_{\substack{k \in \mathcal{N}_c \\ l \in \mathcal{N}_o \\ m \in \mathcal{N}_d}} \{ s_k + P_{lk} + P_{km} \} \right] \qquad \forall i \in \mathcal{N}_d, \tag{37}$$

$$\delta_{llh} = 0 \qquad \forall h < \tilde{e}_l^1,\ l \in \mathcal{N}_c, \tag{38}$$

$$\delta_{ljkh}^3 = 0 \qquad \forall l \in \mathcal{N}_c,\ h < \tilde{e}_l^1,\ j \in \mathcal{N} : (l,j) \in \mathcal{A},\ k \in \mathcal{N}', \tag{39}$$

$$T_{ll} \geq e_l^1 \qquad \forall l \in \mathcal{N}_c, \tag{40}$$

$$\delta_{ilh} = 0 \qquad \forall l \in \mathcal{N}_c,\ i \in \mathcal{N} \setminus \{l\},\ h > \tilde{e}_l^2, \tag{41}$$

$$\delta_{ijlh}^3 = 0 \qquad \forall l \in \mathcal{N}_c,\ h > \tilde{e}_l^2,\ i,j \in \mathcal{N} : (i,j) \in \mathcal{A}, \tag{42}$$

$$T_{il} \leq e_l^2 \qquad \forall l \in \mathcal{N}_c,\ i \in \mathcal{N} \setminus \{l\}, \tag{43}$$

$$\delta_{pml}^2 \leq e_l^2 \qquad \forall l \in \mathcal{N}_c,\ (p,m) \in \mathcal{A} : m \neq l. \tag{44}$$

Constraints (35)–(37) are stronger versions of (32)–(34), by considering the shortest paths P_{ij} instead of S_{ij}. Constraints (38)–(40) arise from the fact that a vehicle cannot depart from customer node $l \in \mathcal{N}_c$ earlier than e_l^1, whereas constraints (41)–(44) arise from the fact that a vehicle cannot arrive at customer node $l \in \mathcal{N}_c$ later than e_l^2.

We will run our experiments on three different models. The basic model, which we call *Model 1*, corresponds to (1)–(31). It consists of the set of constraints that define the problem, i.e., (1)–(24), and the first set of valid inequalities that are imposed to strengthen the formulation, i.e., (25)–(31). The second model, which we call *Model 2*, is an extension of Model 1 by adding a new set of valid inequalities that are imposed to improve the lower bound, i.e., (32)–(34). Hence,

Model 2 is composed of (1)–(34). The third model, which is called *Model 3*, is an extension of the basic model by adding another set of valid inequalities to improve the lower bound, i.e., (35)–(44). Thus, Model 3 is determined by (1)–(24) and (35)–(44).

3 Simulated annealing for the TDSPVRP

The problem introduced in this paper is significantly more difficult than the traditional VRP. Therefore, heuristic algorithms are required to solve it efficiently. Simulated annealing (SA) is a local search-based algorithm that implements a search mechanism to escape from local optima, first introduced by Kirkpatrick et al. [1983]. SA is one of the commonly used metaheuristics, and has been successfully applied to solve several variants of the VRP [Kuo, 2010, Liu et al., 2019, Guimarães et al., 2019], including one with a multigraph similar to ours [Ticha et al., 2019]. In this section we first present the general outline of the SA followed by the description of its components, namely a time-dependent shortest path algorithm, an insertion algorithm, and the neighborhood search operators.

3.1 The SA algorithm

SA uses a stochastic approach to search for new solutions, also called neighborhood solutions. Starting from the current solution X, if a better neighborhing solution X' is found, then the solution is accepted and the current solution X is replaced by X'. In order to escape from a local optimum, a worse solution is accepted with probability $e^{-\Delta/T}$. Parameter T is called the *temperature*, which gradually decreases during the search process, and Δ is calculated as $Fitness(X') - Fitness(X)$, where $Fitness(X)$ is the objective value of solution X. In the beginning of the search, the temperature is high, thus the probability of accepting worse solutions is also high. Over the course of several iterations, the temperature is reduced and as T gets lower, the probability of moving to a worse solution becomes smaller. At the end of the execution, the parameter T reaches the final temperature T_f. If Δ is high, it means that solution X' is much worse than the current solution X, meaning that the probability of accepting X' is smaller. The parameters of the algorithm are its initial temperature T_0, the cooling rate α, and the maximum number of iterations $itermax$. The parameter T_f is a function of T_0, α, and $itermax$ and is calculated as $T_f = T_0\alpha^{itermax}$

Our SA algorithm, summarized in Algorithm 1, starts with the creation of the initial solution (Section 3.3) using an insertion procedure (IP), then all the parameters are initialized (lines 3–4). At each iteration the current solution X undergoes a removal procedure (line 6) in which one of the five removal operators is applied to it. The choice of the operator is done randomly. A repair mechanism is applied on X' (line 7) to transform the partial solution into a feasible one by using the insertion operator (Section 3.4).

