Pricing for the EVRPTW with Piecewise Linear Charging by a Bounding-Based Labeling Algorithm

Jenny Enerbäck
Department of Mathematics, Linköping University, Sweden
Scania CV AB, Södertälje, Sweden

Lukas Eveborn[1]
Department of Mathematics, Linköping University, Sweden

Elina Rönnberg
Department of Mathematics, Linköping University, Sweden

Abstract

The elementary shortest path problem with resource constraints (ESPPRC) is a common problem that often arises as a pricing problem when solving vehicle routing problems with a column generation approach. One way of solving the ESPPRC is to use a labeling algorithm. In this paper, we focus on how different bounding strategies for labeling algorithms can be adapted and strengthened for the ESPPRC that arises from the Electric Vehicle Routing Problem with Time Windows and Piecewise Linear Recharging function (EVRPTW-PLR). We present a new completion bound method that takes charging times into account, and show how the completion bound can be combined with ng-routes. Computational experiments show that the new completion bound combined with ng-routes significantly improves the performance compared to a basic labeling algorithm.

2012 ACM Subject Classification Mathematics of computing → Combinatorics; Mathematics of computing → Graph algorithms; Applied computing → Transportation; Mathematics of computing → Mathematical software

Keywords and phrases ESPPRC, EVRP, Bounding, Labeling Algorithm

Digital Object Identifier 10.4230/OASIcs.ATMOS.2024.3

Supplementary Material
Software (Public Repo): `https://gitlab.liu.se/eliro15/labeling-algorithm-for-evrptw-plr`

Funding This work was funded by the Swedish Energy Agency within the program FFI, Fordonsstrategisk Forskning och Innovation, under the grant Condore (P2022-00952).
Lukas Eveborn: This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Acknowledgements The Condore project is an important platform for this research as it provides close collaborations with the industrial partners Scania and Ragn-Sells as well as the Vehicular Systems division at Linköping University. We also want to thank the reviewers for useful feedback and especially the reviewer that pointed out an issue related to the combination of ng-routes and bounding.

1 Introduction

The interest in using electrical vehicles for transportation has been steadily increasing in the last years. One of the main reasons is the positive environmental aspects compared to using traditional combustion vehicles. As for the traditional vehicles, the routing and scheduling of the electric vehicles is a crucial aspect to consider. This has created a new type of vehicle routing problem, the Electric Vehicle Routing Problem (EVRP).

[1] Corresponding author

As for traditional routing problems, a common way of solving the EVRP is to use a column generation approach. Given a path representation of the routes, the column generation pricing problem can be represented as a shortest path problem. In such a setting, the performance of the pricing problem is crucial for the overall performance of the column generation approach. In this paper we are interested in the pricing problem for the EVRP with Time Windows, Piecewise Linear Recharging function and partial recharging (EVRPTW-PLR). This pricing problem can be represented as an elementary shortest path problem with resource constraints (ESPPRC). The ESPPRC can be solved in several ways, one of them is to use a labeling algorithm [9] which is the method we will be using.

Compared to traditional routing problems, the electrical routing problems come with a new set of challenges, the main one being the need to plan for recharging of the vehicles. This changes the characteristics of the ESPPRC and the labeling algorithm needs to be adapted to handle this, something that has been previously studied (see e.g. [5]). Another challenge is to model the recharging realistically, for which a piecewise linear function has been proven to be a good choice [11]. This does further complicate the ESPPRC and necessitates adaptions of the labeling algorithm, as done in e.g. [10] and [2].

How well a labeling algorithm performs is highly dependent on the different acceleration strategies that are used. Some of the most common acceleration strategies are bidirectional labeling [14], ng-routes [1] and bounding (see e.g. [3, 14]). The implementation of ng-routes is not affected by the piecewise linear charging function and can easily be implemented in a standard way. For bidirectional labeling, the case is different, as the standard strategy cannot be applied for piecewise linear charging functions. To the authors' best knowledge, this has not yet been done, and we find it hard to see how it can be done efficiently. For this reason, we consider bounding to be an important type of acceleration strategy in labeling algorithms for piecewise linear charging functions. Bounding builds on the idea to discard unpromising labels. There are several ways to do this, one is the resource bounding, which is a way to discard labels that cannot be completed to a feasible solution. Another way is the completion bounding, which is a way to discard labels that cannot be completed to a better solution than the best found so far.

In this paper, we focus on how different bounding strategies can be adapted and strengthened for the ESPPRC that arises from the EVRPTW-PLR. The paper extends the result of the master thesis [7] that introduced a bounding-based labeling algorithm for the EVRPTW with linear and partial recharging. The main contributions of this paper are:

- A new time-based completion bound method that takes charging times into account.
- Integration of completion bounds and ng-routes that handles the assumption of elementarity in completion bounds with the relaxation of elementarity in ng-paths.
- Publicly available labeling algorithm solving the ESPPRC for the EVRPTW-PLR (available at: `https://gitlab.liu.se/eliro15/labeling-algorithm-for-evrptw-plr`).

The rest of this paper is structured as follows. In Section 2, the problem statement and model will be presented. A short explanation of the foundations of the labeling algorithm will be presented in Section 3. The implemented bounding methods will then be presented in Section 4. Experimental results will be presented in Section 5 comparing the effect of the different bounding methods implemented, and finally the conclusions are presented in Section 6.

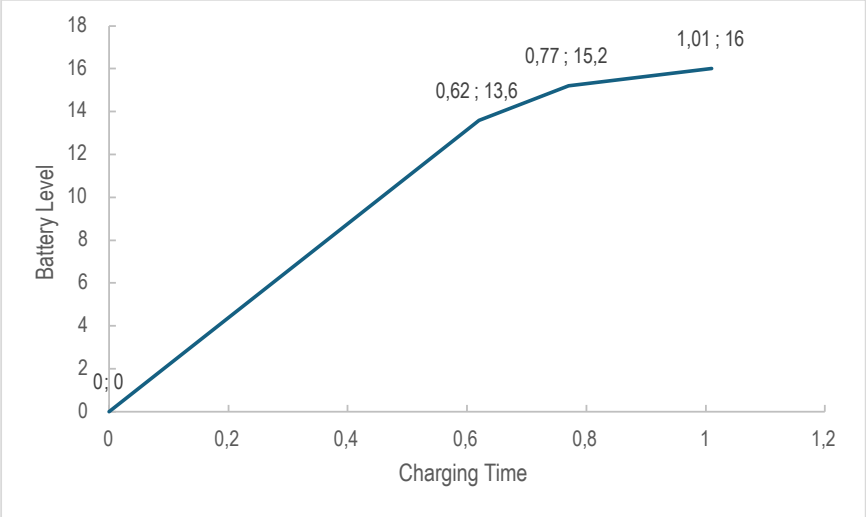


Figure 1 The piecewise linear charging curve used in this paper, with data from [11].

2 Problem Statement and Model

The problem considered is the ESPPRC that arises as a pricing problem from solving the EVRPTW-PLR within a column generation approach. In the EVRPTW-PLR, a fleet of electric vehicles with limited load and battery capacity should visit a set of customers, where each customer has a given time window and a given demand, while minimizing the total cost. Furthermore, the vehicles start and end at a depot, and can recharge at charging stations. The recharging rate is given by a piecewise linear function.

