Optimization of the service start time for an elementary shortest path problem with time windows

Hande Küçükaydın[a,*], Yasemin Arda[b], Yves Crama[b]

[a] *MEF University, Department of Industrial Engineering, Istanbul, Turkey*
[b] *HEC Liège - Management School of the University of Liège, Liège, Belgium*

Abstract

We investigate an elementary shortest path problem with resource constraints where a single capacitated vehicle, initially located at a depot, must serve a set of customers while respecting their individual time windows. When the vehicle visits a customer, it delivers the customer's demand and collects a revenue in return for the delivery. The vehicle can start its trip at any desired time. The transportation cost is a function of both the total distance traveled and the duration of the assigned trip. The objective is to determine the service start time from the depot, the subset of customers to be served, and the trip to be performed so as to minimize the total loss, which is calculated as the difference between the transportation cost and the revenue collected from the customers. We develop two exact dynamic programming algorithms which can deal with an infinite number of Pareto-optimal states arising from the fact that the starting time and the duration of the trip act like continuous decision variables. We report computational results obtained with these algorithms and with a faster heuristic for the elementary shortest path problem. We also examine the performance of these algorithms when they are used to solve the pricing subproblem arising in the framework of a column generation algorithm for a related vehicle routing problem with time windows.

Keywords: elementary shortest path problem with resource constraints, dynamic programming, column generation

*Corresponding author
Email addresses: hande.kucukaydin@mef.edu.tr (Hande Küçükaydın), yasemin.arda@uliege.be (Yasemin Arda), yves.crama@uliege.be (Yves Crama)

1. Introduction

The *Shortest Path Problem with Resource Constraints* (SPPRC) consists in determining a least cost path that starts from a source and ends at a sink of a graph while satisfying a set of constraints that are defined over resources. Each resource corresponds to a quantity, such as the service time consumed or the load carried, whose value varies along the arcs of the path. The reader can refer to Irnich and Desaulniers (2005) for a classification of SPPRCs.

In an *Elementary Shortest Path Problem with Resource Constraints* (ESPPRC), the least cost feasible path has to be an elementary path, that is, a path in which no vertex is repeated. When the graph is acyclic, there is no need to impose elementarity. However, when the graph is cyclic with negative cost consumptions on the arcs, negative cycles may exist and the elementarity requirement has to be explicitly imposed in the problem.

In this article, we study an extension of the *Elementary Shortest Path Problem with Time Windows* where we assume that the vehicle collects a revenue whenever it serves a customer and the transportation cost is a linear function depending on both the distance traveled and the duration of the trip. The time at which the vehicle starts its service at the depot is a continuous decision variable. For a given path, different service start times at the depot may provide different values for the total waiting time of the vehicle and consequently, different values for the transportation cost. Then, the total loss associated with a trip is computed as the transportation cost of serving customers less the revenue collected from the customers who are served on the trip. The objective is to determine the service start time from the depot, the subset of customers to be served, and the trip to be performed so as to minimize the total loss function. This problem arises, for instance, as the pricing sub-problem of a vehicle routing problem (VRP) with time windows when the drivers are paid per time unit worked and the starting times of their shifts are to be determined by the decision maker.

We aim at solving this elementary shortest path problem to optimality using a dynamic programming (DP) algorithm. This approach requires that the total spent time and the total loss be defined as functions of the service start time at the depot. This leads to an infinite number of labels to be handled by the DP algorithm and requires developing appropriate dominance rules to eliminate irrelevant labels. The main contributions of our study are to formulate the resource extension functions of the total spent time and the total loss and to develop suitable dominance rules. In order to clarify these

claims, we now provide an overview of the literature on related problems.

1.1. Literature review

In an SPPRC, *Resource Extension Functions* (REFs) can be used to model the accumulated consumption of resources along partial paths. More precisely, if (i, j) is an arc, f_{ij}^r is the REF associated with resource r, r_i is the amount of resource r consumed up to vertex i, and R_i is the resource consumption vector at vertex i, then $f_{ij}^r(R_i)$ represents the minimum accumulated consumption of resource r along a partial path starting from the source and ending at vertex j after having traversed arc (i, j).

For each resource and each vertex j, there is a *resource interval* $[lb_j, ub_j]$. If the accumulated resource consumption $f_{ij}^r(R_i)$ at vertex j is less than the resource lower bound lb_j, then a certain amount of resource r has to be wasted by setting its value r_j equal to lb_j. A path is said to be resource-feasible if the consumption r_j of each resource r at each vertex j visited along the path can be defined to be at least equal to the corresponding value of the REF and at most equal to the resource upper bound ub_j. Irnich (2008) discusses some important REF properties that may lead to efficient algorithmic procedures. Besides, the author reviews different types of REFs that can be used for modeling complex resource interdependencies inherent in real-life problems.

SPPRCs can be solved using exact DP approaches with a pseudo-polynomial complexity (Desrochers and Soumis (1988); Desrochers et al. (1992)). A label indicating the resource consumptions is associated with each feasible partial path and the partial paths that cannot be extended to Pareto-optimal paths are eliminated with the help of dominance rules. The algorithm extends only the non-dominated labels into all feasible directions until the set of Pareto-optimal complete paths is obtained. In label-setting DP approaches (see Desrochers and Soumis (1988)), a label is chosen to be extended only if it is not possible to dominate this label in the following steps of the algorithm, whereas in label-correcting approaches (see Desrochers et al. (1992)), the non-dominance of a label cannot be guaranteed until the termination of the algorithm and the vertices are treated repeatedly throughout the algorithm to extend their labels. Lagrangian relaxation based bounding and network pre-processing techniques are proposed to improve the efficiency of the exact solution methods (Aneja et al. (1983); Beasley and Christofides (1989); Dumitrescu and Boland (2003)). Irnich and Desaulniers (2005) survey the commonly used SPPRC solution methods. For a more recent survey and for

exact solution approaches of some well-known classes of SPPRCs, the reader may refer to Di Puglia Pugliese and Guerriero (2013).

The ESPPRC in cyclic graphs is strongly NP-hard (Dror (1994)). In order to enforce elementary paths, Beasley and Christofides (1989) suggest associating an additional binary resource with each vertex. The binary resource takes value 1 if the vertex is visited on a partial path and takes value 0 otherwise. The authors underline the fact that this results in a high dimensional label space, but they do not carry out any further computational experiments. Feillet et al. (2004) propose an exact DP algorithm for the ESPPRC, which is an extension of the label-correcting algorithm of Desrochers (1988) developed for the solution of SPPRC and they adopt the idea of Beasley and Christofides (1989) to define the new DP dominance relation. The authors also propose the idea of *unreachable vertices* by identifying vertices which cannot be visited in any feasible extension of a partial path because of the resource limitations. This modification sharpens the dominance relation by discarding a larger number of non-useful labels and reducing the computation time. Following the mono-directional search proposed by Feillet et al. (2004), Righini and Salani (2006) propose a *Bounded Bi-Directional Search* (BBDS) to speed up the DP algorithm. In bi-directional search, labels are extended both forward from the source to its successors and backward from the sink to its predecessors. In order to reduce the number of generated labels, Righini and Salani (2006) limit the length of the forward and backward paths using two different bounding techniques, namely arc bounding and resource bounding. They also show how feasible complete paths can be obtained by linking the generated forward and backward partial paths. Computational experiments show that the BBDS algorithms outperform the mono-directional algorithm. The BBDS algorithm with resource bounding performs better than the one with arc bounding, which is efficient only when the optimal path is long in terms of the number of visited arcs. Boland et al. (2006) describe a label setting algorithm for the ESPPRC, where at each step the lexicographically minimal untreated label is chosen to be extended and marked as treated. Following the state space relaxation idea of Kohl (1995), the state space augmenting algorithm of Boland et al. (2006) solves iteratively a relaxed ESPPRC, considering the binary resources of only a subset of vertices until the least cost path is elementary. At each iteration, this subset is expanded by adding some of the vertices that are visited more than once in the previously generated optimal paths; four different state space augmenting strategies are proposed. Righini and Salani (2008) develop a sim-

ilar approach, called *Decremental State Space Relaxation* (DSSR), in which the relaxed ESPPRCs are solved using the BBDS algorithm suggested by Righini and Salani (2006). They also implement an exact branch-and-bound algorithm, where a lower bound is obtained at each node of the branch-and-bound tree using the BBDS algorithm with state space relaxation. The lower bound is improved by using a 2-cycle elimination technique that discards cycles of at most two arcs (Desrochers et al. (1992)). To compare the BBDS algorithm, the branch-and-bound method, and the DSSR algorithm, the authors work with the instances of Feillet et al. (2004) and Righini and Salani (2006), that are derived from Solomon's benchmark instances. The results show that the BBDS algorithm works faster when the resource constraints are very tight. However, when the constraints get looser, the computation time of the BBDS algorithm grows dramatically. The authors conclude that the DSSR algorithm dominates the other two approaches. Righini and Salani (2009) solve the Orienteering Problem with Time Windows (OPTW) using the DSSR algorithm introduced by Righini and Salani (2008). The performances of the state space augmenting approaches of Boland et al. (2006) are tested on instances obtained from the Solomon's data set and from the instances generated by Cordeau et al. (1997).

In addition to the exact solution procedures, heuristic approaches are also proposed for solving the ESPPRC. Irnich and Villeneuve (2006) introduce a k-cycle elimination algorithm for $k \geq 3$. Desaulniers et al. (2008) propose a tabu search heuristic for solving the ESPPRC. Baldacci et al. (2011) implement an ng-route relaxation of the ESPPRC that provides tighter lower bounds than the non-elementary relaxations. The proposed relaxation is shown to be very effective when the resource constraints are loose.

Several real-life problems can be modeled as ESPPRCs. As mentioned before, Righini and Salani (2009) model the OPTW as an ESPPRC. Moreover, ESPPRCs are faced when more complex problems are solved through column generation (CG) and branch-and-price (BP) techniques. For more detailed explications about CG procedures, the reader is referred to Lübbecke and Desrosiers (2005). Ceselli et al. (2009) develop a CG algorithm for a rich VRP where the pricing ESPPRC is solved using a BBDS algorithm. Gutiérrez-Jarpa et al. (2010) propose an exact BP algorithm for a VRP with deliveries, selective pickups, and time windows, where the ESPPRC solution procedure is accelerated by the DSSR approach. Salani and Vacca (2011) solve the discrete split delivery VRP with time windows using a BP method. The pricing sub-problem is solved again using a DSSR algorithm. A simple

greedy heuristic is employed to obtain a feasible solution at each node of the branch-and-bound tree. In addition to a tabu search heuristic for solving the ESPPRC designed for accelerating the CG algorithm, Desaulniers et al. (2008) compare the quality of the lower bounds when the pricing subproblem is relaxed in terms of elementarity. Different cutting strategies are introduced to strengthen the lower bounds. For recent developments in the exact solution methods of the VRP, the reader is referred to Baldacci et al. (2012).

As we mentioned earlier, the problem that we investigate in this article distinguishes itself from "classical" ESPPRCs by the fact that the start time from the depot (and hence, the duration of the trip) is a decision variable. As a consequence, some resource consumptions are functions of the service start time and DP algorithms must handle an infinite number of labels, thus requiring appropriate dominance rules.

The issue of infinite number of labels is first treated by Ioachim et al. (1998) in the context of a SPPRC with time windows when a linear cost is defined for each vertex as a function of the service start time at that vertex. They define effective dominance conditions for the cost resource and propose an exact DP algorithm to solve this problem. Tagmouti et al. (2007) study an ESPPRC without time windows but they define a service cost for each vertex as a function of the service start time at that vertex. Based on the developed dominance rules, a CG is proposed for the corresponding VRP. Liberatore et al. (2011) consider a VRP with soft time windows, where a linear penalty has to be paid only if a vertex is visited outside of its time window. They use a CG algorithm and develop DP dominance relations for the associated ESPPRC. Bettinelli et al. (2011) treat the service start time of each vehicle as a decision variable while proposing a branch-and-cut-and-price algorithm for the multi-depot heterogeneous VRP with time windows. However, they consider only vehicle fixed costs and distance-based routing costs. Dabia et al. (2013) also treat the service start time of each vehicle as a decision variable when the travel times over the arcs are time-dependent. They propose a BP algorithm for the time-dependent VRP with time windows that aims at minimizing the total duration of the routes.

The structure of the cost function that we treat in our study differs from that of the cited references. Consequently, the REFs formulated for the trip duration and the total loss, as well as the dominance relations, are novel and are developed here in a self-contained fashion. We incorporate our dominance rules into an exact mono-directional DP algorithm and an exact

DP algorithm with BBDS. We also propose a mono-directional DP heuristic which assumes a fixed start time from the depot for each path and applies the traditional dominance rules, and we test the performance of these different approaches.

Further, we develop a CG algorithm to solve the master problem formulation of the corresponding capacitated VRP with time windows. The DP heuristic is employed during the first iterations of the CG algorithm and one of the exact DP algorithms is called only when the heuristic fails to find a column with negative reduced cost.

The paper is organized as follows. Section 2 provides a formal description of the studied problem. In Section 3.1, the total spent time and the total loss functions are derived and the dominance rules are established for a forward DP algorithm. Section 3.2 describes the BBDS algorithm. The DP heuristic is introduced in Section 3.3. The results of the computational experiments with the DP algorithms and with the CG algorithm are reported in Section 4. Section 5 concludes the paper.

2. Problem Definition

Consider a directed graph $G = (V, U)$ with the vertex set V and the arc set U. The vertex set is defined as $V = D \cup \{0, n+1\}$, where $D = \{1, 2, ..., n\}$ is the set of n customers, 0 and $n+1$ represent two copies of the depot known as the source and the sink, respectively. Every customer $i \in D$ can be visited at most once. If a customer $i \in D$ is visited, a non-negative demand d_i has to be delivered to him and this visit brings in a non-negative revenue η_i. Furthermore, every customer $i \in D$ is associated with a service time $s_i \geq 0$ and a time window $[a_i, b_i]$, where a_i shows its earliest service start time and b_i the latest. Consequently, if a vehicle arrives at a customer i before a_i, it must wait until a_i to start serving the customer. At the depot, there is a single vehicle of capacity Q that can start serving the customers at any desired time. However, it can be used for at most a fixed amount of time S. It takes s_0 units of time to load the vehicle at the depot. We assume that $d_0 = d_{n+1} = 0$, $\eta_0 = \eta_{n+1} = 0$, and $[a_0, b_0] = [a_{n+1}, b_{n+1}] = (-\infty, \infty)$. The arc set is defined as $U = \{(i, j) : i, j \in V, i \neq j, a_i + s_i + t_{ij} \leq b_j\}$, where t_{ij} denotes the non-negative travel time from i to j. The non-negative travel distance from i to j is denoted by c_{ij}. We assume that both the travel times and distances satisfy the triangle inequalities.

We further assume that the transportation cost depends on the total distance traveled and the total amount of time that the vehicle spends in order to perform the assigned trip. To be more specific, define α as the cost per distance unit traveled and β as the cost per time unit use of the vehicle. For all $k \in D \cup \{0\}$, let T_k be the service start time of the vehicle at vertex k, and let T_{n+1} be the time when the vehicle returns to the depot. As the vehicle traverses an arc $(i, j) \in U$, it incurs the cost $\alpha c_{ij} + \beta(T_j - T_i)$. Taking into account the revenue η_i obtained by visiting customer i, the incurred loss on arc (i, j) can be written as $\alpha c_{ij} + \beta(T_j - T_i) - \eta_i$ (or equivalently the collected profit can be defined as $\eta_i - \alpha c_{ij} - \beta(T_j - T_i)$). The total loss incurred on a feasible path from source 0 to sink $n + 1$ is then the sum of the losses on the arcs traversed. Our objective is to determine the optimal value of the service start time T_0 at the source, along with the trip to be performed to reach the sink, so as to minimize the total loss (or equivalently to maximize the total profit).