If the fitness of the new solution X' is better than that of the current solution X, the new solution is unconditionally accepted. If it is higher than that of the current solution, the probability of acceptance of X' is $e^{-\Delta/T}$ (line 11). When the solution is accepted, the value of X will be set to X'. If the objective value of the current solution X is better than that of the current best solution, we update X_{best} to be X as shown in line 14. After each iteration, the cooling function is applied to reduce the temperature T. We use a simple factor α to T to reduce the temperature as indicated in line 17.

Result: X_{best}: the best solution found so far by the algorithm

$X \leftarrow BestInsertion(\emptyset)$
$X_{best} \leftarrow X$
$T \leftarrow T_0$
$iter \leftarrow 0$
while $iter \leq itermax$ **do**
  $X' \leftarrow Removal(X)$
  $X' \leftarrow Repair(X')$
  $\Delta \leftarrow Fitness(X') - Fitness(X)$
  $iter = iter + 1$
  $r \leftarrow random(0, 1)$
  if $(\Delta < 0)||(r < e^{-\Delta/T})$ **then**
    $X \leftarrow X'$
    if $Fitness(X) < Fitness(X_{best})$ **then**
      $X_{best} \leftarrow X$
    end
  end
  $T = \alpha T$
end

Algorithm 1: Simulated annealing algorithm

3.2 Dynamic time-dependent shortest path algorithm

When the graph is time-dependent, the cost (time, emissions, etc) of arc (i, j) depends on the arrival time at node i. This means that even a small change in the solution can imply that the shortest path between any two customer nodes may change over time. For example, removing customer k from route (0, i, j, k, l, m, 0) impacts the visiting time of customers l, m and the returning time to the depot 0. If we want the new sequence (j, l, m, 0) to be performed in an optimal way, we need to update the time-dependent shortest paths between customers (j, l), (l, m) and (m, 0), considering the exact arrival time at each node.

As the classical Dijkstra algorithm for computing shortest paths cannot be used, we developed an exact dynamic time-dependent shortest path algorithm (DTDSPA) which identifies for each node the time it takes to reach it from one of its antecessors. Note that, FIFO property is necessary to prove that Dijkstra's algorithm terminates with a correct shortest paths tree on time-dependent networks [Nannicini et al., 2012].

The DTDSPA is used each time a customer is moved from a route to re-optimize the relevant section of the route. Let C be the set of non-visited nodes, which initially contains all the nodes of the problem, and t_{ijh} be the cost (time) of traversing arc (i, j) starting at time h. t_{ijh} depends on the arrival time at node i. Let $Parent[v]$ be the immediate predecessor of node v, and $Cost[v]$ the cost (time) of the shortest path to arrive at v. The pseudocode of the DTDSPA is presented in Algorithm 2.

```
1  Input: Graph 𝒢, start node s, start time h = 0
2  C ← all nodes
3  for each node v do
4  |  Cost[v] = ∞
5  |  Parent[v] = 0
6  end
7  Cost[s] = 0
8  while C ≠ ∅ do
9  |  find v ∈ C with min Cost[v]
10 |  C ← C\{v}
11 |  for each node w adjacent to v do
12 |  |  if Cost[v] + t_{vw(Cost[v])} < Cost[w] then
13 |  |  |  Cost[w] = t_{vw(Cost[v])} + Cost[v]
14 |  |  |  Parent[w] = v
15 |  |  end
16 |  end
17 end
```

Algorithm 2: Dynamic Time-dependent Shortest Path algorithm

3.3 Initial solution

The initial solution provided to the SA metaheuristic is based on a sequential insertion algorithm. At each iteration, the first unserved customer is selected. The insertion of this customer is evaluated in all positions over all the routes without violating the maximum travel time or capacity constraints. We also consider the creation of a new route if the number of vehicles allows it. The insertion yielding the smallest time increase is selected. The procedure stops when all the customers have been inserted. During this procedure, we use the DTDSPA to evaluate each insertion which requires to rebuild the solution (the shortest path between each pair of customers) starting from the predecessor of the inserted customer to the depot.