Inspired by the notation used in [5], we formulate the ESPPRC on a directed graph $G(V, A)$, where V denotes the set of nodes, and A denotes the set of arcs connecting the nodes. The set of nodes V consists of customer nodes, given by the set N, charging nodes, given by the set R, a start node o, and an end node d.

An arc between node $i \in V$ and node $j \in V$ is associated with three parameters: the cost c_{ij} of the arc, the energy consumption r_{ij} for traversing the arc, and the travel time t_{ij}, where also the service time at i is included if $i \in N$, i.e. i is a customer node. We assume that the triangle inequality holds for the time and energy consumption, i.e. $t_{ij} \leq t_{ik} + t_{kj}$ and $r_{ij} \leq r_{ik} + r_{kj}$ for all $i, j, k \in V$. Each customer node $i \in N$ is associated with a service time window $[e_i, l_i]$, during which the service can start. The vehicle is allowed to arrive early at a customer but has to wait until the start time to start service. Each customer node $i \in N$ is associated with a demand q_i, which since we are solving the pricing problem, only needs to be satisfied if the customer is visited.

The vehicle has a maximum load C and a maximum battery capacity Q. It can recharge at charging nodes where partial recharging is allowed. The battery level cannot go below 0.

The charging curve is given by a piecewise linear function, and we define it as in [10] where the charging curve is given by a set of pieces $P = \{1 \dots W\}$. Each piece $p \in P$ is defined by a start and end battery level τ_{p-1} and τ_p, and a recharging rate ρ_p (energy per timestep). In reality, the charging function is often concave since the charging rate decreases as the battery level increases [11]. With these assumptions, it holds that $0 = \tau_0 \leq \tau_1 \leq \ldots \leq \tau_W = Q$ and $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_W > 0$. An example is shown in Figure 1.

The vehicle starts at the start node at time 0 with a full battery and must return to the end node before the max time, $T_{\max}$. It can visit a customer node at most once, while charging nodes can be visited multiple times. The objective is to find a route that minimizes

the total cost of the arcs travelled, while satisfying the constraints. Because we are in a column generation setting, the arc costs are in fact reduced costs and as such they can become negative. In the EVRPTW-PLR, we assume the initial costs to be positive, and customers to be associated with dual variables, leading to that the outgoing arcs from customers can become negative, while the other arcs have their initial cost.

3 Labeling Algorithm

The basic version of the labeling algorithm is presented in Algorithm 1. The algorithm is implemented with a priority queue Γ of labels waiting to be extended and continues until there is no label to be extended from this queue. In order to find a good complete path early, the priority queue is ordered by increasing costs and the label with the lowest cost is first extended. This is especially important when using completion bounds, since the strength of the completion bounds depend on the quality of the incumbent upper bound. A label is denoted by L_i and represents a partial path $o \to i \in N$. Each label contains a number of resources, such as the cost of the partial path and the battery level. When extending a label L_i to a node j the function $Extend(L_i, j)$ updates the label resources given by the defined Resource Extension Functions (REFs) and verifies the feasibility of the extension of the path. All neighbours that can be reached from node i are denoted by Δ_i. The function $Dominance(\Lambda_j, L_j)$ checks dominance between the new label L_j and all labels in Λ_j, which is a bucket containing all labels ending at node j, given the decided dominance criteria. All labels that L_j dominates are discarded, and the $Dominance(\Lambda_j, L_j)$ returns false if L_j is dominated, whereupon L_j is discarded. The shortest path can, at the end of execution, be selected from the set of labels at the end node.

Algorithm 1 Basic labeling algorithm.

```
1  // Initialization of priority queue Γ with a start label Λ_o
2  Γ ← Λ_o
3  while Γ ≠ ∅ do
4  |  // Get next label to extend
5  |  L_i ← pop(Γ)
6  |  // Try to extend label to all outgoing neighbours of i
7  |  for j ∈ Δ_i do
8  |  |  // Try to extend label to node j
9  |  |  if L_j ← Extend(L_i, j) then
10 |  |  |  // If label finishes at end node, it is not added to priority queue
11 |  |  |  if j = d then
12 |  |  |  |  Λ_d ← L_j
13 |  |  |  // Else check dominance between new label and all labels at node j
14 |  |  |  else if Dominance(Λ_j, L_j) then
15 |  |  |  |  Γ ← L_j
16 |  |  |  |  Λ_j ← L_j
17 |  |  |  end
18 |  |  end
19 |  end
20 end
```

The resources of label L_i are defined as suggested by Lam et al. [10], with the notation of Desaulnier et al. [5], and presented below.

T_i^{cost}: Cost of the partial path $o \to i$.

T_i^{load}: Delivered load along the partial path.

T_i^{time}: Earliest service start time at node i that ensures time window and battery feasibility along the path.

T_i^{energy}: Battery level at i, assuming that minimal recharges have been performed at all visited charging stations.

$T_i^{\text{mrt}_p}$: Available charging time at each charging piece $p \in P$ that can be added at previous charging stops while ensuring time window feasibility.

$T_i^{\text{cust}_n}$: Indicates if a customer $n \in N$ cannot be visited in the extension of the path. This can either be because they already are visited or because they are unreachable from the path ending at node j.

A label L_i at node $i \in N$ is extended to a node $j \in N$ along the arc $(i, j) \in A$ by a number of Resource Extension Functions (REFs) that update the resources of the label, creating a new label L_j. The REFs are defined as suggested by Lam et al. [10], with the notation of Desaulnier et al. [5]. The REFs for the cost of the partial path and the delivered load are defined by (1) and (2).

$$T_j^{\text{cost}} = T_i^{\text{cost}} + c_{ij} \tag{1}$$

$$T_j^{\text{load}} = T_i^{\text{load}} + q_j \tag{2}$$

To update the time of the partial path and the energy level, we need to define some new concepts. Whenever a vehicle can arrive early at a customer, i.e. $T_i^{\text{time}} + t_{ij} < e_j$, a slack time is introduced, denoted W_{ij}. This slack time is given by $W_{ij} = \max\{0, e_j - T_i^{\text{time}} - t_{ij}\}$, which represents how much later it is possible to leave node i without delaying service start time at node j. If charging stations have been visited previous to node j, the slack time can be used for charging to the extent allowed by the available charging time $T_i^{\text{mrt}_p}$, $p \in P$. We assume that all available slack time, in the extent that it is possible, is used for charging and denote the energy charged during slack time by S_{ij}. Since the charging curve is piecewise linear, we have several charging pieces $p \in P$. In order to calculate S_{ij}, we therefore need to calculate the amount of time we can recharge at each charging piece, which we denote by $\delta_{pij}^{\text{slack}}$. This is calculated for each piece $p \in P$ in decreasing order of charging rate by $\delta_{pij}^{\text{slack}} = \max\{0, \min\{W_{ij} - \sum_{\mu=1}^{p-1} \delta_{\mu ij}^{\text{slack}}, T_i^{mrt_p}\}\}$. In this formula $T_i^{mrt_p}$ is, as mentioned, the available charging time at charging piece p from previous charging stops, and $W_{ij} - \sum_{\mu=1}^{p-1} \delta_{\mu ij}^{\text{slack}}$ is the remaining available slack time for charging at piece p, given the charging time at previous, more advantageous, pieces. The energy charged during slack time is then given by $S_{ij} = \sum_{p \in P} \rho_p \delta_{pij}^{\text{slack}}$ where ρ_p is the charging rate at piece p.