A path $(0, ..., n + 1)$ is feasible only if it is elementary, respects the time windows of the customers, does not exceed the allowed duration S, and does not exceed the vehicle capacity Q. As shown by Irnich and Desaulniers (2005), these constraints can be modeled using resources, which leads to an ESPPRC. For our problem, we need to define $5+n$ different resources to establish the feasibility of possible paths. Indeed, a time resource needs to be defined to ensure that the customers are served within their given time windows. The value of the time resource at any vertex i indicates the service start time T_i at i. Given the value of the time resource, the value of the total spent time resource at vertex i can easily be determined as $T_i + s_i - T_0$. In order to respect the vehicle capacity, we need to define one more resource related to total delivered demand. The value of this resource at any vertex i indicates the demand already delivered by the vehicle after visiting vertex i. Finally, we define a binary resource for each vertex in V to enforce the elementarity of the path, where a resource is consumed right after the corresponding vertex is visited.

Then, a path is feasible if, at all vertices of the path, the value of each resource falls within an interval called resource interval. For the time resource, the resource window is simply the time window $[a_i, b_i]$ for every vertex $i \in V$. For the total spent time resource and the total delivered demand resource, the resource windows are given as $[0, S]$ and $[0, Q]$, respectively, at every vertex. Finally, the elementarity resource window is defined as $[0, 1]$ for all vertices. If the value of a resource exceeds the upper bound of its associated

window, then the related path is marked as infeasible. We now describe a DP algorithm that solves this ESPPRC exactly.

3. Solution Procedures

The procedure we propose is based on the DP algorithm given by Righini and Salani (2006, 2008). We first present an algorithm considering only the forward-directional search, and then we introduce the bi-directional search version of the same algorithm.

3.1. A Dynamic Programming Algorithm

The algorithm is based on assigning *labels* to every vertex $i \in V$, where a label represents a partial path from source 0 to vertex i and is composed of different components. For our problem, a label at any vertex i can be denoted by $L_i = (Z_i, T_i, H_i, Del_i, (El_k)^i_{k\in V})$, where Z_i is the loss of the partial path ending at vertex i and T_i, H_i, Del_i, $(El_k)^i_{k\in V}$ show the value of the service start time, total spent time, total delivered demand, and elementarity resources, respectively. The algorithm starts with an initial feasible label at source 0 and extends labels over and over again to obtain new labels. This is achieved by extending every feasible label of a vertex i along all arcs $(i, j) \in U$. Every time a label $L_i = (Z_i, T_i, H_i, Del_i, (El_k)^i_{k\in V})$ is extended from a vertex i to a successor vertex j to get a new label $L_j = (Z_j, T_j, H_j Del_j, (El_k)^j_{k\in V})$, the loss and the resource values of the new label at j must be computed using the following REFs:

$$
\begin{aligned}
&Z_j = Z_i + \alpha c_{ij} + \beta(T_j - T_i) - \eta_i \\
&T_j = \max\{a_j, T_i + s_i + t_{ij}\} \\
&H_j = T_j + s_j - T_0 \\
&Del_j = Del_i + d_j \\
&El^j_k = \begin{cases} El^i_k + 1, & \text{if } k = j \\ El^i_k, & \text{if } k \neq j \end{cases} \qquad \text{for all } k \in V
\end{aligned}
$$

A label L_j is feasible only if $T_j \leq b_j$, $H_j \leq S$, $Del_j \leq Q$, and $El^j_k \leq 1$ for all $k \in V$. If at least one of the components exceeds the corresponding upper bound, the label is removed from further consideration.

As mentioned above, the algorithm starts with an initial feasible label at source 0. This initial label is chosen to be $L_0 = (0, 0, 0, 0, (0)_{k\in V})$ in a traditional DP algorithm (see Gutiérrez-Jarpa et al. (2010); Righini and Salani (2006, 2008)). However, in our problem setting, the vehicle can depart from the depot at any point in time. Thus, we consider the initial label as $L_0 = (0, T_0, 0, 0, (0)_{k\in V})$ including the continuous decision variable $T_0 \in (-\infty, \infty)$. Furthermore, since there are several paths from source 0 to any vertex i, we have to apply dominance rules to eliminate those that are not Pareto-optimal. If there are two labels $L_i^1 = (Z_i^1, T_i^1, H_i^1, Del_i^1, (El_k^1)_{k\in V}^i)$ and $L_i^2 = (Z_i^2, T_i^2, H_i^2, Del_i^2, (El_k^2)_{k\in V}^i)$ representing two distinct paths from source 0 to vertex i, then L_i^1 dominates L_i^2 if and only if $Z_i^1 \leq Z_i^2$, $T_i^1 \leq T_i^2$, $H_i^1 \leq H_i^2$, $Del_i^1 \leq Del_i^2$, $(El_k^1)^i \leq (El_k^2)^i$ for all $k \in V$, and $L_i^1 \neq L_i^2$ meaning that at least one of the inequalities is strict. However, in our problem, the service start time T_j, the total spent time H_j, and the loss Z_j at any vertex j are functions of T_0 and can be denoted by $T_j(T_0)$, $H_j(T_0)$, and $Z_j(T_0)$, respectively. Hence, the DP algorithm must take into account an infinite number of Pareto-optimal labels which means for us that we first have to redefine the resource extension functions and then develop new dominance rules that will help us to cut the number of created labels, accordingly. In the next subsection, we explicitly define the extension functions for the service start time, the time cost, and the total loss.

3.1.1. Resources as Functions of the Start Time

Assume that we have a forward partial path starting from source 0 and visiting i more vertices. Let V_j be the vertex at the $(j+1)$st position of the partial path. Then, a partial path starting from V_0 and ending at a vertex V_i is given as (V_0,V_1,V_2,...,V_i), where V_0 represents the source. For simplicity, we define $\bar{t}_{V_{j-1},V_j}$ as the sum of the service time $s_{V_{j-1}}$ of V_{j-1} and the travel time t_{V_{j-1},V_j} from V_{j-1} to V_j, i.e. $\bar{t}_{V_{j-1},V_j} = s_{V_{j-1}} + t_{V_{j-1},V_j}$, and furthermore we denote the index V_j as j in the sequel, e.g. we denote the service start time $T_{V_j}(T_0)$ at V_j as $T_j(T_0)$. When the vehicle starts serving V_0 at T_0, the service start time $T_1(T_0)$ at V_1 is computed as

$$T_1(T_0) = \max\left\{a_1, T_0 + \bar{t}_{01}\right\}. \tag{1}$$

After V_1, the vehicle should visit V_2 following the given path and the service start time $T_2(T_0)$ at V_2 is similarly computed as

$$T_2(T_0) = \max\left\{a_2, T_1(T_0) + \bar{t}_{12}\right\}. \tag{2}$$

Similarly, we get for all $i > 1$

$$T_i(T_0) = \max\left\{a_i, T_{i-1}(T_0) + \bar{t}_{i-1,i}\right\}. \tag{3}$$

So, we obtain $T_i(T_0)$ as a function consisting of several nested maximum functions. Clearly, the first function (1) is a piecewise linear function composed of two linear pieces, one constant and one non-constant piece. At the second extension given by (2), first $\bar{t}_{12}$ is added to (1) which has the effect of moving up (1) by $\bar{t}_{12}$ units and thus obtaining again a piecewise linear function with two pieces. Then, the maximum of a_2 and the piecewise linear function $\max\left\{a_1, T_0 + \bar{t}_{01}\right\} + \bar{t}_{12}$ clearly provides another piecewise linear function with two pieces. Continuing the extensions in the same manner will only generate piecewise linear functions with two pieces.

Since the vehicle can start at any time, we initially choose T_0 in $(-\infty, \infty)$. Then, the vehicle will spend at least $\sum_{k=0}^{j-1} \bar{t}_{k,k+1}$ time units to reach any vertex V_j. For simplicity, we denote $\sum_{k=0}^{j-1} \bar{t}_{k,k+1}$ by θ_j. Thus, in order to be able to serve vertex V_j, T_0 must be less than or equal to $b_j - \theta_j$. Consequently, if the vehicle follows the partial path (V_0,V_1,...,V_i), the latest feasible start time ℓ_i of the vehicle from V_0 is expressed as

$$\ell_i = \min_{1 \le j \le i} \{b_j - \theta_j\}. \tag{4}$$

Whenever the time resource is extended from a vertex V_{j-1} to V_j, the latest feasible start time from V_0 should be updated as $\ell_j = \min\{\ell_{j-1}, b_j - \theta_j\}$, where $\ell_0 = \infty$. As a result, T_0 is restricted to be in the interval $(-\infty, \ell_i]$ which thus defines the domain of the function T_i. The fact that T_0 is restricted to the interval $(-\infty, \ell_i]$ now changes the number of linear pieces that the function T_i consists of. At the first extension from V_0 to V_1, $T_1(T_0)$ always consists of two linear pieces. However, starting from the second extension the number of linear pieces can change. We explained above that at the second extension (2) the piecewise linear function given by (1) is moved up by $\bar{t}_{12}$ units and then the maximum of a_2 and the piecewise linear function $T_1(T_0) + \bar{t}_{12}$ provides a new piecewise linear function $T_2(T_0)$. However, now we know that T_0 can take at most the value ℓ_2 and if $\max\left\{a_2, a_1 + \bar{t}_{12}\right\} \ge T_1(T_0) + \bar{t}_{12}$ for $T_0 \in (-\infty, \ell_2]$, then the new function $T_2(T_0)$ will be equal to $\max\left\{a_2, a_1 + \bar{t}_{12}\right\}$ which means that $T_2(T_0)$ is constant on $(-\infty, \ell_2]$. Once we obtain a constant function at

any extension, we continue to obtain constant functions in subsequent extensions since any a_j is either greater or less than the constant defining the time resource function. Hence, all time resource functions are composed of either two pieces or one piece.

In addition to the latest feasible start time ℓ_i from V_0, we need to define the earliest feasible service start time $\tilde{a}_i$ at vertex V_i. Observe that $\tilde{a}_i$ is not always equal to a_i, since the vehicle can arrive V_i after a_i. In such a case, $\tilde{a}_i$ will be equal to $\tilde{a}_{i-1} + \overline{t}_{i-1,i}$. Hence, if the vehicle follows the partial path (V_0,V_1,...,V_i), earliest service start time $\tilde{a}_i$ at vertex V_i can be determined by

$$\tilde{a}_i = \max\left\{a_i, \tilde{a}_{i-1} + \overline{t}_{i-1,i}\right\} \tag{5}$$

for $i > 0$, where $\tilde{a}_0 = -\infty$. Thus, the vehicle cannot start serving V_i before $\tilde{a}_i$

If we have a partial path (V_0,V_1,...,V_i), the initial start time function $T_1(T_0)$ clearly consists of two pieces unless $a_1 = b_1$. As we continue with extensions of the start time functions, we obtain new functions having either two pieces or one piece, depending on the following conditions at each vertex V_i:

- Condition I: $\tilde{a}_i < \ell_i + \theta_i$.
- Condition II: $\tilde{a}_i \geq \ell_i + \theta_i$.

Indeed, we can explicitly define the function $T_i(T_0)$ for any partial path $(V_0, V_1, \ldots, V_i)$ as follows:

Proposition 1. *For a feasible partial path $(V_0, V_1, \ldots, V_i)$, the service start time $T_i(T_0)$ at vertex V_i for $i \geq 0$ is computed as follows:*

- *When Condition I holds,*

$$T_i(T_0) = \begin{cases} \tilde{a}_i, & \text{if } T_0 \leq \tilde{a}_i - \theta_i \\ T_0 + \theta_i, & \text{if } \tilde{a}_i - \theta_i \leq T_0 \leq \ell_i. \end{cases}$$

- *When Condition II holds,*

$$T_i(T_0) = \tilde{a}_i \text{ for } T_0 \leq \ell_i.$$

Proof. Although the distinction between Condition I and Condition II will be useful in our subsequent discussion, let us start by observing that the statement of the proposition can be reworded as follows: for all $T_0 \leq \ell_i$,

$$T_i(T_0) = \begin{cases} \tilde{a}_i, & \text{if } T_0 \leq \tilde{a}_i - \theta_i \\ T_0 + \theta_i, & \text{if } \tilde{a}_i - \theta_i \leq T_0 \leq \ell_i \end{cases} \tag{6}$$

(indeed, when Condition II holds, the non-constant piece of this function simply vanishes). By induction, we can assume that Eq. (6) holds for vertex V_{i-1}, so that $T_{i-1}(T_0)$ consists of a constant piece with value $\tilde{a}_{i-1}$ followed by a unit-slope piece. Now, $T_i(T_0)$ is defined by Eq. (3). Let us consider the two terms of this equation.

If $a_i < \tilde{a}_{i-1} + \bar{t}_{i-1,i}$, then $a_i < T_{i-1}(T_0) + \bar{t}_{i-1,i}$ for $T_0 \leq \ell_{i-1}$ by Eq. (6), and hence $T_i(T_0) = T_{i-1}(T_0) + \bar{t}_{i-1,i}$ by Eq. (3). So, by Eq. (6) its constant piece is:

$$T_i(T_0) = \tilde{a}_{i-1} + \bar{t}_{i-1,i} \quad \text{if } T_0 \leq \tilde{a}_{i-1} - \theta_{i-1}$$

or equivalently by Eq. (5)

$$T_i(T_0) = \tilde{a}_i \quad \text{if } T_0 \leq \tilde{a}_i - \theta_i$$

since $\tilde{a}_{i-1} - \theta_{i-1} = \tilde{a}_i - \theta_i$. On the other hand, the non-constant piece of $T_i(T_0)$ is by Eq. (6):

$$T_i(T_0) = T_0 + \theta_{i-1} + \bar{t}_{i-1,i} = T_0 + \theta_i \quad \text{if } \tilde{a}_i - \theta_i \leq T_0 \leq \ell_i.$$

since $\tilde{a}_{i-1} - \theta_{i-1} = \tilde{a}_i - \theta_i$ and $\ell_i \leq \ell_{i-1}$.

Suppose now alternatively that $a_i \geq \tilde{a}_{i-1} + \bar{t}_{i-1,i}$. Then, $\tilde{a}_i = a_i$ by Eq. (5) and, by Eq. (3),

$$T_i(T_0) = a_i = \tilde{a}_i \quad \text{if } T_0 \leq \tilde{a}_{i-1} - \theta_{i-1} \leq \tilde{a}_i - \theta_i.$$

However, for $\tilde{a}_{i-1} - \theta_{i-1} \leq T_0 \leq \tilde{a}_i - \theta_i$, $T_{i-1}(T_0) = T_0 + \theta_{i-1}$. Thus, for $T_0 \in [\tilde{a}_{i-1} - \theta_{i-1}, \tilde{a}_i - \theta_i]$, $T_{i-1}(T_0) + \bar{t}_{i-1,i}$ takes a value on the interval $\left[\tilde{a}_{i-1} + \bar{t}_{i-1,i}, \tilde{a}_i\right]$. So, $\max\left\{a_i, T_{i-1}(T_0) + \bar{t}_{i-1,i}\right\} = a_i = \tilde{a}_i$ for $T_0 \in [\tilde{a}_{i-1} - \theta_{i-1}, \tilde{a}_i - \theta_i]$. Hence,

$$T_i(T_0) = \tilde{a}_i \quad \text{if } T_0 \leq \tilde{a}_i - \theta_i.$$

Moreover,

$$T_i(T_0) = T_{i-1}(T_0) + \bar{t}_{i-1,i} = T_0 + \theta_{i-1} + \bar{t}_{i-1,i} = T_0 + \theta_i$$

otherwise. □

Thus, if Condition I is satisfied, $T_i(T_0)$ is a function with two pieces, whereas it is a constant function otherwise. Clearly, the constant piece of the start time function $T_i(T_0)$ will always be equal to $\tilde{a}_i$ regardless of the number of pieces that the function contains. If the start time function consists of two pieces, the second piece is a linear function of the form $T_0 + \theta_i$ whose slope is equal to 1 and T_i-intercept is θ_i.

Now that we are able to explicitly define the start time functions $T_i(T_0)$, we can also express the cost incurred to serve the vertices. Along the partial path (V_0,V_1,...,V_i), the time spent by the vehicle is $(T_i(T_0) - T_0) + s_i$. Thus, the cost incurred while serving the vertices is $\beta\,(T_i(T_0) - T_0) + \beta s_i$, where β is the cost per time unit use of the vehicle. Since βs_i is a constant term, it will be dropped in the sequel. Apparently, the time cost is also a function of T_0 which is denoted by $C_i(T_0)$. Following the definition of the start time function $T_i(T_0)$ given above, the time cost function of the partial path (V_0,V_1,...,V_i) is given as follows for all $T_0 \leq \ell_i$:

- When Condition I holds,

$$C_i(T_0) = \begin{cases} \beta(\tilde{a}_i - T_0), & \text{if } T_0 \leq \tilde{a}_i - \theta_i \\ \beta\theta_i, & \text{if } \tilde{a}_i - \theta_i \leq T_0 \leq \ell_i. \end{cases}$$

- When Condition II holds,

$$C_i(T_0) = \beta(\tilde{a}_i - T_0) \text{ for } T_0 \leq \ell_i.$$

In order to visualize and understand both start time and time cost functions, we consider next an example.