3.4 Neighborhood search operators

As commonly used when solving many vehicle routing problems, the neighborhood search part of the SA is based on some removal operators and on a repair operator. In a time-dependent context, we need to update all the shortest paths between all customer pairs following the position of the inserted or deleted customers. Thus, the impact of each modification is analyzed and the required paths updated by applying the DTDSPA.

3.4.1 Removal operators

In this section, we propose five removal operators for the problem at hand:

- **Random removal**: this operator selects n customers to remove from the current solution, where n is a random number between 1 and 4. The idea of randomly selecting customers helps diversify the search mechanism.

- **Random route removal**: this operator randomly selects a route and removes all its customers.
- **Minimum travel time route removal**: this operator selects the route which has the minimum total travel time and removes all its customers. The idea is to try to re-insert these customers into other routes aiming to minimize the total duration of the solution.
- **Minimum customer demands route removal**: this operator is similar to the Minimum travel time route removal but it selects the route with the minimum total demand.
- **Cluster removal**: this operator initially selects a random customer c and then removes it from the solution. Then the operator looks for the two closest customers to c and removes them. Three customers are removed in total. The closest customers are determined using the DTDSPA starting at $t = 0$. Note that up to three routes may be impacted.

3.4.2 Insertion operator

We propose the best insertion operator to repair the partial solutions:

- **Best insertion**: the best insertion operator works with the list of removed customers. For each customer, its best insertion position between two consecutive nodes among all the routes is found by using the DTDSPA to update the optimal shortest paths linking the customers of the routes.

4 Computational experiments

All experiments were performed on a desktop computer equipped with an Intel i7 processor and 64 GB of RAM, using CPLEX 12.8. The heuristic algorithm was implemented in the Apple Swift programming language. In Section 4.1 the generation and the characteristics of the instances are described. In Section 4.2 we highlight the importance of implicitly considering all shortest paths between two consecutive nodes. Section 4.3 describes the tuning of the SA parameters. The performance of the models compared to the SA and the impact of the different sets of valid inequalities are discussed in Section 4.4. A sensitivity analysis on the impact of travel times is presented in Section 4.5.

4.1 Data set generation

Instances were generated from real data obtained from Québec City, Canada, and were created as follows. Hundred of thousands GPS points were obtained from industrial partners operating in the Québec area. To each data point is associated the exact location of the truck, its speed, and the time of the measurement. Measurements were taken at around every 15 seconds, such that one is able to track its path as well the speed in each segment. By aggregating several observations for each street segment, we were able to estimate the average speed (and traveling time) for each street in any period of the day. The complete methodology can be found in Belhassine et al. [2018] where the authors present the procedure to transform geolocation observations of thousands of home delivery trips into congestion data on segments of a road network.

From this database, we selected representative areas of the city, discarded some small streets, and considered the travel time information that corresponds to business hours, between 7 AM and 8 PM. Following this procedure, we have generated 15 instances divided into three groups. Within each group, five instances range from 10 to 50 customers. Each group is characterized by its number of nodes, arcs and time periods as presented in Table 1. For groups S and M we divided traffic information into 15 minutes interval, thus for each arc we have 52 speed information from 7:00 AM to 8:00 PM. Group T is defined only from 7:00 AM to 2:00 PM. The largest instance has 50 customers on a graph containing 1612 nodes, with 2810 arcs over 52 time periods. Note that each arc has a travel time associated to each time period. We have preprocessed the graph, modifying some travel times in order to guarantee that the FIFO property holds. The complete set of instances and detailed results can be found at `https://www.leandro-coelho.com/instances/time-dependent-vrp/`.

Table 1: Summary of the instances.

Group	# instances	# customers	# nodes	# arcs	# time periods
T	5	10 – 50	357	616	28
S	5	10 – 50	696	1226	52
M	5	10 – 50	1612	2810	52

4.2 Importance of considering the underlying street network

In this section we illustrate that not only the travel time between an origin and a destination should be time-dependent, but we also need to consider the complete street network as a multigraph. For a given pair of origin and destination nodes, Figure 4 shows four different shortest travel time paths obtained at various departure times of the day respectively 9:00 AM for solution (a), 9:15 AM for solution (b), 9:45 AM for solution (c), and 1:00 PM for solution (d). We used the DTDSPA to determine the optimal path starting from the origin towards the destination.