If the energy charged during slack time together with the initial energy is not enough to cover the energy consumption r_{ij} of the arc (i, j), i.e. $T_i^{\text{energy}} - r_{ij} + S_{ij} < 0$, additional charging is required. The required additional charging energy is denoted by Z_{ij} and the time for this additional charging is denoted X_{ij}. Similarly to how the energy charged during slack time was calculated, we need to calculate the amount of time $\delta_{pij}^{\text{extra}}$ we recharge at each charging piece $p \in P$. This is calculated for each piece $p \in P$ in decreasing order of charging rate by $\delta_{pij}^{\text{extra}} = \max\{0, \min\{T_i^{mrt_p} - \delta_{pij}^{\text{slack}}, (-T_i^{\text{energy}} + r_{ij} - S_{ij} - \sum_{\mu=1}^{p-1} \rho_\mu \delta_{\mu ij}^{\text{extra}})/\rho_p\}\}$. In this formula $T_i^{mrt_p} - \delta_{pij}^{\text{slack}}$ is the remaining available charging time for charging at piece p after slack charging has been done and $(-T_i^{\text{energy}} + r_{ij} - S_{ij} - \sum_{\mu=1}^{p-1} \rho_\mu \delta_{\mu ij}^{\text{extra}})/\rho_p$ is the

remaining time, given the piece p charging rate, needed to be recharged after slack charging has been done and additional charging has been done at previous, more advantageous, pieces. The additional charging time and energy are then given by $X_{ij} = \sum_{p \in P} \delta_{pij}^{\text{extra}}$ and $Z_{ij} = \sum_{p \in P} \delta_{pij}^{\text{extra}} \rho_p$, respectively.

The REF for time can now be given in (3), where, as stated, X_{ij} is the required additional charging time and e_j is the earliest allowed arrival at node j. The energy level is updated according to the REF defined in (4), where, as stated, r_{ij} is the energy consumption of the arc, S_{ij} is the energy charged during slack time, and Z_{ij} is the additional energy charged.

$$T_j^{\text{time}} = \max\{T_i^{\text{time}} + t_{ij} + X_{ij}, e_j\} \tag{3}$$

$$T_j^{\text{energy}} = T_i^{\text{energy}} - r_{ij} + S_{ij} + Z_{ij} \tag{4}$$

The REF for the available charging time is calculated for each piece $p \in P$ in decreasing order of charging rate, where τ_p is the end battery level of piece $p \in P$. It is given in (5) and calculated differently depending on whether j is a charging station or not. If a charging station, the time is calculated as the minimum of the maximum charging time at the piece p and the time needed to reach the end battery level at the piece p given current battery level T_j^{energy}. If not a charging station, it is calculated as the minimum of available charging time, after slack and additional charging has been added, and available time until the end of the time window after available charging time at more advantageous pieces has been added.

$$T_j^{\text{mrt}_p} = \begin{cases} \min\{(\tau_p - \tau_{p-1})/\rho_p, \max\{(\tau_p - T_j^{\text{energy}})/\rho_p, 0\}\}, \text{ if } j \in R \\ \min\{T_i^{\text{mrt}_p} - \delta_{pij}^{\text{slack}} - \delta_{pij}^{\text{extra}}, l_j - T_j^{\text{time}} - \sum_{\mu=1}^{p-1} T_j^{\text{mrt}_\mu}\}, \text{ if } j \in V \setminus R \end{cases} \tag{5}$$

Finally, customer reachability is updated in REF (6), either if they are visited or if they are unreachable from the path ending at node j. A node $n \in N \setminus \{j\}$ is marked as unreachable from the path ending at node j in the function $R(T_j^{\text{load}}, T_j^{\text{time}})$ by the value 1 if $T_j^{\text{load}} + q_n > C$ or $T_j^{\text{time}} + t_{jn} > l_n$ holds. This method of marking unreachable nodes as already visited was initially suggested by Feillet et al. [8].

$$T_j^{\text{cust}_n} = \begin{cases} T_i^{\text{cust}_n} + 1, \text{ if } j = n, \\ \max\{T_i^{\text{cust}_n}, R(T_j^{\text{load}}, T_j^{\text{time}})\} \; j \in N \setminus \{n\} \end{cases} \tag{6}$$

After the resources of the label have been updated, the feasibility of the extension is verified. As defined by [10], an extended label is feasible if the following four inequalities hold: $T_j^{\text{load}} \leq C$, $T_j^{\text{time}} \leq l_j$, $T_j^{\text{energy}} \geq 0$, and $(T_j^{\text{cust}_n}) \leq 1$ for all $n \in N$.

Once a label has been feasibly extended to node j, dominance is checked on the other labels finishing in node j. For a label $\tilde{L}_j$ at j to dominate another label $\hat{L}_j$ at j, the criteria given by Equations (7) through (11) must hold as defined by [10]. If $\tilde{L}_j$ dominates $\hat{L}_j$, $\hat{L}_j$ is discarded.

$$\tilde{T}_j^{\text{cost}} \leq \hat{T}_j^{\text{cost}} \tag{7}$$

$$\tilde{T}_j^{\text{load}} \leq \hat{T}_j^{\text{load}} \tag{8}$$

$$\tilde{T}_j^{\text{time}} \leq \hat{T}_j^{\text{time}} \tag{9}$$

$$\tilde{T}_j^{\text{cust}_n} \leq \hat{T}_j^{\text{cust}_n}, \; n \in N \tag{10}$$

$$\tilde{f}_j^{\text{energy}}(t) \geq \hat{f}_j^{\text{energy}}(t), \; t \in [\hat{T}_j^{\text{time}}, l_j] \tag{11}$$

In Equation (11), battery levels are compared given a later arrival time at j with the help of the function $f_j^{\text{energy}}(t)$ which outputs the battery level given an arrival time t. This has to be done because T_j^{energy} is only the energy level given a *minimal* recharge, but since

the extensions are unknown at the time of dominance checking, it might be needed to arrive later but with a higher battery level. So in order for the label $\tilde{L}_j$ to dominate $\hat{L}_j$, it has to be able to achieve a higher battery level for every possible arrival time at node j of $\hat{L}_j$. The possible arrival times are all the times between $\hat{T}_j^{\text{time}}$ and the latest service start time at node j, l_j. However, since the charging curves are piecewise linear, it suffices to verify this at start points, breakpoints and endpoints of the charging curves. For a more thorough explanation and definition of this function, we refer to Lam et al. [10].

4 Bounding Methods

A well-known challenge when applying labeling algorithms is that in the late iterations, a huge number of labels have been generated and a large portion of those cannot be completed into a solution of interest. To avoid this, or at least discard some unpromising ones along the way, bounding methods can be used. Bounding methods use optimistic bounds on the completion of a path to discard labels that cannot yield a best solution. This can be done in different ways, but common for all is that the methods need to rely on fast computations to contribute to computational efficiency. For that reason the methods can often be greedy and do not necessarily use the network structure or all information in the problem. Furthermore, it is often important to use the problem structure to make the bounding methods even more efficient. We choose to implement two types of bounding methods, resource bounds and completion bounds, to improve the computational time of the labeling algorithm. These are presented in Section 4.1 and 4.2, respectively.

4.1 Resource Bounds

The main idea behind resource bounds is that if there is not enough resources left to reach the end node, a label can be discarded – even if there still are resource feasible extensions to some nodes. We implement resource bounds for battery feasibility and time feasibility, as they are the only resources that directly can prevent feasible completions of a path. The resource bounds are implemented in the $Extend(L_i, j)$ function.