Example 1: Consider a partial path (V_0,V_1,V_2,V_3), where V_0 represents the source 0. The time windows of V_1, V_2, V_3 are given by $[8, 15]$, $[10, 16]$, $[20, 23]$, respectively, and furthermore $\bar{t}_{01} = 1$, $\bar{t}_{12} = 3$, and $\bar{t}_{23} = 2$. For simplicity, we assume that $\beta = 1$. First, we compute the earliest service start time $\tilde{a}_1$ and the latest feasible start time ℓ_1 from the depot for V_1 as $\tilde{a}_1 = \max\{8, -\infty\} = 8$ and $\ell_1 = 15 - 1 = 14$. Since $\tilde{a}_1 < \ell_1 + \theta_1$, Condition I is clearly satisfied as $\theta_1 = 1$. Thus, the start time function $T_1(T_0)$ is a piecewise linear function with two pieces and is given by $T_1(T_0) = \begin{cases} 8, & \text{if } T_0 \leq 7 \\ T_0 + 1, & \text{if } 7 \leq T_0 \leq 14 \end{cases}$. Similarly, the time cost func-

tion is a piecewise linear function with two pieces and is given by $C_1(T_0) = \begin{cases} 8 - T_0, & \text{if } T_0 \leq 7 \\ 1, & \text{if } 7 \leq T_0 \leq 14 \end{cases}$. Both functions are depicted in Figure 1.

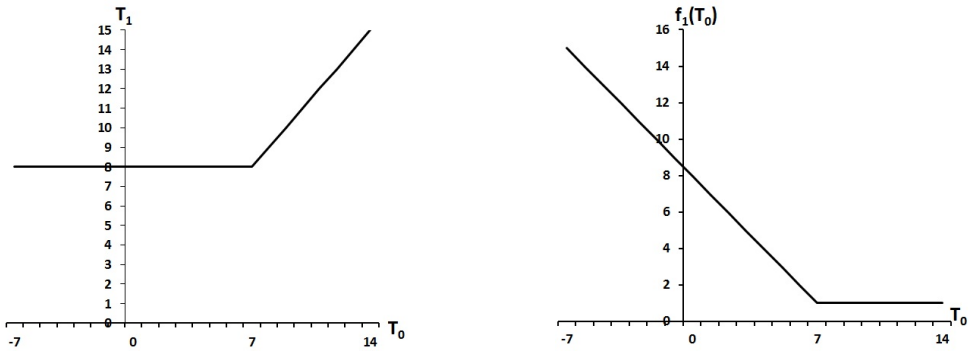

Figure 1: Start time function T_1 and time cost function C_1

It is clear that if the vehicle starts from the depot at $T_0 < 7$, then the service start time T_1 at V_1 is always equal to 8 which means that the vehicle must wait since it arrives at V_1 before a_1. As can be seen, for $T_0 < 7$ the time cost C_1 is linearly decreases as the start time T_0 increases, since the waiting time decreases. For $T_0 \geq 7$, the service start time T_1 at V_1 begins to increase, since the vehicle arrives at V_1 after a_1. In such a case, the vehicle does not wait and immediately starts serving the vertex. Furthermore, it can be observed that the time cost C_1 is a constant equal to $\theta_1 = 1$. Thus, for $\tilde{a}_1 - \theta_1 \leq T_0 \leq \ell_1$, the time cost is minimized and is constant, since it does not involve the waiting of the vehicle.

As the vehicle moves further to V_2, first $\tilde{a}_2$ and ℓ_2 must be updated following the equations (5) and (4), respectively: $\tilde{a}_2 = \max\{10, 8+3\} = 11$ and $\ell_2 = \min\{14, 16-4\} = 12$, where $\theta_2 = 4$. Clearly, Condition I is satisfied again, since $\tilde{a}_2 < \ell_2 + \theta_2$. $T_2(T_0)$ and $C_2(T_0)$ are depicted in Figure 2 and defined as follows: $T_2(T_0) = \begin{cases} 11, & \text{if } T_0 \leq 7 \\ T_0 + 4, & \text{if } 7 \leq T_0 \leq 12 \end{cases}$ and

$C_2(T_0) = \begin{cases} 11 - T_0, & \text{if } T_0 \leq 7 \\ 4, & \text{if } 7 \leq T_0 \leq 12 \end{cases}$.

The last visited vertex is V_3 whose time window is $[20, 23]$. We determine $\tilde{a}_3$ and ℓ_3 as $\tilde{a}_3 = \max\{20, 11+2\} = 20$ and $\ell_3 = \min\{12, 23-6\} = 12$,

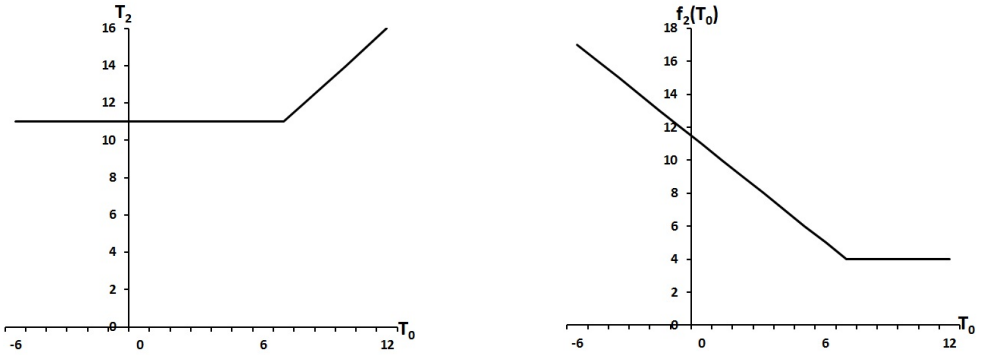

Figure 2: Start time function T_2 and time cost function C_2

where $\theta_3 = 6$. This time, Condition II holds since $\tilde{a}_3 > \ell_3 + \theta_3$. $T_3(T_0)$ and $C_3(T_0)$ are shown in Figure 3 and given as follows: $T_3(T_0) = 20$ for $T_0 \leq 12$ and $C_2(T_0) = 20 - T_0$ for $T_0 \leq 12$.

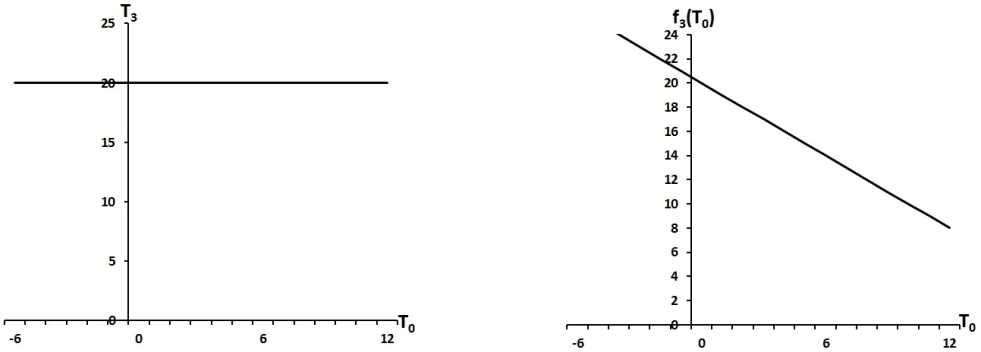

Figure 3: Start time function T_3 and time cost function C_3

The latest feasible start time ℓ_3 from the depot is in fact determined by V_2, since if the vehicle starts after 12, it is impossible that the vehicle visits V_2 on the given path. However, even if the vehicle starts from the depot at $T_0 = 12$, it arrives at V_2 at 16, at V_3 at 18 and waits until 20 as it is the earliest possible service start time of V_3. As a result, $T_3(T_0)$ turns out to be a constant function showing that the path (V_0,V_1,V_2,V_3) creates waiting time. On the other hand, it is obvious that starting from the depot as late as possible decreases the amount of waiting time and thus decreases the time cost as demonstrated by Figure 3. We can observe that the time cost is

minimized at $T_0 = \ell_3 = 12$. □

Besides the time cost, a total loss function additionally involves the travel distance cost and the revenues collected at vertices and is calculated as $Z_i(T_0) = C_i(T_0) + \alpha \sum_{k=0}^{i-1} c_{k,k+1} - \sum_{k=0}^{i-1} \eta_k$ at any vertex V_i, where $\alpha \sum_{k=0}^{i-1} c_{k,k+1}$ is the total traveled distance cost and $\sum_{k=0}^{i-1} \eta_k$ is the sum of the revenues on a path (V_0,...,V_i). Thus, in order to determine the total loss $Z_i(T_0)$, we only need to add the constant term $\alpha \sum_{k=0}^{i-1} c_{k,k+1} - \sum_{k=0}^{i-1} \eta_k$ to the time cost function $C_i(T_0)$ which has the effect of shifting it up or down.

Example 2: We consider the same partial path of the previous example and now determine the total loss of V_2 and V_3. At V_2, the total loss $Z_2(T_0)$ is computed by $Z_2(T_0) = C_2(T_0) + \alpha(c_{01} + c_{12}) - (\eta_0 + \eta_1)$, whereas $Z_3(T_0)$ at V_3 is calculated as $Z_3(T_0) = C_3(T_0) + \alpha(c_{01} + c_{12} + c_{23}) - (\eta_0 + \eta_1 + \eta_2)$. For now assume that $c_{01} = 0.5$, $c_{12} = 2$, and $c_{23} = 1$, the unit distance cost α is equal to 1 and the revenues are given as $\eta_1 = \eta_2 = \eta_3 = 10$ (note that since V_0 represents the source, its revenue η_0 is equal to 0). We first concentrate on the total loss function at V_2. The added constant term of this function is equal to $\alpha(c_{01}+c_{12})-(\eta_0+\eta_1) = -7.5$, in other words the time cost function $C_2(T_0) = \begin{cases} 11 - T_0, & \text{if } T_0 \leq 7 \\ 4, & \text{if } 7 \leq T_0 \leq 12 \end{cases}$ is shifted down by -7.5. The new function can easily be constructed as $Z_2(T_0) = \begin{cases} 3.5 - T_0, & \text{if } T_0 \leq 7 \\ -3.5, & \text{if } 7 \leq T_0 \leq 12 \end{cases}$ and displayed by Figure 4.

Similarly, we add the constant term $\alpha(c_{01}+c_{12}+c_{23})-(\eta_0+\eta_1+\eta_2) = -16.5$ to C_3 to obtain the total loss function $Z_3(T_0) = C_3(T_0)-16.5 = 3.5-T_0$ for $T_0 \leq 12$ as demonstrated by Figure 5. □

Figures 4 and 5 clearly show for which value of T_0 the total loss is minimized. For a piecewise linear total loss function, the total loss is minimized for $T_0 \in [\tilde{a}_i - \theta_i, \ell_i]$, while for a single-piece linear total loss function there is a unique minimizer that is given by ℓ_i.

Having defined the service start time, the time cost, and total loss functions, we can continue developing new dominance rules to be able to discard labels that cannot lead us to the optimal solution. To this end, in the next subsection we develop some propositions that will help us detecting those labels that can be fully dominated by others.

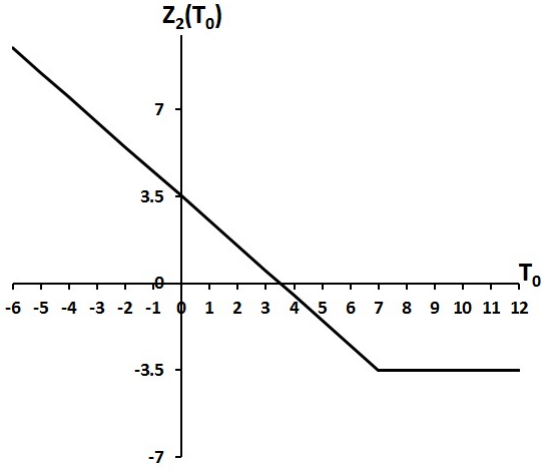

Figure 4: Z_2 as a function of T_0

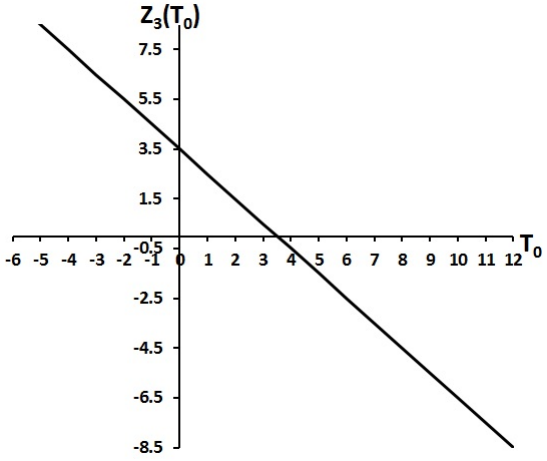

Figure 5: Z_3 as a function of T_0

3.1.2. Dominance Rules

As mentioned above, when there are two distinct partial paths ending at the same vertex, then the first one dominates the second one if the first label is componentwise less than or equal to the second label (with at least one strict inequality). So, when we identify a new path from V_0 to V_i, we have to compare this label with every other non-dominated label of V_i created earlier. Assume that $L_i^1 = (Z_i^1, T_i^1, H_i^1, Del_i^1, (El_k^1)_{k\in V}^i)$ is a non-dominated

label and $L_i^2 = (Z_i^2, T_i^2, H_i^2, Del_i^2, (El_k^2)_{k\in V}^i)$ a newly created label. Following the classical dominance rules, in order to check whether L_i^1 dominates L_i^2, we have to check whether $Z_i^1 \leq Z_i^2$, $T_i^1 \leq T_i^2$, $H_i^1 \leq H_i^2$, $Del_i^1 \leq Del_i^2$, $(El_k^1)^i \leq (El_k^2)^i$ for all $k \in V$. For our problem, we can check easily whether $Del_i^1 \leq Del_i^2$, $(El_k^1)^i \leq (El_k^2)^i$ for all $k \in V$, since the corresponding resources take numerical values. However, comparing the service start time, total spent time, and total loss of two paths is not so easy since they are functions of T_0. Moreover, if we find out that $Z_i^1(T_0) \leq Z_i^2(T_0)$, $T_i^1(T_0) \leq T_i^2(T_0)$, and $H_i^1(T_0) \leq H_i^2(T_0)$ for all T_0, we still cannot guarantee that $Z_{i+1}^1(T_0) \leq Z_{i+1}^2(T_0)$, $T_{i+1}^1(T_0) \leq T_{i+1}^2(T_0)$, and $H_{i+1}^1(T_0) \leq H_{i+1}^2(T_0)$ for all T_0 when both paths are extended to the same vertex V_{i+1}. Therefore, we have to determine conditions that ensure dominance among functions of T_0 when paths are further extended. To this end, we next establish some propositions that explicitly define such conditions.

Our first proposition related to the dominance rules is about satisfying Conditions I and II which appear in the definition of the service start time and time cost functions.

Proposition 2. *If $(V_0, V_1, \ldots, V_i)$ is a feasible partial path and V_k is any vertex satisfying Condition II on this path, then $\ell_j = \ell_k$ and Condition II holds for all vertices V_j with $k \leq j \leq i$.*

Proof. For a vertex V_j with $j \geq k+1$, we have $b_j - \sum_{l=k}^{j-1} \bar{t}_{l,l+1} \geq \tilde{a}_k$ by definition of $\tilde{a}_k$ and by feasibility of the partial path $(V_0, V_1, \ldots, V_j)$. Since vertex V_k satisfies Condition II, we also have

$$\tilde{a}_k \geq \ell_k + \sum_{l=0}^{k-1} \bar{t}_{l,l+1}. \tag{7}$$

Hence, $b_j - \sum_{l=k}^{j-1} \bar{t}_{l,l+1} \geq \ell_k + \sum_{l=0}^{k-1} \bar{t}_{l,l+1}$, or equivalently, $b_j - \sum_{l=0}^{j-1} \bar{t}_{l,l+1} \geq \ell_k$ for $j \geq k+1$. In view of Eq. (4), this implies that $\ell_j = \ell_k$, that is, ℓ_k does not change when additional vertices are visited on the given path.