Detailed results are given in Table 2, where we report *Solution* (the solution name), *Departure time* as the time of the departure from the origin, *Arrival time* as the arrival time at the destination, *Travel time* in seconds, and *Number of segments* in the path of the solution. These results show that not only the travel time varies significantly (from 1621 to 2280 seconds), the number of segments traveled also changes (from 103 to 115). If we look at solutions (b) and (c) which have almost the same travel time, we observe a very different route. Comparing (a) and (c), the three minutes difference lead to very different paths, with mostly unique segments. Also, the difference in departure times between solution (a) and solution (c) is less than one hour, yet the path of the solution differs dramatically. In fact solution (a) shares only 20% of the path with solution (c). Thus throughout the day, not only the travel times of the segments change, but also the related paths. Neglecting the whole underlying network may lead to important missed opportunities to optimize time spent in traffic. More important, a driver not aware of the changing paths may loose time if he travels on the wrong path.

(a) Travel time: 1621s, number of segments: 115

(b) Travel time: 1821s, number of segments: 103

(c) Travel time: 1802s, number of segments: 108

(d) Travel time: 2280s, number of segments: 115

Figure 4: Optimal time-dependent travel times and number of segments for the same origin-destination, across different starting times (see Table 2)

Table 2: Optimal time-dependent travel times for the same origin-destination, across different starting times (see Figure 4)

Solution	Departure time	Arrival time	Travel time (s)	Number of segments
a	09:00 AM	09:27 AM	1621	115
b	09:15 AM	09:35 AM	1821	103
c	09:45 AM	10:15 AM	1802	108
d	01:00 PM	01:38 PM	2280	115

4.3 Parameter tuning for the simulated annealing algorithm

The proposed SA algorithm works with three parameters: T_0, $itermax$ and α. Parameter $itermax$ defines the maximum number of iterations during the execution of the algorithm, and T_0 represents the initial temperature. At the end of the algorithm, we expect the final temperature to be T_f=0.1. Finally, α is the coefficient controlling the cooling rate. A proper parameter tuning is required to reach a good compromise between computing time and solution quality.

First, using an ad-hoc trial-and-error strategy and during the developments of the algorithm, we found four good settings for $itermax$ (1000, 3000, 6000 and 10000) and three for T_0 (500, 1000, 10000). Then we fixed α in such a way that starting at temperature T_0, we reach T_f after $itermax$ iterations. Thus we have $T_f = T_0\alpha^{itermax}$ and $\alpha = \sqrt[itermax]{\left(\frac{T_f}{T_0}\right)}$. The resulting 12 combinations of parameters (C1–C12) are presented in Table 3, where we also present the result of configuration CO which simply returns the initial solution ($itermax = 0$) value provided to the SA algorithm as explained in Algorithm 1. For each configuration of parameters we apply the SA heuristic on our set of 15 instances.

Table 3: Parameters configuration

Parameters	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
$itermax$	0	1000	1000	1000	3000	3000	3000	6000	6000	6000	10000	10000	10000
T_0	–	500	1000	10000	500	1000	10000	500	1000	10000	500	1000	10000
α	–	0.9915	0.9908	0.9886	0.9972	0.9969	0.9962	0.9986	0.9985	0.9981	0.9991	0.9991	0.9988

For each configuration we note the average solution value, the average computing time in seconds, the average deviation with respect to the best solution found by any of the 13 configurations for each instance, and the number of best solutions found by each heuristic configuration. Results of Table 4 show that configurations C9 and C12 are the best with respective deviations of only 0.08% and 0.07%, however, configuration C9 requires only 373 seconds as opposed to the 812 seconds required by C12. In the following, heuristics results are presented for the SA with

parameter configuration C9. Note that if running time is critical, configuration C5 is a good compromise with an average deviation of 0.16% in only 95 seconds. Configuration C0 shows that the creation of the initial feasible solution is quick, with an average computing time under 1s. Configuration C9 improves the C0 results by 3.2%.