When formulating our resource bounds, inspiration is taken from the criterion $W^r + w_{ij}^r + \underline{w}_{jd}^r > W_{\max}^r$ suggested by Boland et al. [3], where W^r is the consumption of resource r along the partial path, w_{ij} the consumption on the arc (i, j), and $\underline{w}_{jd}^r$ a lower bound on the resource consumption from node j to the depot.

Starting with battery feasibility, it is checked that the partial path can reach the end node or a charger without emptying the battery. For a node $j \in V$, let $r_{j\xi}$ be the energy consumption to travel from j to the closest node that is either a charger or the end node, i.e. $\xi \in R \cup \{d\}$. If the current battery level together with the available charging time is not enough to reach ξ, the label can be discarded. This is done by checking if $T_j^{\text{energy}} + \sum_{p \in P} \rho_p T_j^{\text{mrt}_p} - r_{j\xi} < 0$.

For time feasibility, we generate an optimistic bound on the time it takes to reach the end node and use this to check that the end node can be reached within the max time limit. Beyond travel time we also consider possible extra charging time that is needed to reach the end node. In order to make sure it is an optimistic bound, i.e. no labels are discarded wrongly, it is assumed that charging can be done "on the road" i.e. discarding the possible extra travel time and energy it takes to reach a charging station. For the same reason it is also assumed that the recharge is done at the fastest charging rate, ρ_1. To formulate the condition, we first need to calculate the possible required additional energy to reach the end

node, which is given by $Y_j = \max\{0, r_{jd} - T_j^{\text{energy}}\}$, where r_{jd} is the energy consumption to reach the depot from current node $j \in N$. The feasibility check can then be done by $T_j^{\text{time}} + t_{jd} + Y_j/\rho_1 > T_{\text{end}}$. If the inequality holds, the label is discarded.

4.2 Completion Bounds

For completion bounds, the idea is to discard labels that are guaranteed to not be completed to a better solution than the current best solution. This is done by generating an optimistic bound on the cost of completion of the path, denoted z_{LB}. A label L_j with current cost T_j^{cost}, can then be discarded if $T_j^{\text{cost}} + z_{\text{LB}} > UB$, where UB is the current upper bound on the cost for a finished path. We choose to calculate two completion bounds, one by solving a knapsack problem with respect to the vehicle load constraint and one by solving a problem that takes both travel time and charging time into account. Both are inspired by the knapsack bound formulation of Righini and Salani [14].

For the load-based bound, the implementation is similar to the formulation of [14], but with some adjustments to our problem. They do not have charging nodes, but as it turns out, the charging nodes can be disregarded in the calculation of the bound. The reason for this is twofold, firstly, the battery constraints are relaxed in the bound, so visits to chargers are not required for feasibility reasons. Secondly, in a standard column generation context, the dual variables of the charging nodes are non-existent as there are no master constraints related to them, and it can therefore never be profitable to visit chargers assuming positive initial costs. With this in mind, let then S be the set of already visited customers. We can then define the continuous decision variable $y_k \in [0, 1],\ k \in V \setminus S$ that state how much a customer is visited.

We define the cost of visiting a node $i \in N$ as the cost of its cheapest outgoing arc, i.e. $u_i = \min_{l \in N \setminus R} c_{il}$. This is used to get the cost of visiting a customer in the case when $i \in N \setminus S$ which we only would want to do if u_i is negative. It is also used to get the cost of leaving the current node j in the case when $i = j$, which is necessary to get the right number of arcs in the solution. Compared to [14], we consider the outgoing arcs from the nodes $k \in N$ instead of the incoming arcs to calculate the least cost since the service time in our case is included in the outgoing arcs and not the incoming arcs. Note also that to calculate the cost for visiting a customer, outgoing arcs to charging nodes are disregarded, which is valid thanks to the relaxed battery constraints and the non-profitability of visiting charging nodes. This strengthens the bound in the case where the closest node to a customer node is a charging station since the arc between them would then be the arc with the least cost. The load-based completion bound for a label L_j is presented in Equation (12).

$$\begin{aligned} z_L = \min & \sum_{k \in N \setminus S} u_k y_k + u_j \\ \text{s.t. } & T_j^{load} + \sum_{k \in N \setminus S} q_k y_k \leq C \\ & 0 \leq y_k \leq 1, \quad \forall k \in N \setminus S \end{aligned} \tag{12}$$

Another important adjustment compared to the formulation of Righini and Salani [14] is that when selecting the cheapest outgoing arc for a node, *all* arcs to customer nodes are considered, not just arcs to not visited customers, i.e. $\min_{l \in N \setminus R} c_{kl}$ instead of $\min_{l \in N \setminus (S \cup R)} c_{kl}$. Although the second formulation of the two generates a stronger bound, it is more computationally expensive, since the least arc costs must be computed each time the

bounding method is used. With our formulation, however, the least cost arc of a node, and further the least time and least energy consumption arc, which will be used in the second bound, can be precomputed.

The time-based bound can be seen as an extended version of a knapsack problem on maximal route time. It considers arc travel time and, to make sure that the battery level stays positive, the possible extra required charging time. To obtain a lower bound, the same assumptions as in the resource bounds are made, i.e. it is assumed that the charging can be done "on the road" and at the fastest charging rate, ρ_1. Most of the notation is the same as in the vehicle load bound, but some new parameters and variables need to be introduced. To handle the extra potential charging time, a new continuous variable ζ is introduced, which is the total extra charging energy that needs to be added in order to ensure battery feasibility. We also introduce the minimum energy consumption r_i of visiting a node $i \in V$, and the minimum travel time t_i to visit a node, $i \in V$. Both are calculated in the same manner as the lowest cost, u_i, by finding the cheapest outgoing arc from a node $i \in V$ for energy and time respectively, i.e. $r_i = \min_{l \in N \setminus R} r_{il}$ and $t_i = \min_{l \in N \setminus R} t_{il}$. The time-based completion bound for a label L_j is presented in Equation (13).

$$
\begin{aligned}
z_T = \min & \sum_{k \in N \setminus S} u_k y_k + u_j \\
\text{s.t. } & T_j^{\text{time}} + t_j + \zeta/\rho_1 + \sum_{k \in N \setminus S} t_k y_k \leq T_{\max} \\
& r_j + \sum_{k \in N \setminus S} r_k y_k \leq \zeta + T_j^{\text{energy}} \\
& 0 \leq y_k \leq 1, \quad \forall k \in N \setminus S \\
& 0 \leq \zeta
\end{aligned}
\tag{13}
$$

Even if z_T generates a good bound, it is not as straightforward to calculate efficiently as z_L. It can always be solved with an LP-solver, but that would not be that effective. However, if we assume that the problem instance has the properties that $t_1 \geq t_2 \Rightarrow r_1 \geq r_2$, then we can use a greedy algorithm to calculate the bound, presented in Algorithm 2. Note that this assumption holds for the instances tested in this paper, the EVRPTW dataset of Schneider et al. [15], where the relation between travel time and energy consumption is linear, and the service time are same for all customers in one instance. As input to the algorithm we precompute the most profitable customers and add them to a list, denoted K, which is ordered by increasing value of the quotients $\frac{u_k}{t_k + r_k/\rho_1}$. The algorithm then iterates over the nodes in this order and adds the most profitable nodes to the path as long as the max time limit is not violated.