Next we show that Condition II is satisfied for every vertex V_j with $j \geq k+1$. By Eq. (5), we have $\tilde{a}_j \geq \tilde{a}_k + \sum_{l=k}^{j-1} \bar{t}_{l,l+1}$. Together with inequality (7),

this yields $\tilde{a}_j \geq \ell_k + \sum_{l=0}^{k-1} \bar{t}_{l,l+1} + \sum_{l=k}^{j-1} \bar{t}_{l,l+1} = \ell_k + \sum_{l=0}^{j-1} \bar{t}_{l,l+1} = \ell_j + \sum_{l=0}^{j-1} \bar{t}_{l,l+1}$, meaning that Condition II is satisfied for $j \geq k+1$. □

The next two propositions is about comparing the latest feasible start times ℓ_i of two distinct partial paths ending at the same vertex V_i, since ℓ_i determines the domain of T_0: T_0 is restricted to lie in $(-\infty, \ell_i^1]$ for the first label and in $(-\infty, \ell_i^2]$ for the second label. In order to fully discard the second path, we need to have $\ell_i^2 \leq \ell_i^1$ and $\ell_{i+1}^2 \leq \ell_{i+1}^1$, where ℓ_{i+1}^1 (ℓ_{i+1}^2) shows the latest feasible start time of the first (second) path when it is extended to the vertex V_{i+1}. Otherwise the functions $T_{i+1}^2(T_0)$, $H_{i+1}^2(T_0)$, $Z_{i+1}^2(T_0)$ are defined for values in $(\ell_{i+1}^1, \ell_{i+1}^2]$ and this prevents us from getting rid of the second label. Proposition 4 states sufficient conditions for this to happen and Proposition 3 is only used to prove Proposition 4.

Proposition 3. *If there are two distinct feasible partial paths, say path 1 $(V_0, V_1^1, \ldots, V_{i-1}^1, V_i)$ and path 2 $(V_0, V_1^2, \ldots, V_{i-1}^2, V_i)$, starting from the depot and ending at the same vertex V_i such that the total traveling and service time $\theta_i^1 = \sum_{k=0}^{i-1} \bar{t}_{k^1,(k+1)^1}$ of the first path is less than or equal to the total traveling and service time $\theta_i^2 = \sum_{k=0}^{i-1} \bar{t}_{k^2,(k+1)^2}$ of the second path, and furthermore the latest feasible start time ℓ_i^1 for the first path is greater than or equal to the latest feasible start time ℓ_i^2 for the second path, then after extending both paths to the same vertex V_{i+1} the new latest feasible start time ℓ_{i+1}^1 for the first path will still be greater than or equal to the new latest feasible start time ℓ_{i+1}^2 for the second path. In symbols: if $\theta_i^1 \leq \theta_i^2$ and $\ell_i^1 \geq \ell_i^2$, then $\ell_{i+1}^1 \geq \ell_{i+1}^2$.*

Proof. After extending the paths to V_{i+1}, we have $\ell_{i+1}^j = \min\left\{\ell_i^j, b_{i+1} - \theta_i^j - \bar{t}_{i,i+1}\right\}$ for $j = 1, 2$. If $\ell_{i+1}^1 = \ell_i^1$, then $\ell_{i+1}^1 \geq \ell_{i+1}^2$ since $\ell_i^1 \geq \ell_i^2 \geq \ell_{i+1}^2$. If $\ell_{i+1}^1 = b_{i+1} - \theta_i^1 - \bar{t}_{i,i+1}$, then $\ell_{i+1}^1 \geq b_{i+1} - \theta_i^2 - \bar{t}_{i,i+1}$ since $\theta_i^1 \leq \theta_i^2$, and hence, $\ell_{i+1}^2 \leq b_{i+1} - \theta_i^2 - \bar{t}_{i,i+1} \leq \ell_{i+1}^1$. □

Proposition 4. *If there are two distinct feasible partial paths, say path 1 $(V_0, V_1^1, \ldots, V_{i-1}^1, V_i)$ and path 2 $(V_0, V_1^2, \ldots, V_{i-1}^2, V_i)$, starting from the depot and ending at the same vertex V_i such that $\ell_i^1 \geq \ell_i^2$ and the time cost of the first path is less than or equal to the time cost of the second path, i.e. $C_i^1(T_0) \leq C_i^2(T_0)$ for $T_0 \leq \ell_i^2$, then after extending both paths to the same vertex V_{i+1} the new latest feasible start time ℓ_{i+1}^1 for the first path will still be*

greater than or equal to the new latest feasible start time ℓ_{i+1}^2 for the second path, namely $\ell_{i+1}^1 \geq \ell_{i+1}^2$.

Proof. The minimum value of a time cost function is either $\beta\theta_i$ or $\beta\left(\tilde{a}_i - \ell_i\right)$ depending on whether Condition I or II is satisfied at vertex V_i (either two linear pieces or one linear piece, respectively). Let A be the minimum value of C_i^1, and let B be the minimum of C_i^2. Since $C_i^1(T_0) \leq C_i^2(T_0)$ for $T_0 \leq \ell_i^2$, we have $A \leq B$. We prove the proposition according to the values A and B can take.

Case 1: Assume that for both paths Condition I is satisfied at vertex V_i. Then, $A = \beta\theta_i^1 \leq B = \beta\theta_i^2$. Since $\theta_i^1 \leq \theta_i^2$ and $\ell_i^1 \geq \ell_i^2$, $\ell_{i+1}^1 \geq \ell_{i+1}^2$ by Proposition 3.

Case 2: Assume that at vertex V_i, Condition I is satisfied for path 1 and Condition II for path 2. Then, $A = \beta\theta_i^1$ and $B = \beta\left(\tilde{a}_i^2 - \ell_i^2\right)$. Since $A \leq B$, $\ell_i^2 \leq \tilde{a}_i^2 - \theta_i^1$. By Proposition 2, $\ell_{i+1}^2 = \ell_i^2$ and hence $\ell_{i+1}^2 \leq \tilde{a}_i^2 - \theta_i^1$.

Following Eq. (4) we know that ℓ_{i+1}^1 is equal to $\max\left\{\ell_i^1, b_{i+1} - (\theta_i^1 + \overline{t}_{i,i+1})\right\}$. If $\ell_{i+1}^1 = \ell_i^1$, then $\ell_{i+1}^1 \geq \ell_{i+1}^2$. Else, $\ell_{i+1}^1 = b_{i+1} - (\theta_i^1 + \overline{t}_{i,i+1})$. In such a case, since path 2 visits V_{i+1}, we need to have $\tilde{a}_i^2 + \overline{t}_{i,i+1} \leq b_{i+1}$. Thus, $\tilde{a}_i^2 + \overline{t}_{i,i+1} - (\theta_i^1 + \overline{t}_{i,i+1}) \leq b_{i+1} - (\theta_i^1 + \overline{t}_{i,i+1}) = \ell_{i+1}^1$ implying that $\tilde{a}_i^2 - \theta_i^1 \leq \ell_{i+1}^1$. As $\ell_{i+1}^2 \leq \tilde{a}_i^2 - \theta_i^1$, we obtain $\ell_{i+1}^1 \geq \ell_{i+1}^2$ so that the proposition holds.

Case 3: Assume that at vertex V_i, Condition II is satisfied for path 1 and Condition I for path 2. Since Condition II holds for path 1, $\tilde{a}_i^1 - \ell_i^1 \geq \theta_i^1$. Moreover, $\tilde{a}_i^1 - \ell_i^1 \leq \theta_i^2$ due to the fact that $A \leq B$ which translates into $\theta_i^1 \leq \theta_i^2$. Since $\theta_i^1 \leq \theta_i^2$ and $\ell_i^1 \geq \ell_i^2$, $\ell_{i+1}^1 \geq \ell_{i+1}^2$ by Proposition 3.

Case 4: Assume now that for both paths Condition II is satisfied at vertex V_i. Then, $\ell_{i+1}^1 = \ell_i^1$ and $\ell_{i+1}^2 = \ell_i^2$ by Proposition 2. Hence, $\ell_{i+1}^1 \geq \ell_{i+1}^2$ which completes the proof.

□

The following proposition determines the new minimum value of the time cost when a partial path ending at a vertex V_i is extended to include a new vertex V_{i+1}.

Proposition 5. *Assume that $(V_0, V_1, \ldots, V_i)$ is a feasible partial path, and let A be the minimum value of the time cost function $C_i(T_0)$. When a vertex V_{i+1} is added to the path, the minimum value of $C_{i+1}(T_0)$ is equal to $A' = \max\left\{A + \beta\overline{t}_{i,i+1}, \beta(\tilde{a}_{i+1} - \ell_{i+1})\right\}$.*

Proof. For simplicity of notations, let $\overline{t} = \overline{t}_{i,i+1}$. As given in the proof of Proposition 4, the minimum value of the time cost function $C_i(T_0)$ is either

$\beta\theta_i$ or $\beta\,(\tilde{a}_i - \ell_i)$ depending on whether Condition I or II is satisfied at vertex V_i.

Case 1: Assume that Condition I is satisfied at vertex V_{i+1}. Then, by Proposition 2, Condition I necessarily holds at vertex V_i as well, so that $A = \beta\theta_i$ and $A' = \beta\theta_{i+1} = \beta(\theta_i + \bar{t}) = A + \beta\bar{t}$. Condition I at vertex V_{i+1} translates into $\tilde{a}_{i+1} < \ell_{i+1} + \theta_{i+1} = \ell_{i+1} + \theta_i + \bar{t}$, which implies $\theta_i + \bar{t} > \tilde{a}_{i+1} - \ell_{i+1}$, and hence $A' = A + \beta\bar{t} = \beta(\theta_i + \bar{t}) > \beta(\tilde{a}_{i+1} - \ell_{i+1})$, so that the proposition holds in this case.

Case 2: Assume next that Condition II is satisfied at V_{i+1}, so that $A' = \beta(\tilde{a}_{i+1} - \ell_{i+1})$ and $\tilde{a}_{i+1} \geq \ell_{i+1} + \theta_{i+1}$. The latter inequality can be rewritten as $\theta_i + \bar{t} \leq \tilde{a}_{i+1} - \ell_{i+1}$, and hence $\beta(\theta_i + \bar{t}) \leq \beta(\tilde{a}_{i+1} - \ell_{i+1}) = A'$. Consider now two subcases:

Case 2a: If Condition I holds at V_i, then $A = \beta\theta_i$, hence we obtain $A + \beta\bar{t} \leq \beta(\tilde{a}_{i+1} - \ell_{i+1})$ and the proposition holds.

Case 2b: If Condition II holds at V_i, then $A = \beta(\tilde{a}_i - \ell_i)$. Moreover, $\ell_i = \ell_{i+1}$ by Proposition 2, and $\tilde{a}_i + \bar{t} \leq \tilde{a}_{i+1}$ by definition of $\tilde{a}_{i+1}$. Thus, we successively obtain $A + \beta\bar{t} = \beta(\tilde{a}_i - \ell_i + \bar{t}) \leq \beta(\tilde{a}_{i+1} - \ell_{i+1}) = A'$, which completes the proof. □

We are now ready to establish conditions under which dominance rules can be applied to the time cost functions of two distinct paths ending at the same vertex.

Proposition 6. *If there are two distinct feasible partial paths, say path 1* $(V_0, V_1^1, \ldots, V_{i-1}^1, V_i)$ *and path 2* $(V_0, V_1^2, \ldots, V_{i-1}^2, V_i)$*, starting from the depot and ending at the same vertex* V_i *such that* $\ell_i^1 \geq \ell_i^2$ *and the time cost of the first path is less than or equal to the time cost of the second path, i.e.,* $C_i^1(T_0) \leq C_i^2(T_0)$ *for* $T_0 \leq \ell_i^2$*, then the time cost of the first path remains less than or equal to the time cost of the second path when both paths are extended to a same vertex* V_{i+1}*, i.e.,* $C_{i+1}^1(T_0) \leq C_{i+1}^2(T_0)$ *for* $T_0 \leq \ell_{i+1}^2$*.*

Proof. The non-constant (decreasing) piece of the time cost function C_l^j is of the form $\beta(\tilde{a}_l^j - T_0)$ for all vertices l and for each path $j = 1, 2$. Let A be the minimum value of C_i^1, and let B be the minimum of C_i^2. Also, let $E = \beta\tilde{a}_i^1$ and $F = \beta\tilde{a}_i^2$ denote the intercept of C_i^1 and C_i^2, respectively, with the vertical axis $T_0 = 0$. Since $C_i^1(T_0) \leq C_i^2(T_0)$ for $T_0 \leq \ell_i^2$, we have $A \leq B$ and $E \leq F$. Let A', B', E', and F' be the corresponding values for C_{i+1}^1 and C_{i+1}^2 when both paths are extended to V_{i+1}. In order to establish the proposition, we only need to prove that $A' \leq B'$ and $E' \leq F'$.

Since $E' = \beta\tilde{a}^1_{i+1}$ and $F' = \beta\tilde{a}^2_{i+1}$, the inequality $E' \leq F'$ is equivalent to $\tilde{a}^1_{i+1} \leq \tilde{a}^2_{i+1}$. If $\tilde{a}^1_{i+1} = a_{i+1}$, then Eq. (5) implies $\tilde{a}^1_{i+1} \leq \tilde{a}^2_{i+1}$ as required. Else, if $\tilde{a}^1_{i+1} = \tilde{a}^1_i + \bar{t}$ (where $\bar{t} = \bar{t}_{i,i+1}$), then Eq. (5) implies $a_{i+1} \leq \tilde{a}^1_i + \bar{t}$. Since $E \leq F$, we have $\tilde{a}^1_i \leq \tilde{a}^2_i$. Hence, $a_{i+1} \leq \tilde{a}^2_i + \bar{t}$ and (again by Eq. (5)) $\tilde{a}^2_{i+1} = \tilde{a}^2_i + \bar{t} \geq \tilde{a}^1_i + \bar{t} = \tilde{a}^1_{i+1}$. So, we have proved that $E' \leq F'$.

Next we show that $A' \leq B'$. By Proposition 5, $A' = \max\left\{A + \beta\bar{t}, \beta(\tilde{a}^1_{i+1} - \ell^1_{i+1})\right\}$ and $B' = \max\left\{B + \beta\bar{t}, \beta(\tilde{a}^2_{i+1} - \ell^2_{i+1})\right\}$. Let us consider the possible values of A'. If $A' = A+\beta\bar{t}$, then, because $A \leq B$, we obtain $A' \leq B+\beta\bar{t} \leq B'$, as required. Else, $A' = \beta(\tilde{a}^1_{i+1}-\ell^1_{i+1})$. We have shown above that $\tilde{a}^1_{i+1} \leq \tilde{a}^2_{i+1}$ and, by Proposition 4, $\ell^1_{i+1} \geq \ell^2_{i+1}$. Hence, $A' = \beta(\tilde{a}^1_{i+1} - \ell^1_{i+1}) \leq \beta(\tilde{a}^2_{i+1} - \ell^2_{i+1}) \leq B'$. □

Proposition 6 describes conditions that lead to dominance among time cost functions. However, we need one more result to state the conditions needed for dominance among total loss functions of the form $Z_l(T_0) = C_l(T_0) + \alpha \sum_{k=0}^{l-1} c_{k,k+1} - \sum_{k=0}^{l-1} \eta_k$ (see Subsection 3.1.1).