Table 4: Heuristic configuration results

Configuration	Avg value	Avg time (s)	Avg dev (%)	# Best sol
C0	71633	0.68	3.35	0
C1	69467	23.0	0.22	6
C2	69468	31.0	0.21	5
C3	69563	48.2	0.32	5
C4	69475	65.4	0.24	6
C5	69413	94.5	0.16	9
C6	69396	155.9	0.14	6
C7	69402	128.6	0.15	6
C8	69373	193.6	0.12	8
C9	69360	373.3	0.08	8
C10	69385	253.0	0.13	7
C11	69370	379.7	0.10	6
C12	69335	812.5	0.07	8

4.4 Mathematical model evaluation

In this section we evaluate the performance of the three proposed models. Table 5 shows the average number of constraints and variables for Model 1. Clearly, even for the smallest instances the resulting models are very large with on average 1 300 000 constraints and 946 000 variables. For Model 2 and Model 3 we present the difference in percentage with respect to Model 1 for the number of constraints (Δ_C) and variables (Δ_V). Model 2 differs from Model 1 only by constraints (32)–(34) representing only $\mathcal{N}_c + \mathcal{N}_d + 1$ extra constraints, which is negligible among the more than a million of constraints. Compared to Model 1, the size of the problem significantly decreases for Model 3, namely the average number of constraints and variables is decreased by 5.45% and 5.01%, respectively. An explanation of the reduction in the number of constraints and variables in Model 3 is the introduction of constraints (38)–(44). Comparing Models 2 and 3, constraints (35)–(37) in Model 3 are stronger versions of (32)–(34) from Model 2 using the shortest paths P_{ij} instead of S_{ij}. Moreover, in Model 3 we have added constraints

(38)–(44) that are derived from the generation of the shortest paths P_{ij}. Thus, while Model 3 uses stronger versions of the shortest paths, it also introduces completely new constraints that can be seen as by-products from the generation of the shortest paths P_{ij}, which help reduce the problem size. As the models are clearly too large to be solved, we analyse their performance under different initialization procedures.

Table 5: Impact of the different models on the problem size

	Model 1		Model 2		Model 3	
Group	Constraints	Variables	Δ_C	Δ_V	Δ_C	Δ_V
T	1,331,382	946,945	0.00	0.00	–6.70	–6.14
S	4,414,168	3,146,947	0.00	0.00	–4.35	–4.00
M	5,167,447	3,645,341	0.00	0.00	–4.49	–4.13
Total	3,255,220	2,313,345	0.00	0.00	–5.45	–5.01

Table 6 presents the results when the models were run for two hours with no initial solution. As we can see, no model was able to find any feasible solution. Model 1 provides a lower bound equal to zero for 13 instances, while 2 instances could not have their models built (due to time or memory limitations). The valid inequalities used for Model 2 have an important impact helping provide lower bounds for every instance. Valid inequalities in Model 3 are even stronger, as the lower bounds obtained by Model 3 are on average 0.47% higher than those of Model 2. Obviously, all instances were executed up to the 2h limit.

In order to better evaluate the behavior of the model, we provide a warm start to CPLEX. This is achieved by using the solutions provided by configuration C0 of the simulated annealing. Table 7 presents these results. We report *Inst*, the name of the instance, and the solution value (z) from the C0 configuration. Models 1–3 were run for two hours and we present the value of the solution $\overline{z}$, the value of the lower bound $\underline{z}$, and *Gap* reported by CPLEX. Model M1 provides a lower bound equal to zero for 14 instances, while for instance T10, the lower bound provided by M1 is equal to 3.8 with a gap of 99.99%. Providing an initial feasible solution had positive effect on the lower bounds, as Model 2 now obtains five better lower bounds and Model 3 yields 13 better lower bounds.

Finally, we initialize the models with the solutions obtained by the complete SA algorithm with parameter configuration C9. Detailed computational results are given in Table 8.

Even with a good quality initial solution, the results show that Model 1 provides poor lower bounds. For this model, the gaps are between 99.99% and 100.00% and for 14 instances the lower bounds remain at 0.00. For Model 2, in which a set of valid inequalities are added to the basic model, the gaps have significantly improved, ranging between 4.82% and 8.34%. The average gap is now improved from 9.22% to 6.22%.

The valid inequalities of Model 3 are even stronger. The lower bounds obtained by Model 3 are on average 0.48% higher than those obtained by Model 2. Under Model 3 the gaps are between 4.42% and 7.44%, with an average of only 5.68%. This means that the solutions obtained by configuration C9 of our SA algorithm are not worse than 5.68% of the optimal solutions.