4.2.1 Integration of ng-routes with Completion Bounds

Another popular acceleration strategy for a labeling algorithm is ng-routes, first suggested by [1]. In ng-routes, the elementarity constraints on the paths are partly relaxed allowing revisits to a customer as long as the customer is not in the ng-set of the current node. The ng-set is unique for each node and contains the nodes for which elementarity is checked when extending a label. If a customer is not in the ng-set of the node we are extending the label from, it can be added to the path even if it has been visited before, allowing for subtours. The size of the ng-set, n, is set according to what fits the problem and is often chosen to be the n closest customers to the node, including itself. This is also the approach we have chosen to use. One of the benefits of using ng-routes is that if the best

Algorithm 2 Completion bound with time and battery constraints.

$C \leftarrow 0$ // Initial cost
$\sigma \leftarrow \max\{0, (r_j - T^{\text{energy}})/\rho_1\} + t_j$ // Time needed to leave j
$T \leftarrow T^{\text{end}} - T_j^{\text{time}} - \sigma$ // Time left
$E \leftarrow \max\{0, T_j^{\text{energy}} - r_j\}$ // Remaining energy
$i \leftarrow 0$
while $T \geq 0$ **do**
  // Find the most profitable customer to add
  $k \leftarrow K[i]$
  // Check that the customer can be added and have a positive effect
  if $T_j^{cust}[k] = 0 \ \wedge \ u_k < 0$ **then**
    // Time needed to add customer k
    $\sigma \leftarrow \max\{0, (r_k - E)/\rho_1\} + t_k$
    // Check if customer can be fully added, else add partially
    if $T - \sigma \geq 0$ **then**
      $C \leftarrow C + u_k$
      $T \leftarrow T - \sigma$
      $E \leftarrow \max\{0, E - r_j\}$
    else
      // Percentage that can be added
      $\lambda \leftarrow (\sigma - T)/\sigma$
      $C \leftarrow C + \lambda u_k$
      $T \leftarrow T - \lambda\sigma$
      $E \leftarrow E + \lambda r_k$
    end
  end
  i++
end

path is elementary when finishing the labeling algorithm, it will be also be an optimal elementary path. Using ng-routes is a powerful acceleration strategy. It can, however, not be used directly with the implemented completion bounds. The reason for this complication is that the implemented completion bounds create *elementary completions* to the partial paths, which are not necessarily optimistic completions to an ng-route. Since subtours are allowed for ng-routes, it has to be considered that a node can be added multiple times to the completion of the path, and the calculation of completion bounds need to be adjusted accordingly.

There are some existing approaches to combine ng-routes with completion bounds in the literature, but none really fits with our problem. For implementations where each problem is solved several times to find elementarity, i.e. iteratively extending the size of the ng-set until the optimal path is elementary, labels from the previous iteration can be used to compute lower bounds on the reduced cost of the completion of a path (used by [13, 4, 12]). Another approach to use completion bounds in the ng-route ESPPRC was suggested by Baldacci et al. [1], and was applied to the electric ng-route SPPRC by Duman et al. [6]. They use the completion bounds as a big part of the solution method starting each solving process by, for each customer, running an exact labeling algorithm with the customer as a start node

and with $t^{\text{start}} = T_{\max} - \Delta t$ as the start time. The results from these runs are then used as lower bounds when solving again, but with the start time $t^{\text{start}} = T_{\max} - 2\Delta t$. This is then repeated until the full problem is solved.

However, we want to integrate completion bounds with ng-routes without significant changes to the algorithm. When computing the completion bound we therefore assume that there are no elementary constraints, to ensure it to be an optimistic bound on the solutions also in the case of using ng-routes. However, by considering other constraints, the bound can be strengthened. Firstly a customer cannot be added at all if it is not reachable from the current node, i.e. if $T_j^{\text{time}} + t_{jl} < l_l$. Secondly, given it is reachable, an upper bound on the total number of times a node can be added to the path can be calculated. Provided the least time outgoing arc $t_{lk}^{\min}$ from node l and the least time incoming arc $t_{kl}^{\min}$ to node l, then $t_{lk}^{\min} + t_{kl}^{\min}$ is the minimum time required to leave and come back to node l. Furthermore, the earliest arrival at the node l can be calculated as $T_l^{\text{start}} = \max\{T_j^{\text{time}} + t_{jl}, e_l\}$, comparing the earliest allowed arrival e_l with the possible earliest arrival given the label T_j^{time} at node j and the travel time t_{jl}. Using both of these, an upper bound on the number of visits at node l can be calculated by $\left\lfloor \frac{l_l - T_l^{\text{start}}}{t_{lk}^{\min} + t_{kl}^{\min}} + 1 \right\rfloor$.

To further strengthen the upper bound on the number of visits, the elementary constraints that exist in an ng-routes setting can be taken into consideration as follows. In order to be allowed to come back to a node l that already has been visited, the route must be extended from a node k for which l is not in its ng-set. Provided the least time outgoing arc from node l to a node for which l is not in its ng-set $t_{lk}^{\min_{\text{ng}}}$ and the least time incoming arc from a node for which l is not in its ng-set to node l, $t_{kl}^{\min_{\text{ng}}}$, then $t_{lk}^{\min_{\text{ng}}} + t_{kl}^{\min_{\text{ng}}}$ can be used instead as the minimum time required to leave and come back to node l. The strengthened upper bound on the number of visits at node l can hence be calculated as

$$\left\lfloor \frac{l_l - T_l^{\text{start}}}{t_{lk}^{\min_{\text{ng}}} + t_{kl}^{\min_{\text{ng}}}} + 1 \right\rfloor.$$

In practice this adaption to make completion bounds work with ng-routes is implemented by adding nodes to the path in order of decreasing profitability given that it is reachable and that the multiplicity is not violated. Note that in order to ensure an optimistic bound, given the ng-setting, a node can be added multiple times in a row as long as the multiplicity constraint is not violated and that there is place left in the knapsack.

5 Experimental Results

The performance of the labeling algorithm with the suggested acceleration strategies is evaluated using the EVRPTW benchmark data set of Schneider et al. [15] which is based on Solomon's benchmark instances for the VRPTW [16]. Tests are performed on all 29 instances from the groups C100, RC100, and R100, that have 100 customer nodes with narrow time windows and 20 charging nodes with no time windows. Each instance is tested in three different simulated column generation environments, resulting in a total of $3 \times 27 = 89$ instances.

To simulate a column generation environment, values of the dual variables for customers, $\pi_i,\ i \in N$, are generated as random integer variables from a uniform distribution on the interval $\{0, \ldots, 20\}$ as suggested by Feillet et al. [8]. These are used to update the arc costs of outgoing arcs from customers nodes $i \in N$: $c_{ij} = d_{ij} - \pi_i$, where d_{ij} is the Euclidean distance between node i and node j. For all other nodes $i \in V \setminus N$ the outgoing arc costs are set to $c_{ij} = d_{ij}$. The generated dual values are available in the public repo (available at `https://gitlab.liu.se/eliro15/labeling-algorithm-for-evrptw-plr`).