Proposition 7. *If there are two distinct feasible partial paths, say path 1 $(V_0, V^1_1, \ldots, V^1_{i-1}, V_i)$ and path 2 $(V_0, V^2_1, \ldots, V^2_{i-1}, V_i)$, starting from the depot and ending at the same vertex V_i such that $\ell^1_i \geq \ell^2_i$, the time cost of the first path is less than or equal to the time cost of the second path (i.e., $C^1_i(T_0) \leq C^2_i(T_0)$ for $T_0 \leq \ell^2_i$), and furthermore $\alpha\left(\sum_{k=0}^{i-1} c^1_{k,k+1}\right) - \sum_{k=0}^{i-1} \eta^1_k \leq \alpha\left(\sum_{k=0}^{i-1} c^2_{k,k+1}\right) - \sum_{k=0}^{i-1} \eta^2_k$, then the total loss of the first path remains less than or equal to the total loss of the second path for $T_0 \leq \ell^2_{i+1}$ when both paths are extended to a same vertex V_{i+1}, i.e., $Z^1_{i+1}(T_0) \leq Z^2_{i+1}(T_0)$ for $T_0 \leq \ell^2_{i+1}$.*

Proof. When a new vertex V_{i+1} is added to both paths, the total loss function of path j up to V_{i+1} is $Z^j_{i+1}(T_0) = C^j_{i+1}(T_0) + \alpha\left(\sum_{k=0}^{i-1} c^j_{k,k+1}\right) + \alpha c_{i,i+1} - \sum_{k=0}^{i-1} \eta^j_k - \eta_i$ for $j = 1, 2$. As a consequence of Proposition 6, we have $C^1_{i+1}(T_0) \leq C^2_{i+1}(T_0)$ for $T_0 \leq \ell^2_{i+1}$. In view of the hypotheses of the proposition, we immediately derive $Z^1_{i+1}(T_0) \leq Z^2_{i+1}(T_0)$ for $T_0 \leq \ell^2_{i+1}$. □

With Proposition 7, we obtain all the requirements needed to compare the total loss functions of two distinct partial paths ending at the same vertex. Namely, we have to compare their latest feasible start times from the depot,

time cost functions, and their constant terms composed of the total distance costs and collected revenues. If the conditions stated in Proposition 7 are satisfied, we can conclude that one total loss function dominates the other one.

With the following proposition, we give a result which will allow us to compare the time and the total spent time resource functions along distinct paths.

Proposition 8. *If there are two distinct feasible partial paths, say path 1* $(V_0, V_1^1, \ldots, V_{i-1}^1, V_i)$ *and path 2* $(V_0, V_1^2, \ldots, V_{i-1}^2, V_i)$*, starting from the depot and ending at the same vertex such that* $\ell_i^1 \geq \ell_i^2$ *and* $C_i^1(T_0) \leq C_i^2(T_0)$ *for* $T_0 \leq \ell_i^2$*, then the service start time and the spent time at* V_i *along the first path is less than or equal to the service start time and the spent time at* V_i *along the second path, i.e.,* $T_i^1(T_0) \leq T_i^2(T_0)$ *and* $H_i^1(T_0) \leq H_i^2(T_0)$ *for* $T_0 \leq \ell_i^2$*.*

Proof. By definition, $C_i^1(T_0) = \beta(T_i^1(T_0) - T_0)$ and $C_i^2(T_0) = \beta(T_i^2(T_0) - T_0)$. The conclusion immediately follows. □

Proposition 8 lets us paraphrase the dominance rules which apply to our problem (compare with the introduction of Section 3.1.2). Propositions 7 and 8 advise to compare the time cost functions, the latest feasible start times, and the sum of the distance costs minus the collected revenues of two distinct paths in order to compare their total loss, service start time, and total spent time functions. Therefore, it may be necessary to change the representation of labels in order to adapt the resource extension functions and to make the comparison of the labels easier. For this purpose, given a partial path starting from the source and ending at a vertex V_i, we first let the sum of the distance costs minus the collected revenues $\alpha \left(\sum_{k=0}^{i-1} c_{k,k+1} \right) - \sum_{k=0}^{i-1} \eta_k$ denote by δ_i, the minimum value of the time cost function by A_i, and its C_i-intercept by E_i. Then, a label at a vertex V_i can be represented by $L_i = (\delta_i, -\ell_i, E_i, A_i, Del_i, (El_k)_{k \in V}^i)$. If there are two labels $L_i^1 = (\delta_i^1, -\ell_i^1, E_i^1, A_i^1, Del_i^1, (El_k^1)_{k \in V}^i)$ and $L_i^2 = (\delta_i^2, -\ell_i^2, E_i^2, A_i^2, Del_i^2, (El_k^2)_{k \in V}^i)$ representing two partial paths from source 0 to vertex V_i, then L_i^1 dominates L_i^2 if and only if $\delta_i^1 \leq \delta_i^2$, $-\ell_i^1 \leq -\ell_i^2$, $E_i^1 \leq E_i^2$, $A_i^1 \leq A_i^2$, $Del_i^1 \leq Del_i^2$, $(El_k^1)^i \leq (El_k^2)^i$ for all $k \in V$, and $L_i^1 \neq L_i^2$. Furthermore, when a label $L_i = (\delta_i, -\ell_i, E_i, A_i, Del_i, (El_k)_{k \in V}^i)$ is extended from vertex V_i to a successor vertex V_{i+1} to obtain a new label $L_{i+1} = (\delta_{i+1}, -\ell_{i+1}, E_{i+1}, A_{i+1}, Del_{i+1}, (El_k)_{k \in V}^{i+1})$, the components of the new label L_{i+1} can be computed as follows:

$$
\begin{aligned}
&\delta_{i+1} = \delta_i + \alpha c_{i,i+1} - \eta_i \\
&-\ell_{i+1} = -\min\{\ell_i, b_{i+1} - \theta_{i+1}\} \\
&E_{i+1} = \beta \tilde{a}_{i+1} \\
&A_{i+1} = \max\left\{A_i + \beta \bar{t}_{i,i+1}, \beta(\tilde{a}_{i+1} - \ell_{i+1})\right\} \\
&Del_{i+1} = Del_i + d_{i+1} \\
&El_k^{i+1} = \begin{cases} El_k^i + 1, & \text{if } k = i+1 \\ El_k^i, & \text{if } k \neq i+1 \end{cases} \qquad \text{for all } k \in V
\end{aligned}
$$

where θ_{i+1} is updated as $\theta_i + \bar{t}_{i,i+1}$ and $\tilde{a}_{i+1}$ as $\max\left\{a_{i+1}, \tilde{a}_i + \bar{t}_{i,i+1}\right\}$.

As a summary, we need to make $5 + n$ comparisons to eliminate the partial paths which do not lead to optimality. Therefore, if we can reduce the number of comparisons even by 1, the DP algorithm will be accelerated. In order to do that, we introduce the following proposition.

Proposition 9. *If there are two distinct feasible partial paths, say path 1* $(V_0, V_1^1, \ldots, V_{i-1}^1, V_i)$ *and path 2* $(V_0, V_1^2, \ldots, V_{i-1}^2, V_i)$*, starting from the depot and ending at the same vertex, then label* L^1 *dominates* L^2 *if* $\delta_i^1 \leq \delta_i^2 + \beta \min\{(\ell_i^1 - \ell_i^2), 0\}$*,* $E_i^1 \leq E_i^2$*,* $A_i^1 \leq A_i^2$*,* $Del_i^1 \leq Del_i^2$*,* $(El_k^1)^i \leq (El_k^2)^i$ *for all* $k \in V$*, and* $L_i^1 \neq L_i^2$

Proof. If $\ell_i^1 \geq \ell_i^2$, then $\delta_i^1 \leq \delta_i^2 + \beta \min\{(\ell_i^1 - \ell_i^2), 0\}$ translates into $\delta_i^1 \leq \delta_i^2$. Then, we have $\delta_i^1 \leq \delta_i^2$, $-\ell_i^1 \leq -\ell_i^2$, $E_i^1 \leq E_i^2$, $A_i^1 \leq A_i^2$, $Del_i^1 \leq Del_i^2$, $(El_k^1)^i \leq (El_k^2)^i$ for all $k \in V$ which brings us back to the original dominance rules obtained by Proposition 8. Because of Propositions 4 and 6, we know that $\delta_{i+1}^1 \leq \delta_{i+1}^2$, $-\ell_{i+1}^1 \leq \ell_{i+1}^2$, $E_{i+1}^1 \leq E_{i+1}^2$, $A_{i+1}^1 \leq A_{i+1}^2$ when both paths are extended to the same vertex V_{i+1}. Furthermore, the inequality $\delta_{i+1}^1 \leq \delta_{i+1}^2 + \beta \min\left\{(\ell_{i+1}^1 - \ell_{i+1}^2), 0\right\}$ will hold, since $-\ell_{i+1}^1 \leq \ell_{i+1}^2$ and $\delta_{i+1}^1 = \delta_i^1 + \alpha c_{i,i+1} - \eta_{i+1}$, $\delta_{i+1}^2 = \delta_i^2 + \alpha c_{i,i+1} - \eta_{i+1}$.

Now assume that $\ell_i^1 < \ell_i^2$. Then we have $\delta_i^1 \leq \delta_i^2 + \beta(\ell_i^1 - \ell_i^2)$ and furthermore $\delta_i^1 \leq \delta_i^2$ since $(\ell_i^1 - \ell_i^2)$ is a negative number. When both paths are extended to the same vertex V_{i+1}, we know that we are going to have $\tilde{a}_{i+1}^1 \leq \tilde{a}_{i+1}^2$ due to the proof of Proposition 6. However, in order to show that the minimum of the total cost of path 1 is less than or equal to the minimum of the total cost of path 2, we need to have $A_{i+1}^1 + \delta_{i+1}^1 \leq A_{i+1}^2 + \delta_{i+1}^2$. For simplicity of notations, let $\bar{t} = \bar{t}_{i,i+1}$ and $\gamma = \alpha c_{i,i+1} - \eta_{i+1}$. As given

by Proposition 5, the minimum value of the time cost function $C_{i+1}(T_0)$ is equal to $A' = \max\{A + \beta\bar{t}, \beta(\tilde{a}_{i+1} - \ell_{i+1})\}$. Thus, we want to show that $A^1_{i+1} + \delta^1_{i+1} \leq A^2_{i+1} + \delta^2_{i+1}$ according to the values of A^1_{i+1} and A^2_{i+1}.

Case 1: Assume that the minimum of the total cost function of path 1 is $A^1_{i+1} = A^1_i + \beta\bar{t} + \delta^1_i + \gamma$ and the minimum of the total cost function of path 2 is $A^2_{i+1} = A^2_i + \beta\bar{t} + \delta^2_i + \gamma$. Since $A^1_i \leq A^2_i$ and $\delta^1_i \leq \delta^2_i$, we have $A^1_i + \delta^1_i \leq A^2_i + \delta^2_i$ so that the proposition holds in this case.

Case 2: Assume that the minimum of the total cost function of path 1 is $A^1_{i+1} = A^1_i + \beta\bar{t} + \delta^1_i + \gamma$ and the minimum of the total cost function of path 2 is $A^2_{i+1} = \beta(\tilde{a}^2_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma$. We have $\beta(\tilde{a}^2_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma \geq A^2_i + \beta\bar{t} + \delta^2_i + \gamma$ according to Proposition 5. $A^2_i + \beta\bar{t} + \delta^2_i + \gamma \geq A^1_i + \beta\bar{t} + \delta^1_i + \gamma$ since $A^1_i \leq A^2_i$ and $\delta^1_i \leq \delta^2_i$ so that the proposition holds in this case.

Case 3: Assume that the minimum of the total cost function of path 1 is $A^1_{i+1} = \beta(\tilde{a}^1_{i+1} - \ell^1_{i+1}) + \delta^1_i + \gamma$ and the minimum of the total cost function of path 2 is $A^2_{i+1} = A^2_i + \beta\bar{t} + \delta^2_i + \gamma$. $A^2_i + \beta\bar{t} + \delta^2_i + \gamma \geq \beta(\tilde{a}^2_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma$ according to Proposition 5. $\beta(\tilde{a}^2_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma \geq \beta(\tilde{a}^1_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma$ since $\tilde{a}^1_{i+1} \leq \tilde{a}^2_{i+1}$. $\beta(\tilde{a}^1_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma \geq \beta(\tilde{a}^1_{i+1} - \ell^2_i) + \delta^2_i + \gamma$ since $\ell^2_i \geq \ell^2_{i+1}$ by definition of ℓ_i. Since $\delta^1_i \leq \delta^2_i + \beta(\ell^1_i - \ell^2_i)$, we have $\delta^2_i \geq \delta^1_i - \beta(\ell^1_i - \ell^2_i)$, thus $\beta(\tilde{a}^1_{i+1} - \ell^2_i) + \delta^2_i + \gamma \geq \beta(\tilde{a}^1_{i+1} - \ell^2_i) + \delta^1_i - \beta(\ell^1_i - \ell^2_i) + \gamma = \beta(\tilde{a}^1_{i+1} - \ell^1_{i+1}) + \delta^1_i + \gamma$ because $\ell^1_{i+1} = \ell^1_i$ due to Proposition 2.

Case 4: Assume that the minimum of the total cost function of path 1 is $A^1_{i+1} = \beta(\tilde{a}^1_{i+1} - \ell^1_{i+1}) + \delta^1_i + \gamma$ and the minimum of the total cost function of path 2 is $A^2_{i+1} = \beta(\tilde{a}^2_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma$. $\beta(\tilde{a}^2_{i+1} - \ell^2_{i+1}) + \delta^2_i + \gamma \geq \beta(\tilde{a}^1_{i+1} - \ell^2_i) + \delta^2_i + \gamma$ since $\tilde{a}^2_{i+1} \geq \tilde{a}^1_{i+1}$ and $\ell^1_{i+1} = \ell^1_i$. $\beta(\tilde{a}^1_{i+1} - \ell^2_i) + \delta^2_i + \gamma \geq \beta(\tilde{a}^1_{i+1} - \ell^2_i) + \delta^1_i - \beta(\ell^1_i - \ell^2_i) + \gamma$ since $\delta^2_i \geq \delta^1_i - \beta(\ell^1_i - \ell^2_i)$. Then $\beta(\tilde{a}^1_{i+1} - \ell^2_i) + \delta^1_i - \beta(\ell^1_i - \ell^2_i) + \gamma = \beta(\tilde{a}^1_{i+1} - \ell^1_{i+1}) + \delta^1_i + \gamma$ since $\ell^1_{i+1} = \ell^1_i$.

So, for each possible case we show that the minimum of the total cost function of path 1 is less than or equal to the minimum of the total cost function of path 2 at the next extension. In order to apply the new dominance rules for the rest of the extensions, we should prove that the difference $\beta(\ell^2_{i+k} - \ell^1_{i+k})$, where $k \geq 1$, does not decrease below the difference $(\delta^2_{i+k} - \delta^1_{i+k})$. In Case 1 and 2 we do not need to have it, because they hold for any value of $(\ell^2_i - \ell^1_i)$. Case 3 and 4 are the cases where Condition II holds for path 1. Once Condition II holds for path 1, it holds for all vertices V_{i+k} according to Proposition 2, where $k \geq 1$. This means that $\ell^1_{i+k} = \ell^1_i$ due to Proposition 2 and $\ell^2_{i+1} \leq \ell^2_i$ by definition of ℓ_i. Hence, $(\ell^2_{i+k} - \ell^1_{i+k}) \leq (\ell^2_i - \ell^1_i)$ for $k \geq 1$.

As a last point, we have to discuss whether the relationship between $A^1_{i+k} \leq A^2_{i+k}$ should continue to hold for $k \geq 1$. Now suppose that both

paths are already extended to V_{i+1}. If at the end of this first extension either Case 1 or Case 2 holds, $A^1_{i+1} \leq A^2_{i+1}$ due to the proof of Proposition 6. Moreover, if Case 1 or Case 2 continue to hold for further extensions to V_{i+k}, where $k > 1$, then again due to the proof of Proposition 6 we will have the inequality $A^1_{i+k} \leq A^2_{i+k}$. Now assume that either Case 3 or Case 4 hold at the first extension to V_{i+1}. Case 3 and Case 4 are cases where Condition II holds for path 1 which automatically implies that Condition II will continue to be satisfied and thus either Case 3 or Case 4 will hold for further extensions to V_{i+k} with $k > 1$. As can be observed in the proof of Case 3 and Case 4, the minimum of the total cost function of path 1 will be less than or equal to to the minimum of the total cost function of path 2 independently from the relationship between A^1_{i+k} and A^2_{i+k} for $k > 1$. □

With Proposition 9, the number of comparisons which are needed to be realized at every extension of a partial path decreases to $4 + n$. As a result, the dominance rules obtained by the previous propositions are strengthened.