Table 6: Models performance with no initial solution

	Model 1		Model 2		Model 3	
Inst.	$\overline{z}$	$\underline{z}$	$\overline{z}$	$\underline{z}$	$\overline{z}$	$\underline{z}$
T10	–	0.00	–	26990.00	–	27401.76
T20	–	0.00	–	45400.66	–	45603.62
T30	–	0.00	–	76645.19	–	76860.36
T40	–	0.00	–	70866.25	–	70996.42
T50	–	0.00	–	81453.35	–	81646.80
S10	–	0.00	–	28739.73	–	29049.05
S20	–	0.00	–	52108.65	–	52375.00
S30	–	0.00	–	83234.00	–	83588.10
S40	–	0.00	–	71778.50	–	72016.16
S50	–	0.00	–	99413.03	–	99623.60
M10	–	0.00	–	29389.00	–	29678.04
M20	–	0.00	–	52778.69	–	53221.86
M30	–	0.00	–	84191.70	–	84717.67
M40	–	–	–	–	–	–
M50	–	–	–	–	–	–
Average	–	0.00	–	61768.37	–	62059.88

Table 7: Models performance with C0 configuration

	C0	Model 1			Model 2			Model 3		
Inst.	z	$\overline{z}$	$\underline{z}$	Gap (%)	$\overline{z}$	$\underline{z}$	Gap (%)	$\overline{z}$	$\underline{z}$	Gap (%)
T10	30139	30139	3.80	99.99	30139	26990.00	10.45	30139	27405.18	9.07
T20	50886	50886	0.00	100.00	50886	45376.70	10.83	50886	45606.17	10.38
T30	82804	82804	0.00	100.00	82804	76614.01	7.48	82804	76869.56	7.17
T40	77522	77522	0.00	100.00	77522	70862.74	8.59	77522	71064.64	8.33
T50	90264	90264	0.00	100.00	90264	81453.35	9.76	90264	81647.09	9.55
S10	31728	31728	0.00	100.00	31728	28747.70	9.39	31728	29028.90	8.51
S20	57462	57462	0.00	100.00	57462	52104.42	9.32	57462	52375.00	8.85
S30	89884	89884	0.00	100.00	89884	83234.00	7.40	89884	83588.36	7.00
S40	78774	78774	0.00	100.00	78774	71809.93	8.84	78774	72018.23	8.58
S50	106984	106984	0.00	100.00	106984	99449.73	7.04	106984	99650.16	6.86
M10	31645	31645	0.00	100.00	31645	29389.00	7.13	31645	29685.02	6.19
M20	60090	60090	0.00	100.00	60090	52778.85	12.17	60090	53232.71	11.41
M30	92198	92198	0.00	100.00	92198	84201.62	8.67	92198	84729.48	8.10
M40	82572	82572	0.00	100.00	82572	73042.43	11.54	82572	73487.15	11.00
M50	111546	111546	0.00	100.00	111546	100771.85	9.66	111546	101195.41	9.28
Average	71633	71633	0.25	100.00	71633	65121.76	9.22	71633	65438.87	8.68

Table 8: Models performance with C9 configuration

	SA	Model 1			Model 2			Model 3		
Inst.	z	$\overline{z}$	$\underline{z}$	Gap (%)	$\overline{z}$	$\underline{z}$	Gap (%)	$\overline{z}$	$\underline{z}$	Gap (%)
T10	28927	28927	3.99	99.99	28927	26990.00	6.70	28927	27401.75	5.27
T20	48346	48346	0.00	100.00	48346	45377.45	6.14	48346	45594.50	5.69
T30	80513	80513	0.00	100.00	80513	76616.48	4.84	80513	76865.16	4.53
T40	74850	74850	0.00	100.00	74850	70866.24	5.32	74850	71063.50	5.06
T50	85793	85793	0.00	100.00	85793	81453.35	5.06	85793	81647.09	4.83
S10	31362	31362	0.00	100.00	31362	28747.70	8.34	31362	29028.90	7.44
S20	55146	55146	0.00	100.00	55146	52105.76	5.51	55146	52375.00	5.02
S30	87450	87450	0.00	100.00	87450	83234.00	4.82	87450	83588.36	4.42
S40	76477	76477	0.00	100.00	76477	71809.21	6.10	76477	72018.23	5.83
S50	105174	105174	0.00	100.00	105174	99450.13	5.44	105174	99652.22	5.25
M10	31308	31308	0.00	100.00	31308	29396.00	6.11	31308	29698.83	5.14
M20	56963	56963	0.00	100.00	56963	52778.85	7.35	56963	53221.87	6.57
M30	90093	90093	0.00	100.00	90093	84191.70	6.55	90093	84729.48	5.95
M40	78975	78975	0.00	100.00	78975	73042.43	7.51	78975	73487.15	6.95
M50	109024	109024	0.00	100.00	109024	100771.85	7.57	109024	101195.41	7.18
Average	69360	69360	0.27	100.00	69360	65122.08	6.22	69360	65437.83	5.68