Table 1 Number of instances solved for each group and acceleration strategy

Series	basic	ng-routes	bounds load	bounds time	ng-routes+bounds
$C100$	21/27	27/27 (20)	21/27	21/27	27/27 (20)
$RC100$	21/24	24/24 (19)	23/24	24/24	24/24 (19)
$R100$	18/36	25/36 (16)	20/36	21/36	29/36 (17)
Total	60/87	76/87 (55)	64/87	66/87	80/87 (56)

Table 2 Aggregated results for each group and acceleration strategy

Series	basic	ng-routes		bounds load		bounds time		ng-routes+bounds	
	$\bar{t}$	$\bar{t}$	Δt[%]	$\bar{t}$	Δt[%]	$\bar{t}$	Δt[%]	$\bar{t}$	Δt[%]
$C100$	15.11	3.09	-79.60	9.06	-40.07	1.80	-88.06	**0.74**	**-95.11**
$RC100$	24.33	12.69	-47.87	14.15	-41.84	11.83	-51.38	**6.17**	**-74.64**
$R100$	620.67	515.09	-17.01	417.23	-32.79	256.75	-58.63	**165.01**	**-73.41**

In the dataset for EVRPTW it is assumed that the charging curve is linear, where the battery capacity Q and the inverse recharging rate ω is unique for each instance. To adapt the dataset to a piecewise linear charging function, we fit the charging curve in Figure 1, by scaling the time of breakpoints by $\omega Q \frac{t_{\text{old}}}{1.01}$ and the battery levels by $Q\frac{e_{\text{old}}}{16}$. From this it is then easy to calculate, for each instance, the set of pieces in the piecewise linear charging function with its corresponding breakpoints and recharging rates which are available in our public repo.

Before running the algorithm the graph is reduced by removing infeasible arcs, i.e. arcs between nodes $i \in V$ and $j \in V$ such that $e_i + t_{ij} > l_j$, $q_i + q_j > C$ or $r_{ij} > Q$.

The algorithm was implemented in C++ and compiled with GCC 11.4.0. It was tested on a PC with AMD Ryzen Pro565OU CPU at 2.301 GHz, 32 GB RAM on Ubuntu with WSL.

We report results of each instance solved using (i) no acceleration strategy, (ii) ng-routes, (iii) resource bounds + load-based completion bound, (iv) resource bounds + time-based completion bound. Finally, a combination of all acceleration strategies is tested (v): ng-routes + resource bounds + load-based completion bound + time-based completion bound. A maximum time of 1 hour was set for each instance.

To minimize unnecessary computational effort, the completion bounds are only computed if the resource consumption for the constrained resource exceeds 20% of the total resource availability. This threshold was chosen from pretrial tests (Figure 2 in Appendix A) of different thresholds on resource consumption. In the combined setting, the time-based bound is computed first and if $T_j^{\text{cost}} + z_T > UB$ the label is discarded straight away. Otherwise, the load-based bound is computed to check if $T_j^{\text{cost}} + z_L > UB$.

The size of the ng-sets were set to 10 ($C100$), 15 ($RC100$), and 20 ($R100$) nodes respectively. For each group of instances, these were the smallest sizes of multiples of 5 of the ng-sets for which the optimal solutions of at least half of the instances were elementary.

In Table 1, we report how many instances from each group were solved to optimality within the 1-hour time limit. For the acceleration strategies using ng-routes, it is reported in parentheses how many of the optimal paths were elementary. It is worth to remember that in a column generation environment, as long as there is one elementary path with negative reduced cost among the finished paths, the result is useful. Full results for each instance and acceleration strategy can be found in Table 3–Table 5 in Appendix B.

Table 2 shows aggregated time results per group and acceleration strategy. To make a fair comparison, only instances that were solved with an elementary optimal solution by all acceleration strategies within the 1-hour limit are included in the calculation. Deviation for a strategy x and group y is calculated compared to the basic version by $\Delta\bar{t}^y_x = 100 \cdot (\bar{t}^y_x - \bar{t}^y_{\text{basic}})/\bar{t}^y_{\text{basic}}$.

Comparing the results in Table 1 and Table 2, we can state that the combination of ng-routes and completion bounds (v) is the most effective acceleration strategy. It solves the largest number of instances within the time limit, and achieves the largest reduction in computational time compared to the basic version, from -73% to -95% in average, depending on the instance group. When it comes to the computational time, the pattern for the instances not included in the aggregated results is the same as for the instances included, i.e. the combination of ng-routes and completion bounds is the most effective strategy.

Comparing the two implemented completion bounds, the time-based bound (iv) is on average more efficient than the load-based bound (iii) for all groups of instances, and for some the difference is significant. This indicates that the vehicle load is not as often a binding constraint as the time constraint for the instances tested, at least when taking charging time into account.

Looking a bit closer at the full results, Table 3–Table 5 in Appendix B, it can be seen that the main difference in computational time between the other acceleration strategies and acceleration strategy (v) shows on the more difficult instances. On instances that are easier to solve, the difference in computational time between the strategies is not as large, and often one of the plain bounding strategies is the most effective. The reason for this is most likely that completion bounds generated when combining it with ng-routes is weaker compared to when not combined with ng-routes, and the usage of ng-routes does not compensate for this on easier instances.

6 Conclusion

In this paper, we propose bounding methods to accelerate a labelling algorithm used for solving the ESPPRC as the pricing problem of the EVRPTW with a piecewise linear charging curve and partial recharging. Two types of bounding methods are implemented, resource bounds and completion bounds. For completion bounds we implemented two versions, one load based and one time based. The latter is a new stronger completion bound that takes travel time as well as charging time into account. A strength of both the completion bounds is that they are cheap to compute, since the information required can be preprocessed. Furthermore, we propose a way to integrate the bounding methods with ng-routes, another popular acceleration strategy.

Experimental results show that the combination of ng-routes and bounding methods was most efficient, reducing the computational time from -73% to -95% in average, depending on the instance group, compared to a basic labelling algorithm on benchmark instances with 100 customers and 20 chargers. The results also show that the time-based completion bound was more efficient than the load-based completion bound for all groups of instances.

Our efficient way of computing the time-based completion bound uses a specific relation between travel time and energy consumption. Of interest for future work is to investigate efficient ways of computing this time-based completion bound when these assumptions do not hold.

References

1 Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Oper. Res.*, 59(5):1269–1283, 2011. doi:10.1287/OPRE.1110.0975.

2 Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. *CoRR*, abs/1910.09812, 2019. arXiv:1910.09812.

3 Natashia Boland, John Dethridge, and Irina Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Oper. Res. Lett.*, 34(1):58–68, 2006. doi:10.1016/J.ORL.2004.11.011.

4 Claudio Contardo and Rafael Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discret. Optim.*, 12:129–146, 2014. doi:10.1016/J.DISOPT.2014.03.001.

5 Guy Desaulniers, Fausto Errico, Stefan Irnich, and Michael Schneider. Exact algorithms for electric vehicle-routing problems with time windows. *Oper. Res.*, 64(6):1388–1405, 2016. doi:10.1287/OPRE.2016.1535.

6 Ece Naz Duman, Duygu Tas, and Bülent Çatay. Branch-and-price-and-cut methods for the electric vehicle routing problem with time windows. *Int. J. Prod. Res.*, 60(17):5332–5353, 2022. doi:10.1080/00207543.2021.1955995.

7 Jenny Enerbäck. A labelling algorithm for the resource constrained elementary shortest path problem. *Master Thesis, Linköping University*, 2024. URL: urn:nbn:se:liu:diva-205286.