In order to accelerate the DP algorithm, we additionally make use of the *improved definition of labels* proposed by Feillet et al. (2004). In view of their remarks, it is viable to determine vertices which cannot be visited in any feasible extension of a given partial path, because any extension would result in exceeding a resource limit. Such vertices are called *unreachable* and the value of their elementary resource is set to 1 although they have not been visited. This definition of labels increases the number of dominated labels and thus makes the algorithm more efficient with respect to the computational time. Since we assume that the travel times are non-negative and satisfy the triangle inequality, we can easily apply the technique of unreachable vertices.

3.2. A Bounded Bi-directional Dynamic Programming Algorithm

In the previous subsection, we have described a mono-directional DP algorithm, where all paths are extended in forward direction. Within the framework of this algorithm, every label of a vertex i is extended to every possible successor of i, which can generate an exponential number of new labels owing to the number of arcs in the network. In order to accelerate the solution procedure, we use the BBDS proposed by Righini and Salani (2006), where labels are extended *backward* from sink $n+1$ to the rest of the vertices in addition to the forward extensions from source 0. Therefore, before going into the details of our BBDS, we first give the DP algorithm based on the backward extensions.

3.2.1. Backward Dynamic Programming Algorithm

The classical backward directional version of the algorithm also assigns labels to every vertex j and is denoted by $L_j^b = (Z_j^b, T_j^b, H_j^b, Del_j^b, (El_k)_{k\in V}^{jb})$, where Z_j^b is the loss of backward partial path ending at j, and T_j^b, H_j^b, Del_j^b, $(El_k)_{k\in V}^{jb}$ represent the service start time, total spent time, delivered demand, and elementarity resources, respectively. The following resource extensions are utilized, when a label L_j^b of vertex j is extended to a predecessor vertex i:

$$
\begin{aligned}
&Z_i^b = Z_j^b + \alpha c_{ij} + \beta(T_j^b - T_i^b) - \eta_j \\
&T_i^b = \min\left\{b_i, T_j^b - \bar{t}_{ij}\right\} \\
&H_i^b = T_{n-1} - T_i^b \\
&Del_i^b = Del_j^b + d_i \\
&El_k^{ib} = \begin{cases} El_k^{jb} + 1, & \text{if } k = i \\ El_k^{jb}, & \text{if } k \neq i \end{cases} \qquad \text{for all } k \in V
\end{aligned}
$$

A label L_i^b is feasible only if $T_i^b \geq a_i$, $H_i^b \leq S$, $Del_i^b \leq Q$, and $El_k^{ib} \leq 1$ for all $k \in V$. If T_i^b turns out to be less than a_i or if at least one of the other components exceeds its corresponding upper bound, the label is eliminated.

In the backward-directional search process, the service start time T_{n+1} at sink $n+1$ is a decision variable that lies in the interval $(-\infty, \infty)$. Thus, we need to adapt the definitions of the time resource T_j^b and the total loss Z_j^b for any vertex j, since obviously they are functions of T_{n+1} and should be denoted by $T_j^b(T_{n+1})$ and $Z_j^b(T_{n+1})$, respectively. For this purpose, assume that we are given a backward partial path starting from sink $n+1$ and visiting i additional vertices. Let V_j^b be the vertex at the $(j+1)$st position of the partial path. Then, a backward partial path starting from V_0^b and ending at a vertex V_i^b is given as $(V_0^b, V_1^b, ..., V_{i-1}^b, V_i^b)$, where V_0^b represents the sink $n+1$. We define $\bar{t}_{j,j-1}$ as the sum of the travel time $t_{V_j^b, V_{j-1}^b}$ from V_j^b to V_{j-1}^b and the service time $s_{V_j^b}$ of V_j^b, i.e. $\bar{t}_{j,j-1} = t_{V_j^b, V_{j-1}^b} + s_{V_j^b}$. Let θ_j be equal to $\sum_{k=1}^{j} \bar{t}_{k,k-1}$. For simplicity, we denote the start of the service time $T_{V_j^b}^b(T_{n+1})$ at V_j^b as $T_j^b(T_{n+1})$ in the sequel. When the vehicle starts serving V_0^b at T_{n+1},

then the service start time $T_1^b(T_{n+1})$ at V_1^b is computed as

$$T_1^b(T_{n+1}) = \min\left\{b_1, T_{n+1} - \bar{t}_{10}\right\}. \tag{8}$$

Hence, if the vehicle reaches V_i^b following the partial path $(V_0^b, V_1^b, ..., V_{i-1}^b, V_i^b)$, then we obtain Eq. 9 by applying the time resource extension rule repeatedly starting with Eq. (8).

$$T_i^b(T_{n+1}) = \min\left\{b_i, T_{i-1}^b(T_{n+1}) - \bar{t}_{i,i-1}\right\} \tag{9}$$

Let e_i be the earliest feasible arrival time at sink V_0^b, which is defined as $e_i = max_{1\leq j\leq i}\{a_j + \theta_j\}$. The latest feasible service start time $\tilde{b}_i$ at vertex V_i^b is expressed as $\tilde{b}_i = \min\left\{b_i, \tilde{b}_{i-1} - \bar{t}_{i,i-1}\right\}$ for $i > 0$ and $\tilde{b}_0 = \infty$. Thus, the vehicle cannot start serving V_i^b after $\tilde{b}_i$ on a feasible backward partial path $(V_0^b, V_1^b, ..., V_{i-1}^b, V_i^b)$. Then, the service start time at vertex V_i^b can be defined as a function $T_i^b(T_{n+1})$ and is given as follows:

- Condition I: If $e_i - \theta_i < \tilde{b}_i$

$$T_i^b(T_{n+1}) = \begin{cases} T_{n+1} - \theta_i, & \text{if } e_i \leq T_{n+1} \leq \tilde{b}_i + \theta_i \\ \tilde{b}_i, & \text{if } T_{n+1} \geq \tilde{b}_i + \theta_i \end{cases}$$

- Condition II: Else if $e_i - \theta_i \geq \tilde{b}_i$

$$T_i^b(T_{n+1}) = \tilde{b}_i \text{ for } T_{n+1} \geq e_i$$

However, it is easy to show that the forward time resource extensions given in Subsection 3.1.1 can be utilized in order to define backward time resource extensions. Now let M be a constant. Then, following Eq. (8), $M - T_1^b(T_{n+1})$ will be equal to

$$M - T_1^b(T_{n+1}) = \max\left\{M - b_1, (M - T_{n+1}) + \bar{t}_{10}\right\}. \tag{10}$$

Similarly, $M - T_i^b(T_{n+1})$ can be rewritten as

$$M - T_i^b(T_{n+1}) = \max\left\{M - b_i, (M - T_{i-1}^b(T_{n+1})) + \bar{t}_{i,i-1}\right\}. \tag{11}$$

Thus, when we reverse the time windows of every visited vertex on the partial path $(V_0^b, V_1^b, ..., V_{i-1}^b, V_i^b)$ as $[M - b_k, M - a_k]$ for $0 < k \leq i$ and

apply the forward extensions, we obtain Eq. (10) at the first extension and Eq. (11) at the final extension. Since the vehicle moves back symmetrically from the sink to the source, all we need to do is to reverse the time windows of the vertices on a backward path by subtracting them from a constant M and implement the forward time resource extensions. In the forward DP constructed in this way, the service start time T_0^b from the source corresponds to $M - T_{n+1}$ and the service start time function at vertex i is given by $T_i(T_0^b) = M - T_i^b(T_{n+1})$. Since $T_i(T_0^b) \leq M - a_i$ is required for the feasibility of a forward partial path, we automatically have $T_i^b(T_{n+1}) \geq a_i$ for any label of vertex i.

The earliest feasible arrival time e_i at sink V_0^b can be obtained by $M - \ell_i^b$, where $\ell_i^b = \min_{1 \leq j \leq i} \{(M - a_j) - \theta_j\}$ is the latest feasible service start time at the source in the equivalent forward search. Similarly, the earliest service start time $\tilde{a}_i^b$ at vertex V_i^b helps us to find the latest feasible service start time $\tilde{b}_i$ at V_i^b by subtracting it from M, i.e. $\tilde{b}_i = M - \tilde{a}_i^b$. It is clear that the backward DP can be solved efficiently using the constructed froward search process together with the dominance rules given in Subsection 3.1.2

3.2.2. Concatenation of Partial Paths

As suggested by Righini and Salani (2006), in a bounded bi-directional algorithm a critical resource is selected in order to restrict the number of forward and backward labels generated. For our problem, we choose the time resource as the critical resource as it seems to be the most crucial one. Therefore, paths are extended forward from source 0 without generating new paths on which the amount of time that the vehicle spends is greater than $S/2$. The amount of time that the vehicle spends on a forward path ending at a vertex i is given by $T_i(T_0) - T_0$. We decide to choose T_0^* for which the total loss is minimized on that path. Hence, we continue to extend the paths forward from 0 as long as $(T_i(T_0^*) - T_0^*) \leq S/2$. Similarly, paths are extended backward from $n+1$ as long as $(T_{n+1}^* - T_i^b(T_{n+1}^*)) \leq S/2$, where T_{n+1}^* is the minimizer of the total loss function $Z_i^b(T_{n+1})$.

After realizing all possible forward and backward extensions, we need to concatenate the partial paths to obtain complete paths starting from source 0, ending at sink $n+1$ and remove the infeasible ones. Let $L_i = (\delta_i, -\ell_i, E_i, A_i, Del_i, (El_k)_{k \in V}^i)$ and $L_i^b = (\delta_i^b, -\ell_i^b, E_i^b, A_i^b, Del_i^b, (El_k)_{k \in V}^{ib})$ be a forward and backward label associated to feasible partial paths $(V_0, ..., V_i)$ and $(V_i, ..., V_{n+1})$ associated to a vertex V_i, respectively. The feasibility of the complete path $(V_0, ..., V_i, ..., V_{n+1})$ can be achieved if and only if the con-

catenation is feasible in terms of time, capacity, and elementarity resources. It is feasible concerning the capacity if $Del_i + Del_i^b - d_i \leq Q$, while it is feasible with regard to the elementarity if $El_k^i + El_k^{ib} \leq 1$ for all $k \in V - \{V_i\}$. Nevertheless, the feasibility of the concatenated path in terms of the time resource is established by the following proposition.

Proposition 10. *Let $(V_0, ..., V_i)$ and $(V_i, ..., V_{n+1})$ be a feasible forward and feasible backward partial path associated with a vertex V_i, respectively. The concatenation of these two partial paths is feasible in terms of time if and only if the earliest service start time $\tilde{a}_i$ at V_i is less than or equal to the latest feasible service start time $\tilde{b}_i$ at V_i, i.e. $\tilde{a}_i \leq \tilde{b}_i$.*

Proof. Assume that $\tilde{a}_i \leq \tilde{b}_i$. The concatenated path is feasible if for every $j \leq i$ and for every $l \geq i$, $\tilde{a}_j + \sum_{k=j}^{i-1} \bar{t}_{k,k+1} + \sum_{k=i}^{l-1} \bar{t}_{k,k+1} \leq b_l$ and similarly $\tilde{b}_l - \sum_{k=i}^{l-1} \bar{t}_{k,k+1} - \sum_{k=j}^{i-1} \bar{t}_{k,k+1} \geq a_j$. For simplicity, let $\sum_{k=j}^{i-1} \bar{t}_{k,k+1}$ be θ_i^j and $\sum_{k=i}^{l-1} \bar{t}_{k,k+1}$ be θ_i^l. Since the forward partial path is feasible, we have $\tilde{a}_j + \theta_i^j \leq \tilde{a}_i$ by Eq. (5). So, $\tilde{a}_j + \theta_i^j \leq \tilde{a}_i \leq \tilde{b}_i \leq \tilde{b}_l - \theta_i^l \leq b_l - \theta_i^l$ since $\tilde{b}_i = \min\left\{b_i, \tilde{b}_{i+1} - \bar{t}_{i,i+1}\right\}$. Thus, $\tilde{a}_j + \theta_i^j + \theta_i^l \leq b_l$. Since $a_j \leq \tilde{a}_j$, we consequently obtain $\tilde{b}_l - \theta_i^l - \theta_i^j \geq a_j$ following the same reasoning. Hence, the concatenated path is feasible.

Now assume that $\tilde{a}_i > \tilde{b}_i$. Since the partial paths are feasible, $\tilde{a}_i \leq b_i$ and $\tilde{b}_i \geq a_i$. Thus, $\tilde{a}_i > \tilde{b}_i$ if and only if there exist a vertex V_j and a vertex V_l with $j < i$ and $l > i$ for which $\tilde{a}_i = a_j + \theta_i^j$ and $\tilde{b}_i = b_l - \theta_i^l$. Then, $\tilde{a}_i > \tilde{b}_i$ implies that $a_j + \theta_i^j > b_l - \theta_i^l$, which is equivalent to $a_j + \theta_i^j + \theta_i^l > b_l$. Thus, the concatenated path is infeasible. □

In light of Proposition 10, we can simply summarize the conditions for concatenation of two partial paths. Two labels $L_i = (\delta_i, -\ell_i, E_i, A_i, Del_i, (El_k)_{k\in V}^i)$ and $L_i^b = (\delta_i^b, -\ell_i^b, E_i^b, A_i^b, Del_i^b, (El_k^b)_{k\in V}^i)$ can be linked if the following holds:

$$\begin{aligned} &\tilde{a}_i \leq \tilde{b}_i \\ &Del_i + Del_i^b - d_i \leq Q \\ &El_k^i + El_k^{ib} \leq 1 \qquad \text{for all } k \in V - \{V_i\} \end{aligned}$$

In the traditional ESPPRC, the objective is to minimize the total distance traveled. Therefore, once two partial paths are concatenated, the calculation

of the transportation cost is straightforward, since it only requires the summation of the total distance traveled on two partial paths. However, the transportation cost of the problem at hand depends on both the total distance traveled and the total amount of time that the vehicle spends. The main drawback is that the waiting time that will occur on a complete path cannot be calculated a priori. If there are one forward and one backward partial paths $(V_0, ..., V_i)$ and $(V_i, ..., V_{n+1})$ with waiting times w_1 and w_2, respectively, the total waiting time on the complete path $(0, ...V_i, ..., V_{n+1})$ is not necessarily equal to $w_1 + w_2$. To overcome this difficulty, we make use of the concatenation theorem and the computation of total waiting time on a concatenated path of Savelsbergh (1992) given for the VRPTW with minimization of route duration as the objective. This enables us to concatenate two partial paths and to compute the resulting total loss in constant time.

The concatenation theorem of Savelsbergh (1992) introduces the notion of a *forward time slack* which indicates how much the service start time of a visited vertex can be shifted forward without making the path infeasible with regard to the time resource. The theorem states that if two feasible partial paths $(V_0, ..., V_i)$ and $(V_{i+1}, ..., V_{n+1})$ with forward time slacks F_1 and F_2 for the first vertices are concatenated, the start time T_0 from the depot on the concatenated path can be shifted forward in time by

$$F = \min \left\{ F_1, F_2 + w_1 + T^b_{i+1} - (T_i + \bar{t}_{i,i+1}) \right\} \tag{12}$$

where w_1 is the total waiting time occurred on $(V_0, ..., V_i)$. Then the start time from the depot minimizing the duration on the concatenated path $(V_0, ..., V_i, ..., V_{n+1})$ is given by $T_0 + F$ which coincides with ℓ_{n+1} in our problem. In order to compute the waiting time on the concatenated path assuming that the vehicle starts at a certain T_0 from the depot, Savelsbergh (1992) defines the change λ in the service start time at V_{i+1} as $T_i + \bar{t}_{i,i+1} - T^b_{i+1}$ and introduces the *backward time slack*. The backward time slack of a path indicates how much the service start time of a visited vertex can be shifted backward without creating additional waiting time on that path. Then four different cases are determined according to the sign of λ and to whether the total waiting time w_2 on $(V_{i+1}, ..., V_{n+1})$ is positive or zero. The results given by Savelsbergh (1992) can be outlined as follows, where B_2 is the backward time slack of V_{i+1} on the path $(V_{i+1}, ..., V_{n+1})$:

In order to calculate w_1, w_2, λ, F, and B_2, we assume that $T_i = \tilde{a}_i$ and $T^b_{i+1} = \tilde{b}_{i+1}$. Hence, we find the total waiting time w that will occur on the

Table 1: The total waiting time on a concatenated path

	$\lambda \geq 0$	$\lambda < 0$
$w_2 = 0$	w_1	$w_1 + \max\{0, -\lambda - B_2\}$
$w_2 > 0$	$w_1 + \max\{0, w_2 - \lambda\}$	$w_1 + w_2 - \lambda$

concatenated path $(V_0, ..., V_i, ..., V_{n+1})$ with the help of Table 1 assuming that the vehicle starts from the depot at $T_0 = \tilde{a}_i - \theta_i$ if the time function consists of two pieces and at $T_0 = \ell_i$ otherwise. It is also important to emphasize that for our problem λ can take at most the value 0, since $T_i + \bar{t}_{i,i+1} - T^b_{i+1} = \tilde{a}_i + \bar{t}_{i,i+1} - \tilde{b}_{i+1} \leq 0$ because $\tilde{a}_i \leq \tilde{b}_i \leq \tilde{b}_{i+1} - \bar{t}_{i,i+1}$. If the forward time slack F calculated by Eq. (12) is greater than or equal to w, then the real total waiting time on $(V_0, ..., V_i, ..., V_{n+1})$ is equal to 0, because the start time from the depot is $T_0 + F$. Otherwise, it is equal to $w - F$.