4.5 Impact of time-dependent traveling times

In Table 9 we give some insights on the impact of using accurate travel time information. Column SA reports previous heuristics results. We then run the SA algorithm with different values for the travel time on each segment. The results under columns *Min. travel time*, *Avg. travel time* and *Max. travel time* are obtained by taking, respectively, the minimum, the average and the longest travel time observed through the day for each street segment. The percentage gaps between *Obj* and *SA* are computed as $100(Obj - SA)/SA$. Solutions are marked with an asterisk when at least one vehicle returns too late to the depot.

Table 9: Impact of traffic

	SA	Min. travel time		Avg. travel time		Max. travel time	
Inst.	z	Obj	Gap (%)	Obj	Gap (%)	Obj	Gap (%)
T10	28927	27345	-5.47	31748	9.75	36270	25.38
T20	48346	46079	-4.69	50727	4.92	55914	15.65
T30	80513	77820	-3.34	83870	4.17	90604*	12.53
T40	74850	71967	-3.85	78249	4.54	85098*	13.69
T50	85793	82704	-3.60	90338	5.30	99022*	15.42
S10	31362	28925	-7.77	34600	10.32	39882	27.17
S20	55146	52461	-4.87	59397	7.71	65670	19.08
S30	87450	84116	-3.81	92825*	6.15	101839*	16.45
S40	76477	72366	-5.38	81741	6.88	90860	18.81
S50	105174	100466	-4.48	111796*	6.30	123097*	17.04
M10	31308	29905	-4.48	35766	14.24	42574	35.98
M20	56963	54139	-4.96	63152	10.86	73278	28.64
M30	90093	85622	-4.96	95033*	5.48	106153*	17.83
M40	78975	74681	-5.44	86953	10.10	100512	27.27
M50	109024	102473	-6.01	115084	5.56	129476*	18.76
Average	69360	66071	-4.87	74085	7.49	82683	20.65

Results of Table 9 show that neglecting time-dependent travel times leads to underestimate the average arrival times back at the depot by 4.87% when minimum traversal times are used. Note that companies that do not take any information on traffic into account will perform even

worse, as we did consider *some* real information by using the minimum traversal time as observed from the database, which can be slower than estimating the travelling time with the maximum allowed speed.

Surprisingly, running the SA algorithm with the average travel time did not provide a better estimation, with the duration of the solutions being overestimated on average by 7.49%. At the other extreme, the solutions under the maximum (worse) travel time column indicate just how bad solutions can get if one is trapped in traffic. Our experiments show that vehicles would arrive back to the depot on average 20.78% later than when the routes are optimized based on real traffic data, and many trucks exceed their driving duration constraint.

These results clearly show that neglecting traffic may result in substantial delays at the locations that need to be visited, which in turn would require more vehicles and more mileage to perform the deliveries. To conclude, these results show that the impact of traffic is large and it is important to incorporate it into practical routing models.

5 Conclusions

In this paper, we have introduced the time-dependent shortest path and vehicle routing problem. We provided different mathematical formulations for the problem and we developed valid inequalities to strengthen them and to improve the lower bounds. We have developed three models, differing by a set of valid inequalities developed to improve the lower bounds, based on different and more intricate shortest paths. A heuristic based on local search with a SA acceptance criteria is also developed. We have created a set of 15 instances generated from real traffic data on the road network in Québec City, Canada. The instances range from 357 nodes, 616 arcs, 28 time periods, and 10 customers up to 1612 nodes, 2810 arcs, 52 time periods, and 50 customers. These instances can serve as benchmark instances for the TDSPVRP. We have provided heuristic solutions based on the SA algorithm. Results have shown that the size of the instances under Models 1 and 2 are almost the same, whereas the number of constraints and variables is on average reduced by 5.45% and 5.01%, respectively, under Model 3. Providing the solutions of the SA algorithm as initial solutions, helped CPLEX find better lower bounds, but it could never improve the solutions we provided. The standard formulation of Model 1 provides very weak lower bounds, where the gap with the best found solutions is between 99.99% and 100.00%. The valid inequalities that are imposed to improve the lower bounds are strong and significantly improve the gaps, i.e., they are between 4.82% and 8.34% for Model 2, and between 4.42% and 7.44% for Model 3. The use of a stronger set of valid inequalities to improve the lower bounds, based on different shortest paths, provides lower bounds that are on average 0.48% higher. The SA heuristic provides high quality solutions, as confirmed by an average gap of only 5.68%.