8 Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004. doi:10.1002/NET.20033.

9 Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In *Column generation*, pages 33–65. Springer, 2005.

10 Edward Lam, Guy Desaulniers, and Peter J. Stuckey. Branch-and-cut-and-price for the electric vehicle routing problem with time windows, piecewise-linear recharging and capacitated recharging stations. *Comput. Oper. Res.*, 145:105870, 2022. doi:10.1016/J.COR.2022.105870.

11 Alejandro Montoya, Christelle Guéret, Jorge E. Mendoza, and Juan G. Villegas. The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110, 2017. doi:10.1016/j.trb.2017.02.004.

12 Diego Pecin, Artur Alves Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. *Math. Program. Comput.*, 9(1):61–100, 2017. doi:10.1007/S12532-016-0108-8.

13 Rafael Martinelli Pinto. Exact algorithms for arc and node routing problems. *These de doctorat, Pontifcia Universidade Católica do Rio de Janeiro*, 2012.

14 Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discret. Optim.*, 3(3):255–273, 2006. doi:10.1016/J.DISOPT.2006.05.007.

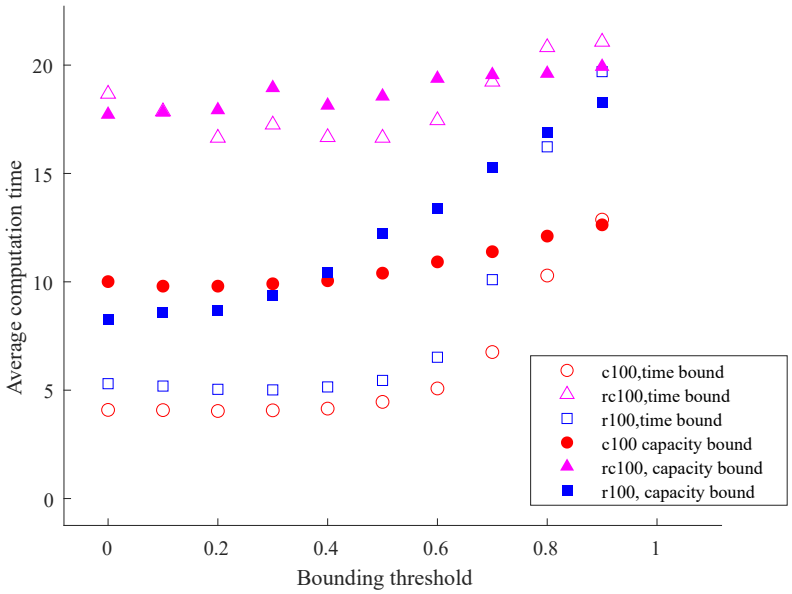
15 Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transp. Sci.*, 48(4):500–520, 2014. doi:10.1287/TRSC.2013.0490.

16 Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.*, 35(2):254–265, 1987. doi:10.1287/OPRE.35.2.254.

A Pretrial results

In Figure 2 we present the average computation time of instances in $C100$ ($c101$, $c105$, $c106$, $c107$, $c108$), $RC100$ ($rc101$, $rc102$, $rc105$, $rc106$, $rc107$) and $R100$ ($r101$, $r105$, $r109$) using different starting thresholds of resource consumption for the two completion bounds.

Figure 2 Average computation times using different bounding thresholds.



B Results

We report results of each instance in Table 3–Table 5, divided by the three simulated column generation environments, solved using (*i*) no acceleration strategy, (*ii*) ng-routes, (*iii*) resource bounds + load-based completion bound, (*iv*) resource bounds + time-based completion bound and (*v*): ng-routes + resource bounds + load-based completion bound + time-based completion bound. For each instance and strategy we report the solving time in seconds, and how many non-dominated labels were left at the end of the algorithm, in thousands. We indicate for the results using ng-routes with a star which instances did not have an elementary optimal solution. The ng-set size was set to 10, 15, and 20 for the instance groups $C100$, $RC100$, and $R100$, respectively. The instances that could not be solved within the 1-hour time limit are indicated with '-'.

Table 3 C100, RC100 and R100: basic version, ng-routes, completion bounds and ng-routes with bounds. First simulated column generation environment.

	basic		ng-routes		bound load		bound time		ng+bounds	
	Labels	Time	Labels	Time	Labels	Time	Labels	Time	Labels	Time
$c101$	10.57	0.72	13.64	1.06	7.02	0.38	**6.64**	**0.26**	6.91	0.33
$c102$	292.68	1639.79	61.19*	20.03*	257.23	828.44	**153.99**	**156.62**	41.22*	7.04*
$c103$	-	-	187.81	450.61	-	-	-	-	**122.50**	**81.83**
$c104$	-	-	300.45*	1136.58*	-	-	-	-	237.44*	509.01*
$c105$	13.75	1.22	16.81	1.49	11.00	0.76	**7.28**	**0.38**	8.65	0.42
$c106$	22.98	2.72	25.80	2.97	19.07	1.65	8.87	0.48	**9.04**	**0.39**
$c107$	16.64	1.76	19.37	2.21	14.03	1.20	**8.73**	**0.54**	10.00	0.57
$c108$	32.11	5.03	34.28	4.65	27.84	3.05	15.90	1.01	**15.64**	**0.85**
$c109$	70.44	21.13	58.71	12.37	63.35	11.06	39.55	3.57	**29.65**	**2.34**
$rc101$	9.92	0.23	16.29	0.70	6.19	0.15	**5.42**	**0.13**	7.46	0.19
$rc102$	56.39	4.33	41.89*	2.77*	48.15	2.75	**46.68**	**2.56**	31.40*	1.33*
$rc103$	331.06	182.51	124.35	24.99	277.25	94.08	257.62	74.68	**100.38**	**10.83**
$rc104$	-	-	232.14	93.65	-	-	1013.24	2355.03	**182.59**	**34.67**
$rc105$	32.25	1.67	42.00	3.69	26.43	1.08	**25.69**	**1.03**	29.92	1.34
$rc106$	26.83	1.08	38.22	2.91	22.44	0.76	**21.59**	**0.71**	26.95	1.02
$rc107$	128.75	18.88	127.96	21.16	114.46	12.95	113.12	10.91	**107.04**	**10.32**
$rc108$	284.19	100.13	223.03	72.62	254.82	62.64	250.77	52.10	**190.37**	**33.86**
$r101$	9.96	0.24	30.10	2.85	4.08	0.08	**1.86**	**0.05**	2.66	0.07
$r102$	268.04	455.46	192.05*	676.26*	187.85	152.03	**165.44**	**105.60**	89.49*	76.10*
$r103$	-	-	580.34*	3383.36*	-	-	-	-	375.53*	847.43*
$r104$	-	-	-	-	-	-	-	-	-	-
$r105$	32.35	2.15	79.55	26.84	13.14	0.50	**8.39**	**0.32**	10.85	0.51
$r106$	361.49	1957.72	258.61*	1365.62*	252.39	659.79	**221.34**	**563.77**	127.36*	171.22*
$r107$	-	-	-	-	-	-	-	-	250.37*	1364.76*
$r108$	-	-	-	-	-	-	-	-	-	-
$r109$	77.27	11.18	124.72	57.29	45.36	3.40	**28.16**	**2.07**	35.72	3.38
$r110$	609.45	1034.56	454.91	773.58	528.69	620.33	388.87	423.37	**284.19**	**241.75**
$r111$	617.17	1218.69	408.33	996.71	538.91	709.02	398.09	509.14	**268.08**	**323.70**
$r112$	-	-	-	-	-	-	-	-	**788.28**	**3323.50**