Having defined how we find the minimum total waiting time that will occur on a concatenated path, we can easily find its minimum total loss and optimal start time from the depot. Since we already know the waiting times occurred on the partial paths $(V_0, ..., V_i)$ and $(V_{i+1}, ..., V_{n+1})$, we can now add the missing waiting times multiplied by β to their minimum total losses. Moreover, the optimal start time from the depot is given by the interval $[T_0 + \min\{F, w\}, T_0 + F]$.

3.3. A Simple Heuristic

Beside the exact mono-directional and bounded bi-directional DP algorithms presented above, we additionally developed a mono-directional DP heuristic. The heuristic assumes that the vehicle should start serving its first customer as early as possible. So, it considers in turn each customer $i \in D$ and commences with a partial path P_i starting from the depot and visiting customer i first. The start time from the depot is then set equal to $a_i - \bar{t}_{0i}$. Since the start time from the depot is fixed under these assumptions, the problem reduces now to a classical ESPPRC, where the arcs costs include an additional time cost component that is constant for a given path. So, each initial path P_i can be extended by implementing feasible resource extensions to obtain complete paths, as in Righini and Salani (2006). Once the search is over for all $i \in D$, we select the elementary shortest path with minimum loss among the non-dominated ones and further modify it to get a better start time from the depot. In order to do that, we first compute the latest feasible start time from the depot on the chosen path using Eq. (4), i.e., we try to

shift the start time from the depot forward in time without making the path infeasible. If such a shift is possible, we recompute the service start times of the remaining vertices and the total loss incurred on the path.

4. Computational Analysis

In order to assess the performance of the solution procedures proposed in Section 3, we conducted our experiments on the well-known data sets of Solomon (1983). Our computational analysis consists of two parts. In the first part, we test the three algorithms we have developed for our minimum loss ESPPRC. Then in the second part, we consider the set covering formulation of the corresponding VRPTW, in which the vehicles can start their trips at any desired time. We utilize the ESPPRC as the pricing subproblem of a column generation approach that solves the linear relaxation of this problem to optimality.

The algorithms are tested on the clustered, random, and random-clustered Solomon instances by considering either the first 50 or the first 100 customers. The values of the time windows, service times, locations, and demands of the customers are kept as in the classical instances. The vehicle capacity Q is set either to 50 or to 100. Since for each data set the number of customers and the vehicle capacity are assigned two different values, we obtain 116 instances. The travel distances between the vertices are defined as the Euclidean distances (with one decimal point and truncation). The travel times are set equal to the travel distances. The maximum amount of time S that the vehicle can be used is half the length of the horizon of the original instances. As suggested by Feillet et al. (2004) and Righini and Salani (2006, 2008), the revenues η_i are integer-valued parameters generated from a uniform distribution as $\eta_i \sim U(1, 20)$. Note that the revenues are only needed for the first section of the computational study. Since the objective of a classical ESPPRC is to minimize the total distance traveled, the cost α per distance unit traveled is taken implicitly as 1. Thus, in order for our experimental results to be coherent, we assign a value of 1 to α. Then, we carried out some preliminary experiments to determine the cost β per time unit use of the vehicle. Based on these initial tests, we observe that when β is greater than 0.1, the total time cost can be significantly higher than the total distance cost on an optimal path, which thus usually visits only one or at most two customers. Hence, we decided to set $\beta = 0.1$ to allow more customers to be visited on an optimal path. When $\beta = 0.1$, the preliminary

tests show that the ratio of the time cost to the distance cost on an optimal path varies between 0.15 and 1.53.

All the algorithms have been coded in C# and the computations have been performed on a workstation Intel Core i7-3930K 3.20 GHz processor with 64GB of RAM working under Windows 8 Pro operating system. A time limit of 3600 seconds is allotted for each solution method in the first part, whereas 10800 seconds are allowed for the column generation procedure in the second part.

4.1. Experimental Results for the ESPPRC

We compare the performances of the three methods in terms of accuracy and efficiency on the set of 116 instances. The results are presented in Tables 2-5. For each solution procedure, we report the computation times in seconds in column CPU and the number of Pareto-optimal complete paths in column Labels. The column Dominance shows the percentage of the eliminated partial paths among all generated partial paths for the mono-directional algorithm. Finally, in order to understand the accuracy of the heuristic, we compute the absolute value of the percent deviation (PD) of the best solution (BS) obtained by the heuristic from the optimal objective function value z^*. PD is expressed by the formula $|100 \times (BS - z^*)/z^*|$. An empty row in any table indicates that the instance cannot be solved by the corresponding algorithm within the time limit.

As expected, the heuristic turns out to be the most efficient among the three approaches as its CPU time is the smallest for most of the instances and it is always able to find a solution, except for two instances, namely $R104$ and $R112$ with 100 customers and capacity 100 (see Table 5). However, although the BBDS algorithm is an exact approach, it obtains an optimal solution within the allowed time for all but two instances which are $C104$ and $R112$ with 100 customers and capacity 100. (Note that the PD of the best solution obtained by the heuristic algorithm cannot be reported for $C104$ with 100 customers and capacity 100 even though the heuristic is able to find a solution.) On the other hand, for $R104$ with 100 customers and capacity 100, the only algorithm that is able to find a solution is the BBDS algorithm. Tables 2-5 exhibit an increase in the PD of the heuristic algorithm as the instances get larger. Especially, the PD can be very high for the random instances with 100 customers and capacity 100. The mono-directional algorithm is not able to provide an optimal solution for 15 instances and for almost all the instances that can be solved by the mono-directional algorithm

Table 2: ESPPRC Results for 50 Customers & Capacity 50

Instance	Bi-directional		Mono-directional			Heuristic		
	CPU	Labels	CPU	Labels	Dominance	CPU	PD	Labels
C101_50	0.12	1271	0.09	670	66.65%	0.02	0.00	90
C102_50	0.45	7267	0.58	3232	77.10%	0.01	0.00	176
C103_50	1.14	15416	1.89	5799	83.80%	0.01	0.00	179
C104_50	7.65	60406	13.95	13940	90.85%	0.04	0.00	618
C105_50	0.07	1979	0.05	1009	71.53%	0.00	0.00	85
C106_50	0.04	1276	0.03	696	73.07%	0.00	0.00	101
C107_50	0.10	2693	0.10	1248	77.01%	0.00	0.00	105
C108_50	0.45	8058	0.63	3538	73.59%	0.01	0.00	197
C109_50	1.58	18803	3.65	7067	78.17%	0.01	0.00	217
R101_50	0.08	162	0.03	106	72.25%	0.02	0.00	89
R102_50	0.04	867	0.03	395	77.69%	0.01	0.00	185
R103_50	0.06	1376	0.07	564	81.16%	0.02	0.00	343
R104_50	0.14	2462	0.17	899	84.26%	0.06	0.00	564
R105_50	0.02	406	0.01	233	74.25%	0.01	0.00	156
R106_50	0.05	1180	0.07	585	75.85%	0.02	0.00	263
R107_50	0.06	1389	0.07	555	82.04%	0.02	0.00	295
R108_50	0.07	1336	0.09	527	84.17%	0.03	0.00	304
R109_50	0.03	864	0.04	401	76.49%	0.01	0.00	200
R110_50	0.07	1628	0.17	692	75.70%	0.03	0.00	350
R111_50	0.07	1495	0.11	701	78.22%	0.03	0.00	327
R112_50	0.10	2003	0.15	768	82.30%	0.04	0.00	407
RC101_50	0.08	149	0.04	125	49.63%	0.02	4.79	86
RC102_50	0.01	206	0.02	156	55.91%	0.01	0.00	101
RC103_50	0.01	267	0.04	222	56.14%	0.01	0.00	135
RC104_50	0.02	325	0.03	225	68.40%	0.02	0.00	166
RC105_50	0.01	214	0.02	162	54.07%	0.01	0.00	101
RC106_50	0.01	252	0.02	195	57.45%	0.01	0.00	120
RC107_50	0.01	302	0.04	238	53.27%	0.02	0.00	159
RC108_50	0.02	340	0.05	241	62.14%	0.02	0.00	170

Table 3: ESPPRC Results for 50 Customers & Capacity 100

Instance	Bi-directional		Mono-directional			Heuristic		
	CPU	Labels	CPU	Labels	Dominance	CPU	PD	Labels
C101_50	0.15	2278	0.10	1115	65.52%	0.02	0.00	107
C102_50	2.00	25462	23.52	10112	75.54%	0.02	1.34	314
C103_50	8.97	64980	197.73	23690	82.25%	0.03	0.00	362
C104_50	101.18	316338				0.23	0.00	1804
C105_50	0.24	5205	0.52	2422	68.45%	0.00	0.00	100
C106_50	0.10	2683	0.13	1467	69.85%	0.01	10.07	150
C107_50	0.39	8449	2.29	3479	72.44%	0.01	0.00	160
C108_50	1.71	24120	24.10	10571	71.40%	0.02	0.57	303
C109_50	8.21	71251	213.14	26110	77.52%	0.02	0.00	309
R101_50	0.09	166	0.03	107	72.15%	0.02	0.00	89
R102_50	0.05	1113	0.06	488	75.49%	0.02	0.00	218
R103_50	0.11	2063	0.22	808	78.07%	0.06	0.00	439
R104_50	0.38	4205	0.82	1492	79.16%	0.25	30.79	807
R105_50	0.02	442	0.02	246	73.25%	0.01	0.00	162
R106_50	0.10	1686	0.32	902	67.65%	0.04	0.00	353
R107_50	0.13	2173	0.41	851	75.14%	0.06	0.00	402
R108_50	0.12	1844	0.29	687	81.33%	0.08	0.00	379
R109_50	0.04	1002	0.09	460	73.06%	0.02	0.00	210
R110_50	0.10	2182	0.47	891	72.54%	0.05	34.13	395
R111_50	0.10	2029	0.33	937	74.27%	0.06	59.37	388
R112_50	0.21	3157	0.89	1225	73.90%	0.11	220.48	515
RC101_50	0.09	183	0.06	150	42.31%	0.03	16.63	99
RC102_50	0.01	265	0.25	197	47.64%	0.02	10.67	115
RC103_50	0.02	358	0.79	295	49.69%	0.06	0.00	159
RC104_50	0.03	476	0.68	318	64.39%	0.10	4.03	212
RC105_50	0.01	271	0.27	204	44.55%	0.03	2.85	107
RC106_50	0.02	328	0.22	244	50.96%	0.03	6.42	137
RC107_50	0.03	441	2.06	332	46.31%	0.20	9.83	195
RC108_50	0.04	506	2.22	366	57.84%	0.32	0.00	243

Table 4: ESPPRC Results for 100 Customers & Capacity 50

Instance	Bi-directional		Mono-directional			Heuristic		
	CPU	Labels	CPU	Labels	Dominance	CPU	PD	Labels
C101_100	0.37	4729	0.29	2574	77.72%	0.03	0.00	191
C102_100	4.94	31818	8.43	8.43081	86.28%	0.04	0.00	412
C103_100	19.88	82579	65.42	31101	90.33%	0.07	0.00	671
C104_100	97.69	220940	209.39	46341	94.67%	0.12	0.00	1026
C105_100	0.73	8767	0.89	4313	80.84%	0.02	0.00	217
C106_100	0.92	10083	1.58	4422	83.95%	0.02	0.00	212
C107_100	1.12	12820	1.62	5269	84.92%	0.02	0.00	270
C108_100	4.33	32305	8.84	12268	84.66%	0.03	0.00	377
C109_100	36.50	81057	289.55	32925	84.58%	0.04	0.00	441
R101_100	0.16	1295	0.11	694	72.80%	0.05	0.00	451
R102_100	2.44	15192	6.35	5877	79.47%	0.38	0.00	1700
R103_100	14.97	48536	177.82	16787	79.88%	5.97	1.04	5782
R104_100	62.17	105580	1329.00	31600	82.26%	15.56	0.00	9373
R105_100	0.74	6363	1.99	3140	69.77%	0.25	11.82	1387
R106_100	9.17	31298	132.22	14207	74.24%	1.83	0.00	3480
R107_100	15.67	51275	250.51	18949	79.88%	5.12	27.41	5223
R108_100	18.94	63603	420.65	20449	81.93%	5.93	0.00	5724
R109_100	3.63	20308	20.92	7815	78.59%	0.89	0.00	2619
R110_100	20.62	53413	407.42	17891	76.78%	3.58	22.09	4613
R111_100	33.84	60128	1103.11	25055	73.99%	7.28	12.12	6517
R112_100	83.05	129694	2355.56	35084	81.51%	25.97	0.00	11664
RC101_100	0.19	1494	0.17	783	72.36%	0.06	75.32	425
RC102_100	0.50	4976	1.03	1848	75.46%	0.15	39.68	775
RC103_100	1.53	9645	3.64	3401	79.06%	0.53	0.00	1310
RC104_100	2.68	13737	12.32	4760	80.02%	1.25	64.88	2033
RC105_100	0.30	3668	0.52	1470	76.38%	0.12	0.15	758
RC106_100	0.45	4352	1.75	2059	71.25%	0.25	0.00	923
RC107_100	0.89	7947	2.68	3048	76.58%	0.49	0.00	1574
RC108_100	2.36	13500	10.30	4447	78.61%	1.35	0.00	2294

Table 5: ESPPRC Results for 100 Customers & Capacity 100

Instance	Bi-directional		Mono-directional			Heuristic		
	CPU	Labels	CPU	Labels	Dominance	CPU	PD	Labels
C101_100	1.10	13905	2.36	7070	76.73%	0.04	0.00	349
C102_100	188.54	289505				0.09	2.97	812
C103_100	858.48	870414				0.43	0.00	2217
C104_100						1.42		4521
C105_100	4.85	39375	25.82	16836	80.91%	0.04	0.00	416
C106_100	6.23	49732	112.57	19023	84.22%	0.03	19.96	385
C107_100	11.75	72867	144.46	26134	84.11%	0.05	9.89	569
C108_100	40.13	183247	1387.18	65829	84.29%	0.08	17.78	734
C109_100	444.10	642447				0.11	0.00	812
R101_100	0.20	1758	0.18	876	70.22%	0.07	22.80	518
R102_100	56.08	88340				43.84	0.87	8797
R103_100	238.16	216272				1346.35	1.63	22135
R104_100	2947.77	611441						
R105_100	2.22	13160	15.78	5942	62.32%	1.17	0.00	2192
R106_100	231.95	178815				223.07	4.98	14492
R107_100	362.00	262743				864.62	10.49	21541
R108_100	923.26	368591				999.06	13.09	28943
R109_100	33.22	66797				15.86	4.47	6488
R110_100	430.05	236487				300.08	12.27	15601
R111_100	924.76	317585				702.59	2.44	24321
R112_100								
RC101_100	0.36	2564	1.31	1205	66.58%	0.22	75.32	587
RC102_100	2.10	10523	40.52	4034	67.38%	1.23	24.91	1233
RC103_100	7.56	21925	247.24	7973	74.51%	9.06	4.59	2199
RC104_100	18.55	36304	648.53	12923	79.76%	24.00	0.00	4374
RC105_100	0.89	6752	22.69	2508	65.07%	0.77	0.15	1066
RC106_100	1.89	9115	112.54	4147	62.31%	2.50	6.24	1464
RC107_100	4.81	19075	508.36	7050	68.09%	1.40	0.00	2758
RC108_100	15.37	35812	2257.68	12409	71.34%	42.42	0.00	4690

within the time limit, the bounded bi-directional algorithm is faster than the mono-directional one. We can clearly claim that the bounded bi-directional algorithm beats the mono-directional one in terms of both accuracy and efficiency, and the heuristic in terms of accuracy. On the other hand, it can be observed that the heuristic algorithm produces solutions of good quality since it obtains the optimal solution for 72 instances. Another important remark about the heuristic is that among 116 instances, it can provide a solution in less than 1 second for 88 instances and in less than 10 seconds for 102 instances. This makes it a natural candidate for finding good solutions at the initial iterations of a CG procedure (see Subsection 4.2).