We have also provided a sensitivity analysis that supports the importance of including time-dependent travel times in routing models, by showing that ignoring it can impose substantial delays. To find good upper bounds for the TDSPVRP, state-of-the-art heuristics and more elaborated exact methods should be developed in future research. The benchmark instances generated and the lower bounds provided in this research can be used to test the results of those heuristics.

References

E. Angelelli, V. Morandi, and M. G. Speranza. Congestion avoiding heuristic path generation for the proactive route guidance. *Computers & Operations Research*, 99:234–248, 2018.

T. Bektaş and G. Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011.

K. Belhassine, J. Renaud, J-P. Gagliardi, and L. C. Coelho. Improved home deliveries in congested areas using geospatial technology. Technical Report CIRRELT-2018-02, Québec, Canada, 2018.

M. Bruglieri, S. Mancini, F. Pezzella, and O. Pisacane. A path-based solution approach for the green vehicle routing problem. *Computers & Operations Research*, 103:109–122, 2019.

I. Chabini. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. *Transportation Research Record: Journal of the Transportation Research Board*, 1645:170–175, 1998.

H.-K. Chen, C.-F. Hsueh, and M.-S. Chang. The real-time time-dependent vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review*, 42(5):383–408, 2006.

K. L. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, 14(3):493–498, 1966.

B. Ding, J.X. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, pages 205–216, 2008.

A. V. Donati, R. Montemanni, N. Casagrande, A.E. Rizzoli, and L. M. Gambardella. Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, 185(3):1174–1191, 2008.

S. E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17(3): 395–412, 1969.

A. Franceschetti, D. Honhon, T. Van Woensel, T. Bektaş, and G. Laporte. The time-dependent pollution-routing problem. *Transportation Research Part B: Methodological*, 56:265–293, 2013.

T. A. Guimarães, L. C. Coelho, C. M. Schenekemberg, and C. T. Scarpin. The two-echelon multi-depot inventory-routing problem. *Computers & Operations Research*, 101:220–233, 2019.

H. Hashimoto, M. Yagiura, and T. Ibaraki. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, 5(2):434–456, 2008.

H. Heni, L. C. Coelho, and J. Renaud. Determining time-dependent minimum cost paths under several objectives. *Computers & Operations Research*, 105:102–117, 2019.

Y. Huang, L. Zhao, T. Van Woensel, and J.-P. Gross. The time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B: Methodological*, 95:169–195, 2017.

İ. Kara, G. Laporte, and T. Bektaş. A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for the capacitated vehicle routing problem. *European Journal of Operational Research*, 158(3):793–795, 2004.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte. The fleet size and mix pollution-routing problem. *Transportation Research Part B: Methodological*, 70:239–254, 2014.

Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte. The impact of depot location, fleet composition and routing on emissions in city logistics. *Transportation Research Part B: Methodological*, 84:81–102, 2016.

A. L. Kok, E. W. Hans, and J. M. J. Schutten. Vehicle routing under time-dependent travel times: the impact of congestion avoidance. *Computers & Operations Research*, 39(5):910–918, 2012.

Y. Kuo. Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering*, 59(1):157–165, 2010.

R. Liu, Y. Tao, and X. Xie. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research*, 101:250–262, 2019.

C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

G. Nannicini, D. Delling, D. Schultes, and L. Liberti. Bidirectional A$*$ search on time-dependent road networks. *Networks*, 59(2):240–251, 2012.

M I. Nasri, T. Bektaş, and G. Laporte. Route and speed optimization for autonomous trucks. *Computers & Operations Research*, 100:89–101, 2018.

A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3):607–625, 1990.

D. Soler, J. Albiach, and E. Martínez. A way to optimally solve a time-dependent vehicle routing problem with time windows. *Operations Research Letters*, 37(1):37–42, 2009. ISSN 0167-6377.

H. Ben Ticha, N. Absi, D. Feillet, and A. Quilliot. Multigraph modeling and adaptive large neighborhood search for the vehicle routing problem with time windows. *Computers & Operations Research*, 104:113–126, 2019.

A. K. Ziliaskopoulos and H. S. Mahmassani. Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications. *Transportation Research Record*, pages 94–94, 1993.