Table 4 C100, RC100 and R100: basic version, ng-routes, completion bounds and ng-routes with bounds. Second simulated column generation environment.

	basic		ng-routes		bound load		bound time		ng+bounds	
	Labels	Time	Labels	Time	Labels	Time	Labels	Time	Labels	Time
*c*101	6.26	0.30	8.07	0.30	4.15	0.17	**3.04**	**0.10**	3.60	0.12
*c*102	262.03	993.25	41.29*	5.61*	232.83	516.59	**129.38**	**64.23**	22.86*	2.10*
*c*103	-	-	129.01*	82.20*	-	-	-	-	96.60*	26.76*
*c*104	-	-	133.37*	229.87*	-	-	-	-	104.75*	80.61*
*c*105	9.80	0.58	12.32	0.67	7.69	0.41	**5.69**	**0.22**	6.75	0.25
*c*106	16.49	1.41	18.35	1.31	13.99	0.91	**8.88**	**0.38**	10.74	0.44
*c*107	10.33	0.78	12.41	0.81	8.78	0.58	**6.56**	**0.32**	7.36	0.34
*c*108	23.15	2.71	24.61	2.18	20.09	1.70	**13.79**	**0.79**	15.04	0.78
*c*109	46.66	10.22	36.66	5.27	43.07	6.25	32.68	2.81	**26.32**	**2.00**
*rc*101	9.22	0.25	14.67	0.63	4.92	0.13	**4.04**	**0.12**	5.29	0.16
*rc*102	63.35	5.80	50.94	4.69	42.70	2.34	39.28	1.96	**33.93**	**2.67**
*rc*103	233.18	97.14	135.98*	28.09*	**179.02**	**42.10**	175.89	43.30	105.43*	13.47*
*rc*104	-	-	231.74	85.70	1254.22	2238.00	1077.48	1492.99	**185.34**	**37.29**
*rc*105	23.85	0.89	33.49	1.98	18.26	0.54	**17.52**	**0.53**	21.51	0.73
*rc*106	21.40	0.81	31.36	1.76	16.33	0.51	**15.46**	**0.51**	19.52	0.68
*rc*107	78.47	8.19	88.11	11.74	68.98	5.24	**67.85**	**4.96**	71.19	5.23
*rc*108	161.45	39.18	135.66	24.69	140.67	19.64	135.28	19.30	**107.70**	**10.69**
*r*101	5.49	0.11	14.35	0.65	3.24	0.07	**1.94**	**0.05**	2.73	0.07
*r*102	509.03	1610.95	264.09	672.55	415.35	873.40	341.72	547.88	**177.42**	**201.77**
*r*103	-	-	388.42*	1165.05*	-	-	-	-	219.83*	232.07*
*r*104	-	-	-	-	-	-	-	-	-	-
*r*105	15.31	0.56	36.57	4.35	8.11	0.24	**6.22**	**0.19**	8.18	0.29
*r*106	-	-	411.79*	1755.61*	-	-	714.61	2795.80	289.07*	652.41*
*r*107	-	-	519.23*	2438.46*	-	-	-	-	304.52*	571.81*
*r*108	-	-	-	-	-	-	-	-	-	-
*r*109	64.95	13.62	107.91	72.88	48.34	5.45	**41.61**	**4.12**	52.29	6.84
*r*110	379.97	612.04	335.82	628.84	332.98	317.98	290.90	200.27	**237.60**	**146.19**
*r*111	415.15	640.51	339.85	546.87	370.74	274.64	332.89	270.65	**250.83**	**178.28**
*r*112	-	-	808.89*	3512.37*	-	-	-	-	645.10*	1889.91*

Table 5 C100, RC100 and R100: basic version, ng-routes, completion bounds and ng-routes with bounds. Third simulated column generation environment.

	basic		ng-routes		bound load		bound time		ng+bounds	
	Labels	Time	Labels	Time	Labels	Time	Labels	Time	Labels	Time
$c101$	10.74	0.56	14.58	0.68	6.35	0.25	**5.26**	**0.20**	6.28	0.25
$c102$	189.41	212.56	43.98	4.73	158.19	126.25	89.23	18.95	**23.85**	**1.50**
$c103$	-	-	145.53*	120.09*	-	-	-	-	88.56*	37.24*
$c104$	-	-	250.34*	347.26*	-	-	-	-	177.54*	130.94*
$c105$	14.92	0.81	19.44	1.11	11.21	0.53	**7.25**	**0.28**	8.65	0.31
$c106$	24.54	1.81	28.38	2.02	20.27	1.25	10.23	0.42	**9.86**	**0.41**
$c107$	16.19	1.03	20.00	1.29	13.48	0.75	**9.10**	**0.39**	10.46	0.50
$c108$	33.56	4.07	35.73	3.97	28.20	2.55	15.52	0.84	**13.28**	**0.58**
$c109$	67.46	17.71	54.62	9.49	59.27	12.37	36.37	3.24	**22.81**	**1.65**
$rc101$	8.72	0.18	14.43	0.50	5.70	0.12	**5.05**	**0.12**	7.06	0.19
$rc102$	81.43	10.20	42.28*	2.91*	**69.06**	**6.66**	65.75	6.87	31.21*	1.61*
$rc103$	277.02	170.70	94.47*	11.93*	228.45	118.19	**213.85**	**102.08**	69.46*	5.94*
$rc104$	-	-	202.47*	54.83*	1212.49	2985.51	**1078.13**	**2329.54**	169.58*	32.54*
$rc105$	26.29	1.04	34.30	2.21	20.42	0.70	**19.25**	**0.68**	23.30	0.94
$rc106$	23.22	0.75	31.87	1.56	17.90	0.51	**16.92**	**0.51**	21.42	0.72
$rc107$	105.39	10.72	110.37	13.92	90.57	8.16	**87.71**	**7.46**	88.95	8.26
$rc108$	198.30	41.37	163.56	25.93	175.65	31.02	169.05	25.43	**135.59**	**18.03**
$r101$	8.48	0.28	22.52	2.25	3.79	0.09	**1.50**	**0.05**	2.01	0.07
$r102$	-	-	235.24*	1178.80*	291.00	1939.15	**193.89**	**891.05**	117.96*	266.17*
$r103$	-	-	-	-	-	-	-	-	343.92*	3242.89*
$r104$	-	-	-	-	-	-	-	-	-	-
$r105$	26.81	1.96	59.52	15.06	12.38	0.61	**5.68**	**0.29**	7.25	0.45
$r106$	-	-	278.71*	1272.39*	332.47	2962.30	**217.65**	**1046.66**	134.17*	244.67*
$r107$	-	-	-	-	-	-	-	-	284.58*	2133.11*
$r108$	-	-	-	-	-	-	-	-	-	-
$r109$	82.11	32.52	129.89	196.64	51.20	11.44	**26.56**	**4.59**	36.29	9.02
$r110$	653.41	1482.01	456.48	1007.39	563.37	1211.10	400.06	791.16	**260.14**	**332.05**
$r111$	848.99	3269.41	615.19	3239.52	730.46	2548.20	481.84	1353.81	**376.60**	**1195.76**
$r112$	-	-	-	-	-	-	-	-	-	-