In addition to the CPU and the number of Pareto-optimal complete paths, the percentage of the dominated partial paths is reported for the mono-directional algorithm. Although the dominance for the mono-directional algorithm is realized functionwise following Proposition 7, the percentage of the dominated partial paths turns out to be 73.09% on the average showing that the dominance rules proposed in the paper are profitable. Furthermore, for 95 instances the percentage of the dominated paths is over 50% indicating the fact that we are able to detect more than half of the created partial paths as eligible to be eliminated.

4.2. Experimental Results for Column Generation

When VRPs with additional constraints are solved via branch-and-price, a set-covering formulation of the VRP is used where each column corresponds to a feasible vehicle path. Very tight lower bounds can be provided by solving the linear relaxation of the set-covering formulation, called the master problem, and branching techniques are used to discard fractional solutions. The master problem is solved through CG and the pricing subproblem turns out to be an ESPPRC.

In particular, the ESPPRC we are investigating can be encountered as a pricing subproblem if a BP algorithm is applied to solve the corresponding VRP with time windows and capacities, when the objective function has a time-dependent cost component. Then, the revenues are the dual prices of the visited vertices. In order to figure out the capability of the developed DP algorithms, we embed them into a CG procedure devised to solve the master problem. Note that our major concern is not to develop a full-fledged branch-and-price procedure; we rather focus on solving the linear relaxation of the VRPTW.

We employ CPLEX 12.5.1 to solve the restricted master problem at every iteration. In order to initialize the CG with a feasible linear program, a column is generated for every customer, where each column corresponds to a path that visits only one customer and starts and ends at the depot. At every iteration, at most $2 \times n$ columns with a negative reduced cost are generated and among them the most negative n ones are included into the master problem. If the number of generated columns reaches 3000, columns that never entered the basis are removed. In the first iterations, the heuristic algorithm presented in Subsection 3.3 is utilized until it fails to find columns with a negative reduced cost. Then, the exact bi-directional DP algorithm is called to find additional columns with negative reduced cost, if any.

Table 6: CG Results for 50 Customers

Instance	**Capacity 50**				**Capacity 100**			
	CPU	**# iter**	**# exact iter**	**column**	**CPU**	**# iter**	**# exact iter**	**column**
C101_50	0.83	13	2	459	3.13	20	6	772
C102_50	52.45	19	4	728	370.44	33	5	1451
C103_50	555.65	22	5	932	9559.48	51	8	2326
C104_50	537.32	27	3	1200				
C105_50	2.55	15	4	546	11.49	26	7	1047
C106_50	1.63	16	4	512	6.07	25	7	1013
C107_50	6.22	16	5	631	47.38	33	10	1469
C108_50	24.57	18	6	718	170.52	31	8	1382
C109_50	105.93	22	7	959	2151.21	51	12	2374
R101_50	0.31	7	3	248	0.32	7	3	245
R102_50	0.64	13	5	398	1.04	12	5	448
R103_50	2.01	14	4	466	3.73	14	4	555
R104_50	4.69	16	5	607	12.76	16	4	666
R105_50	0.28	10	4	354	0.32	9	3	320
R106_50	1.15	13	5	444	1.65	13	5	481
R107_50	2.26	13	4	496	5.32	15	4	608
R108_50	4.02	13	3	499	14.50	15	3	604
R109_50	0.92	12	5	471	1.72	13	5	475
R110_50	2.69	15	6	514	5.33	14	4	548
R111_50	2.67	14	4	518	6.08	14	4	524
R112_50	6.50	17	4	545	19.65	17	4	664
RC101_50	0.31	8	3	179	0.37	8	4	193
RC102_50	0.15	8	3	172	0.46	7	3	211
RC103_50	0.30	9	4	181	1.42	8	3	216
RC104_50	0.45	8	3	188	5.05	9	3	239
RC105_50	0.17	8	4	187	0.80	9	4	215
RC106_50	0.20	9	4	202	0.67	7	3	197
RC107_50	0.33	8	3	186	5.05	7	2	220
RC108_50	0.51	7	2	194	16.49	8	2	206

We assess the performance of the CG procedure on the same set of 116 instances used for the ESPPRC. The computational results are displayed in Tables 6 and 7. For each instance, we give the corresponding CPU time in seconds, the number of columns generated (# column), the total number of iterations (# iter), and the number of iterations at which the exact DP algorithm is used to solve the subproblem (# exact iter). We observe that the CG algorithm provides an optimal solution within the time limit for all but 16 instances. In particular, CG is able to solve more than half of the

Table 7: CG Results for 100 Customers

Instance	Capacity 50				Capacity 100			
	CPU	# iter	# exact iter	column	CPU	# iter	# exact iter	column
C101_100	8.94	16	2	1166	158.67	33	6	2792
C102_100	5128.26	23	3	1918				
C103_100								
C104_100								
C105_100	46.72	18	3	1361	2754.01	43	8	1537
C106_100	114.02	20	4	1570	10023.71	48	8	2161
C107_100	140.01	18	4	1454				
C108_100	1550.70	21	5	1780				
C109_100								
R101_100	3.53	16	6	942	5.60	15	5	1090
R102_100	58.23	21	6	1543	227.48	27	8	2167
R103_100	553.39	28	9	2237	9346.25	39	14	791
R104_100	878.87	31	7	2664				
R105_100	18.59	18	6	1295	125.51	25	8	1760
R106_100	155.33	23	6	1903	4146.51	32	9	2712
R107_100	502.10	25	5	2230				
R108_100	974.14	31	5	2792				
R109_100	113.19	22	6	1845	1945.79	31	8	2634
R110_100	556.09	27	7	2260				
R111_100	503.35	28	6	2363				
R112_100	666.77	30	3	2695				
RC101_100	3.55	13	5	832	17.68	15	6	1027
RC102_100	20.76	16	7	1091	110.41	18	6	1336
RC103_100	62.64	17	6	1147	895.65	20	6	1665
RC104_100	99.68	16	4	1173	7263.92	22	5	1884
RC105_100	13.64	14	5	918	97.20	17	6	1260
RC106_100	16.00	15	6	1098	317.42	19	8	1476
RC107_100	51.19	15	5	1146	2545.94	22	6	1792
RC108_100	108.04	16	3	1223	9566.38	21	4	1730

largest instances, namely the instances with 100 customers and capacity 100.

Another interesting observation is that the the number of iterations at which the exact DP algorithm is called is significantly smaller than the total number of iterations. This indicates that the heuristic is generally able to find columns with negative reduced costs when they exist and hence, that the heuristic performs well in many situations.

Finally, let us mention that we have also implemented a version of the CG algorithm based on the mono-directional DP algorithm, rather than the bi-directional one. In line with our expectations, the mono-directional variant is much less efficient than the previous one, be it in terms of CPU time (up to 35 times slower), or of the number of instances solved to optimality (it fails for 34 instances and can only solve 7 instances with 100 customers and capacity 100). In fact, there is no instance for which the mono-directional variant provides an optimal solution within the allowed time, and the bi-directional variant does not. For the sake of brevity, we do not provide more detailed results here.

5. Conclusion

In this paper we have considered a variant of the ESPPRC in which a capacitated single vehicle can start from the depot at any desired time and incurs costs based on both the total distance traveled and the total duration of the trip. The goal is to determine the optimal service start time of the vehicle from the depot as well as the trip to be performed so as to minimize a total loss function. To solve this problem, we propose two exact DP algorithms, a mono-directional one and a bounded bi-directional one, and a heuristic solution procedure. Since the structure of the problem creates infinitely-many Pareto-optimal states, we adapt the well-known DP algorithms according to this feature and develop piecewise linear functions modeling resources of the problem. We introduce appropriate dominance rules to discard feasible paths that cannot result in an optimal solution.

The solution methods are first tested on a set of ESPPRC instances of varying sizes. The results indicate that the bounded bi-directional DP algorithm outperforms the mono-directional DP algorithm. The heuristic algorithm is also quite satisfactory in the sense that it is the most efficient one and is able to find the optimal solution for 72 out of 116 instances. We also incorporate our algorithms in a CG scheme designed for the solution of the corresponding vehicle routing problem with side constraints. The computational study demonstrates that the heuristic algorithm works efficiently in this framework, so that the number of subproblems solved by the exact DP is kept reasonably small. As a consequence, the CG procedure is able to solve the linear relaxation of many large instances to optimality. This suggests that our methods can be successfully applied to the related VRP and can be used to solve instances of moderate sizes within a reasonable time by branch-and-price.

The ESPPRC we propose here can be extended in several ways. For example, one can handle multiple trips performed by a same vehicle. It is also possible to consider the delivery and backhaul case by dividing the customers into two disjoint subsets and forcing the vehicle to visit a backhaul customer after all its delivery customers are visited. We can easily adapt our exact algorithms to tackle such additional side constraints. Another direction which can be pursued is the acceleration of the ESPPRC. In order to accelerate the DP algorithm, we have employed the bounded bi-directional algorithm and identification of unreachable vertices. It is also possible to utilize other acceleration techniques such as the decremental state space relaxation proposed

by Righini and Salani (2008). Furthermore, the heuristic we propose in this study is in fact based on an exact DP algorithm for the classical ESPPRC. Thus, in order to accelerate the CG procedure, it may be possible to develop sophisticated ESPPRC heuristics to be used at the initial iterations. We leave all these questions for future work.

Acknowledgements. The project leading to these results was partially supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office (Grant P7/36) and by Fonds de la Recherche Scientifique (FNRS).

References

Aneja, YP, Aggarwal, V & Nair, KPK 1983, 'Shortest chain subject to side constraints', *Networks*, vol. 13, no. 2, pp. 295–302.

Baldacci, R, Mingozzi, A & Roberti, R 2011, 'New route relaxation and pricing strategies for the vehicle routing problem', *Operations Research*, vol. 59, no. 5, pp. 1269–1283.

Baldacci, R, Mingozzi, A & Roberti, R 2012, 'Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints', *European Journal of Operational Research*, vol. 218, no. 1, pp. 1–6.

Beasley, J & Christofides, N 1989, 'An algorithm for the resource constrained shortest path problem', *Networks*, vol. 19, no. 4, pp. 379–394.

Bettinelli, A, Ceselli, A & Righini, G 2011, 'A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows', *Transportation Research Part C*, vol. 19, no. 5, pp. 723–740.

Boland, N, Dethridge, J & Dumitrescu, I 2006, 'Accelerated label setting algorithms for the elementary resource constrained shortest path problem', *Operations Research Letters*, vol. 34, no. 1, pp. 58–68.

Ceselli, A, Righini, G & Salani, M 2009, 'A column generation algorithm for a rich vehicle-routing problem', *Transportation Science*, vol. 43, no. 1, pp. 56–69.

Cordeau, JF, Gendreau, M & Laporte, G 1997, 'A tabu search heuristic for periodic and multi-depot vehicle routing problems', *Networks*, vol. 30, pp. 105–119.

Dabia, S, Ropke, S, van Woensel, T & De Kok, T 2013, 'Branch and price for the time-dependent vehicle routing problem with time windows', *Transportation Science*, vol. 47, no. 3, pp. 380–396.

Desaulniers, G, Lessard, F, & Hadjar, A 2008, 'Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows', *Transportation Science*, vol. 42, no.3, pp. 387–404.

Desrochers, M 1988, 'An algorithm for the shortest path problem with resource constraints', Technical Report G-88-27, GERAD.

Desrochers, M, Desrosiers, J & Solomon, M 1992, 'A new optimization algorithm for the vehicle routing problem with time windows', *Operations Research*, vol. 40, no. 2, pp. 342–354.

Desrochers, M & Soumis, F 1988, 'A generalized permanent labeling algorithm for the shortest path problem with time windows', *Information Systems and Operations Research*, vol. 26, pp. 191–212.

Di Puglia Pugliese, L & Guerriero, F 2013, 'A survey of resource constrained shortest path problems: Exact solution approaches', *Networks*, vol. 62, no. 3, pp. 183–200.

Dror, M 1994, 'Note on the complexity of the shortest path models for column generation in VRPTW', *Operations Research*, vol. 42, no. 5, pp. 977–978.

Dumitrescu, I & Boland, N 2003, 'Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem', *Networks*, vol. 42, no. 3, pp. 135–153.

Feillet, D, Dejax, P, Gendreau, M & Gueguen, C 2004, 'An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems', *Networks*, vol. 44, no. 3, pp. 216–229.

Gutiérrez-Jarpa, G, Desaulniers, G, Laporte, G & Marianov, V 2010, 'A branch-and-price algorithm for Vehicle Routing Problem with Deliveries,

Selective Pickups and Time Windows', *European Journal of Operational Research*, vol. 206, no. 2, pp. 341–349.

Ioachim, I, Gélinas, S, Soumis, F & Desrosiers, J 1998, 'A dynamic programming algorithm for the shortest path problem with time windows and linear node costs', *Networks*, vol. 31, no. 3, pp. 193–204.

Irnich, S 2008, 'Resource extension functions: properties, inversion, and generalization to segments', *OR Spectrum*, vol. 30, no. 1, pp. 113–148.

Irnich, S & Desaulniers, G 2005, 'Shortest path problems with resource constraints' in Column Generation, G Desaulniers, J Desrosiers & MM Solomon, Springer, New York, pp. 33–65.

Irnich, S & Villeneuve, D 2006, 'The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$', *INFORMS Journal on Computing*, vol. 18, pp. 391–406.

Kohl, N 1995, 'Exact methods for time constrained routing and related scheduling problems'. Ph.D. Thesis, Institute of Mathematical Modeling, Technical University of Denmark, DK-2800 Lyngby, Denmark, Ph.D. Dissertation no. 16.

Liberatore, F, Righini, G & Salani, M 2011, 'A column generation algorithm for the vehicle routing problem with soft time windows', *4OR: A Quarterly Journal of Operations Research*, vol. 9, no. 1, pp. 49–82.

Lübbecke, ME & Desrosiers, J 2005, 'Selected topics in column generation', *Operations Research*, vol. 53, no. 6, pp. 1007–1023.

Righini, G & Salani, M 2006, 'Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints', *Discrete Optimization*, vol. 3, no. 3, pp. 255–273.

Righini, G & Salani, M 2008, 'New dynamic programming algorithms for the resource-constrained elementary shortest path problem', *Networks*, vol. 51, no. 3, pp. 155–170.

Righini, G & Salani, M 2009, 'Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming', *Computers and Operations Research*, vol. 36, no. 4, pp. 1191–1203.

Salani, M & Vacca, I 2011, 'Branch and price for the vehicle routing problem with discrete split deliveries and time windows', *European Journal of Operational Research*, vol. 213, no. 3, pp. 470–477.

Savelsbergh, MWP 1992, 'The vehicle routing problem with time windows: Minimizing route duration', *ORSA Journal on Computing*, vol. 4, no.2, pp. 146–154.

Solomon, MM 1983, 'Vehicle routing and scheduling with time window constraints: Models and algorithms'. PhD Thesis, University of Pennsylvania.

Tagmouti, M, Gendreau, M & Potvin, J-Y 2007, 'Arc routing problems with time-dependent service costs', *European Journal of Operational Research*, vol. 181, no. 1, pp. 30–39.